POLITECNICO DI TORINO

III Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea Magistrale

Sviluppo di una tecnica per mitigare la sensibilità alle radiazioni di un processore commerciale per satellite modulare AraMiS

Relatori:
Prof. Leonardo Reyneri
PhD. Maurizio Tranchero
dott. Stefano Speretta

Candidato:
Tomás A. Alarcón Rivera

Novembre 2009

Sommario

Il progetto AraMiS (Architettura Modulare per Satelliti) ha inizio nell'autunno del 2006 sulla base dell'esperienza appurata dal progetto PiCPoT (Piccolo Cubo del Politecnico di Torino), il primo nanosatellite realizzato dal Politecnico di Torino, che ha così aderito ad un'iniziativa internazionale, dall'elevata valenza didattica, che concerne la progettazione di un satellite in ambito universitario. La filosofia progettuale standard adottata da diverse università in tutto il mondo si basa sul concetto di Cubesat, un satellite di forma cubica con un lato di circa 10 cm e una massa di un 1 Kg al massimo.
PiCPoT è un piccolo cubo di 13 cm di lato contenente, all'interno, sottosistemi elettronici sviluppati ad hoc per svolgere le funzioni di trasmettere dati (per esempio, le misure dei sensori di bordo alla Stazione di Terra), scattare fotografie, valutare il funzionamento del GPS in orbita Low Earth Orbit (LEO) e quello dei componenti Commercial Off-The-Shelf (COTS) nello spazio.
Per proseguire l'attività iniziata col progetto precedente, è nata l'idea di base del progetto AraMiS che si sviluppa seguendo il concetto di modularità. Quest'ultimo si compone di un discreto numero di moduli, detti 'tiles', preassemblati e precollaudati che hanno la caratteristica di poter essere riutilizzati a seconda delle esigenze di ciascuna missione. Ciò permette una realizzazione low cost e tempi di sviluppo molto ristretti per il progetto.
I moduli, comunicanti tra di loro tramite un bus seriale, hanno dimensione standard; alcuni saranno collocati sulla superficie esterna del satellite, altri al suo interno.
Esistono diverse tecniche a livello commerciale per mitigare gli effetti alle radiazione nei dispositivi elettronici, ma solitamente sono tecniche applicate nei processi d'integrazione e di produzione, e che implicano forti incrementi nel costo finale dei dispositivi o del sistema complessivo. Lo scopo principale del progetto AraMiS è quello di creare soluzioni di basso costo, mantenendo sempre qualità ed efficienza. Sono questi i motivi che ci hanno portato a produrre, attraverso la progettazione ingegneristica, nuove e innovative soluzioni che abbiano elevati standard; per un progetto importante come quello di AraMiS.

Questo documento si compone di 5 capitoli, che spiegano la base teorica e lo sviluppo della tecnica Smart Wacth Dog.

A mia mamma, perché tutto quello che sono lo devo a te!,
ed a mio nonno per essere stato sempre una guida ed un gran esempio.
A mi madre, porque todo lo que soy te lo debo a ti!
y a mi abuelo porque siempre fuiste un guía y un ejemplo a seguir.

Ringraziamenti

In questa sezione devo scrivere in due lingue diverse per esprimere la mia gratitudine.

En primer lugar debo agradecerles a ti madre y a ti hermano por haberme dado todo su apoyo en todos estos años de estudios profesionales, es por ustedes y para ustedes que he realizado esta aventura de convertirme en ingeniero y en dottore magistrale. Han sido años de esfuerzos y sacrificios, que gracias a la vida los hemos podido compartir juntos, evolucionando y aprendiendo a vivir en el proceso. Ustedes han sido no solo un apoyo sino un motor para recorrer este camino. Mamá tu has sido mi guía mi confidente y mi amiga eterna, lo seguirás siendo por los siglos de los siglos. Luis son tus locuras, tu alegría de vivir y ese apoyo incondicional los que me han permitido tener muchos motivos para superar tantos escollos durante estos años. Mi gratitud se extiende a toda mi familia, Mili, Ceci, Lilian, Zoraida, Luisa, Abraham —Ustedes has sido un gran apoyo no solo durante mis estudios sino a lo largo de toda mi vida— y Luis Alfredo —Tío espero que donde quiera que estés recibas un fuerte abrazo y todo mi amor—. Te agradezco a ti Abuelo que siempre estuviste presente y siempre fuiste el padre y el amigo, se que desde tu puesto al lado del arquitecto mayor nos observas y nos cuidas, en fin, porque son todos ustedes la razón de mi vida y mis metas los amo con todo el corazón.

Por otra parte quiero nombrar a ustedes hermanos y hermanas que me busque en este camino de evolución personal y profesional, que con su apoyo y comprensión me ayudaron a convertirme en Ingeniero, ya que los 5 años en la Universidad Central de Venezuela, fueron los que me permitieron llegar aquí, ustedes son parte de mi familia y los amo de la misma forma, Mariana, Jose Argenis, Ernesto, Leoner, Juan Carlos, Barbara, Geraldy, Manuel, Luis Daniel, Guillermo, Francisco y Andrea les doy las gracias por haberme acompañado, apoyado y comprendido durante todos estos años.

De igual forma extiendo mi gratitud ha esta nueva familia venezolana que me di Italia Ani —Hermanita torinense que me has acompañado y apoyado durante estos dos años—, Carlitos —eres un hermano para mi le doy gracias a las casualidades por haberte conocido—, Patricia, Pedro, Elka, Jonnahtan y Elyka —porque apareciste

en el momento justo y te has convertido en parte indispensable de mi vida— ustedes a lo largo de estos dos años de aventura Italiana han sido de gran apoyo y un motivo de felicidad para mi, me han dado la fuerza y las ganas de seguir, por más difícil que haya sido este camino, me dieron cada uno a su tiempo lo que me hacia falta para no perder el rumbo y la meta que vine a cumplir, es también de ustedes esta 'Laurea'.

Poi devo ringraziarvi voi l'ala europea della mia famiglia torinese, siete stati un grande appoggio, ed ho molti motivi per considerarvi come participi di questa tesi e di questa laurea, con voi ho condiviso cose molto importanti, e siete stati un altro motivo di gioia e sviluppo per me, nel percorso di questi due anni, Jacopo—Sei diventato un gran amico, e te ringrazio per l'aiuto che mi hai proporzionato per mantenere la concentrazione in questa tesi—, Carlo, Valentina—Sei un angelo che ho conosciuto per caso, e sei diventata una gran amica—, Helen and Fey—you are both the best british friends that i have, thanks for all—, Marco, Stefano, Jakson, Lele, Tony ed il Mauro, ogni uno di voi consapevole o no, avete collaborato in diversi modi nei miei studi, nella mia vita qua in Italia e nello svolgimento di questa Tesi.

Para finalizar agradezco a la Universidad Central de Venezuela por haberme dado la oportunidad de hacer los estudios de doble titulación, e della stessa maniera stendo la mia gratitudine al Politecnico di Torino per avermi presso nel suo ateneo per questa doppia laurea.

Como dije antes es también por ustedes que este fin es posible. Come ho detto prima è anche per voi che questo fine è possibile.

Gracias, Grazie.

Indice

Elenco delle abbreviazioni

API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
BSL	Bootstrap Loader
CB	Complementary Bipolar
COTS	Commercial Off-The-Shelf
FPGA	Field-Programmable Gate Array
CCD	Charge-Copuled Devices
IDE	Identifier Extension
LEO	Low Earth Orbit
LUT	Look-Up Table
OBC	On-Board Computer
PDB	Power Distribution Bus
SAA	South Atlantic Anomaly
SEE	Single Event Effects
SEU	Single Event Upset
SEL	Single Event Latch-up
SEB	Single Event Burnout
SEFI	Single-Event Functional Interrupt
SEGR	Single-Event Gate Rupture

SWD	Smart Watch Dog
TBD	To Be Define
TID	Total Ionizing Dose
TSI	Total Solar Irradiance
UML	Unified Model Language

Capitolo 1

Introduzione

1.1 Il progetto PiCPoT

Nell'Autunno del 2003, nacque il progetto (PiCPoT) acronimo di "Piccolo Cubo del Politecnico di Torino", al fine di costruire un nanosatellite, nell'arco di un anno, a scopo educativo e di ricerca. Il progetto fu basato sui seguenti requisiti:

- Forma cubica con lato di 13 cm;
- Massa inferiore a 5 Kg;
- Potenza media non superiore a 1,5W;
- Almeno 90 giorni di vita;
- Utilizzo di componenti COTS nello spazio;
- Compatibilià con il lanciatore POD;

Esternamente il satellite si presentava come in figura 1.1.

Le principali funzioni del satellite riguardavano: l'acquisizione di misure di temperatura, illuminamento, scattare fotografie e trasmettere i dati raccolti alla stazione di terra.

Su cinque delle sue sei facce esterne, erano presenti celle solari utilizzate per convertire l'energia solare in energia elettrica; sulla sesta faccia erano presenti due antenne (437MHz e 2.4GHz), tre fotocamere, due kill switch (per 'spegnere' il satellite durante il lancio) e un conettore di collaudo, per verificare il corretto funzionamento dell'elettronica di bordo. Internamente era alimentato da sei gruppi di batterie ricaricabili disposte fra i pannelli solari e le schede elettroniche. Su PiCPoT erano presenti tre processori:

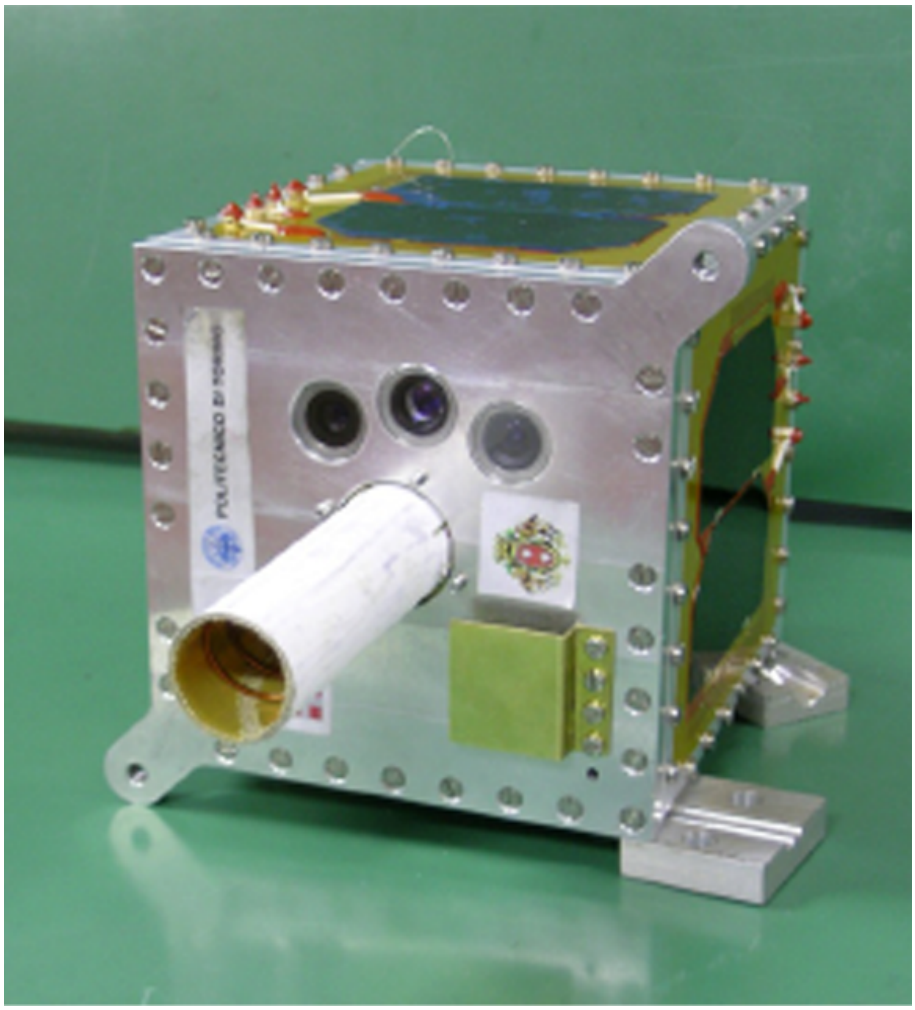


Figura 1.1: ***Nano Satellite PiCPoT***

- Processore 1: per la gestione di bordo, associato al canale di comunicazione a 437MHz, aveva una frequenza di clock a 11 MHz e una tensione di alimentazione di 3,3 V.

- Processore 2: utilizzato per la gestione di bordo associato al canale di comunicazione a 2,4 GHz. Per questo scopo è stato utilizzato un μprocessore MSP430 a basso consumo, con frequenza di clock a 4MHz e tensione di alimentazione di 3,3 V.

- Payload: dedicato all'acquisizione delle immagini delle tre telecamere, che successivamente sono inviate a terra tramite i processori 1 e 2.

I due processori ProcA e ProcB erano indipendenti fra loro e condividevano tutti i circuiti di condizionamento dei sensori, ma ciascuno disponeva di un suo multiplexer.
Tutti e tre i processori erano alimentati periodicamente ed eseguivano sempre lo stesso codice.
La comunicazione a terra avveniva tramite due canali, entrambi in banda amatoriale con protocollo APRS.

1.2 Il progetto AraMiS [1]

L'architettura AraMiS è pensata per lo sviluppo di micro e nano satelliti per orbite tra i 500 e gli 800 km di quota (LEO, Low Earth Orbit).
L'idea alla base dell'architettura è la realizzazione delle strutture meccaniche e delle strutture elettroniche di base di un satellite tramite l'assemblaggio di moduli prefabbricati. I moduli sono progettati per essere assemblati e utilizzati nella maniera più semplice e flessibile, permettendo di ridurre costi e tempi di sviluppo.
Ogni modulo sarà inoltre completamente caratterizzato e collaudato, lasciando all'utilizzatore il solo compito della realizzazione e del testing del payload.
I moduli, o tile, che compongono l'architettura presentano delle caratteristiche comuni:

- Dimensione di 16, 5 X 16,5 cm;
- struttura portante realizzata con una lastra di alluminio alodinato o anodizzato;
- fori di passaggio sull'intero perimetro con passo di 11,75 mm, per garantire un'elevata rigidita e un ottimo isolamento elettromagnetico a frecuenze di 2,4 GHz o superiori.

Sono attualmente definiti diversi tile del satellite, tra quelle le più importanti: 'Power Management Tile' e 'Comunication Tile'.

La Power Management Tile è costruita su di una lastra di alluminio di 1.5 mm di spessore e presenta sulla faccia esterna (rivolti verso lo spazio) i pannelli solari in GaAs a tripla giunzione che forniscono l'energia al satellite (vedi figura 1.2). Sulla faccia interna sono invece fissate, tramite una struttura metallica, le due batterie che immagazzinano l'energia e la ruota di inerzia con il relativo motore brushless per il controllo di assetto (vedi figura 1.3). Al di sotto della struttura metallica, sono incollati con colle termoconduttive alla lastra di alluminio il circuito stampato principale del modulo e il solenoide del controllo di assetto. Sul circuito stampato saranno presenti i circuiti switching di conversione dell'energia proveniente dai

pannelli solari, i circuiti di carica delle batterie, i sensori per la determinazione dell'assetto del satellite e i microcontrollori che gestiscono il funzionamento di tutti i sottosistemi.

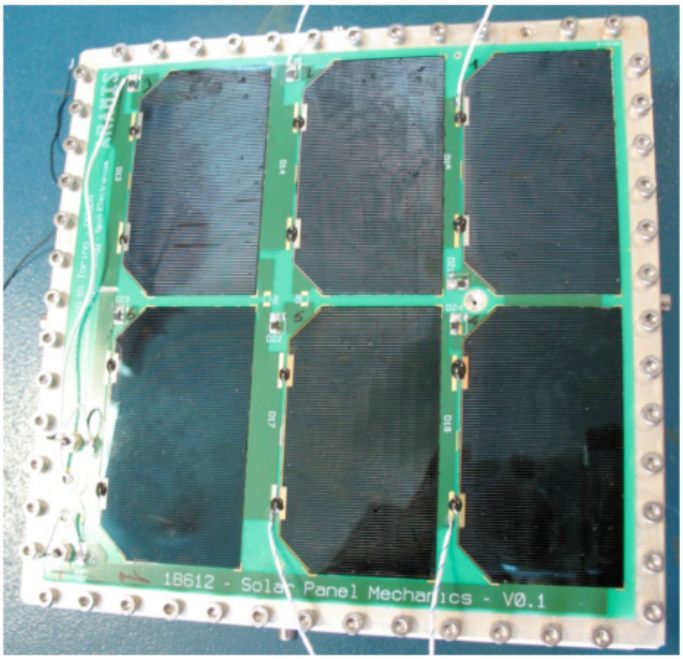


Figura 1.2: Faccia esterna della Power Management Tile [8].

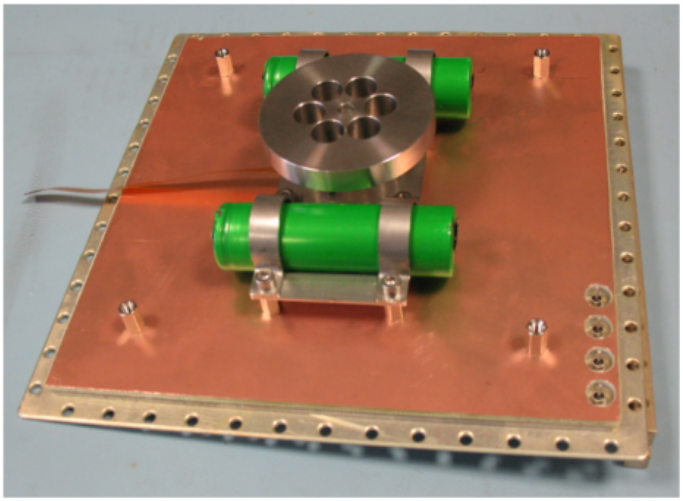


Figura 1.3: Faccia interna della Power Management Tile [8].

Ogni modulo di power management sarà quindi indipendente, in grado di alimentarsi autonomamente tramite i propri pannelli solari e di caricare le proprie batterie. Ma sarà anche in grado di interagire e scambiare energia con gli altri moduli del satellite tramite il Power Distribution Bus (PDB). Questo permetterà, ad esempio, di caricare le batterie di un pannello in ombra sfruttando i pannelli solari di un modulo illuminato. Oppure potrà permettere di dissipare l'energia in eccesso generata da un lato illuminato su un carico di shunt su di un lato in ombra (quindi più freddo).

Oltre alla gestione dell'energia, la power management tile si occupa anche della determinazione e del controllo di assetto. Contiene infatti un sensore di campo magnetico in grado di misurare il campo terrestre e un sensore di sole in grado di determinare l'orientamento del modulo rispetto alla stella. La correlazione di questi dati con una conoscenza più approssimativa dell'orbita del satellite, consente di determinare con precisione il punto e l'assetto del satellite. Tramite la ruota d'inerzia ed il solenoide, sarà poi possibile effettuare delle correzioni o delle variazioni intenzionali dell'assetto del satellite.

Il solenoide consente infatti di generare un campo magnetico che, interagendo con quello terrestre, determinerà una rotazione del satellite. La rotazione potrà essere piuttosto lenta e richiederà una discreta quantità di energia, ma sarà permanente nel tempo. Questo sistema verrà quindi impiegato principalmente per le correzioni di assetto a lungo termine e per il *detumbling* iniziale dopo la separazione dal vettore.

La ruota di inerzia permette invece, con un'accelerazione del motore, la generazione di una coppia che ruota in maniera relativamente veloce il satellite. La rotazione viene però mantenuta solo con una rotazione continua del motore: se il motore decelera infatti, il satellite torna nella posizione originale. Questo sistema è quindi utile soprattutto per orientare temporaneamente il satellite in una certa direzione per esigenze, ad esempio, del payload.

Il modulo di power management, essendo composto da molti sottosistemi, svolgerà anche una sostanziale funzione di housekeeping, con l'acquisizione e la segnalazione all'On-Board Computer (OBC) dei valori istantanei di molte grandezze come le correnti e le tensioni dei pannelli solari, le temperature dei componenti più critici, ecc. L'OBC comunicherà inoltre con il modulo per l'invio dei comandi di gestione dell'energia e di controllo d'assetto.

La Telecommunication Tile è invece costruita con una lastra di alluminio più spessa (5 mm) e rappresenta il punto preferenziale di ancoraggio del satellite al vettore. Presenta sulla faccia esterna le due antenne (a 437 MHz e 2.4 GHz) per la comunicazione verso terra e l'eventuale struttura di ancoraggio (vedi figura 1.4).

L'ancoraggio è in realtà considerato in maniera separata dalla telecommunication tile in quanto fortemente dipendente dal vettore e dal tipo di lancio. La disposizione delle antenne sulla faccia esterna lascia inoltre uno spazio centrale che può essere utilizzato per il posizionamento di una fotocamera di osservazione terrestre, essendo il modulo di telecommunication quello generalmente puntato verso terra.

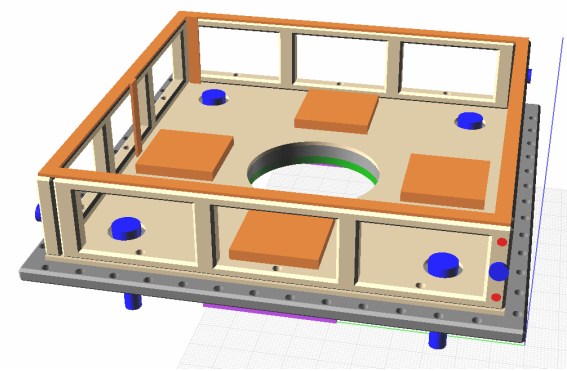


Figura 1.4: Faccia esterna della Telecommunication Tile [8].

Sul lato interno trovano invece spazio i moduli a radiofrequenza nelle due bande e due OBC ridondati (vedi figura 1.5). La Telecommunication Tile è il cuore dell'intero sistema e interagisce con le Power Management Tile attraverso l'invio dei comandi di controllo e la ricezione dei dati di telemetria. Gestisce inoltre le comunicazioni verso terra attraverso la ricezione e l'attuazione dei telecomandi e la trasmissione in downlink dei dati di telemetria o dei dati un eventuale payload.

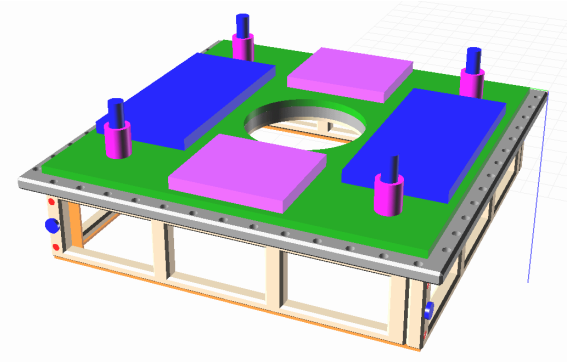


Figura 1.5: Faccia interna della Telecommunication Tile [8].

Per ridurre peso e costi del satellite, i moduli sono parte integrante della struttura meccanica. Nella configurazione minima, il satellite è formato da sei moduli che

costituiscono le facce esterne di un cubo e che racchiudono un volume di circa 1 dm^3, disponibile per il payload. I moduli sono connessi meccanicamente grazie a delle semplici barrette su cui vengono avvitati i lati dei moduli a formare gli spigoli del cubo. In figura 1.6 si può osservare un esempio di cubo minimo: si noti la presenza sulla faccia inferiore della telecommunication tile e l'utilizzo delle barrette forate per il fissaggio dei moduli. Lo spazio vuoto all'interno del cubo è interamente disponibile per il payload.

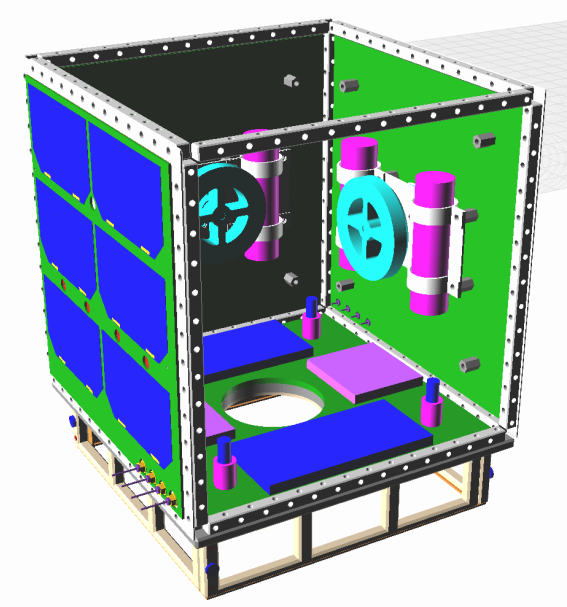


Figura 1.6: Vista interna del cubo minimo [8].

Oltre al cubo minimo, con la realizzazione di un opportuno telaio è possibile disporre i moduli in strutture più complesse, che meglio si adattino alle dimensioni e alla forma del payload. In figura 1.7 si può osservare un cubo con due moduli per lato (si noti la presenza di due Telecommunication Tile per offrire maggiore ridondanza o maggior guadagno) e in figura 1.8 si può osservare l'ipotesi di una struttura a prisma esagonale utilizzabile, ad esempio, con un payload costituito da un telescopio ottico.

1.2.1 Requisiti di sistema e specifiche

L'architettura per satelliti AraMiS è formata, come già visto, da due tipi di mattonelle modulari: la power management tile e la telecommunication tile.

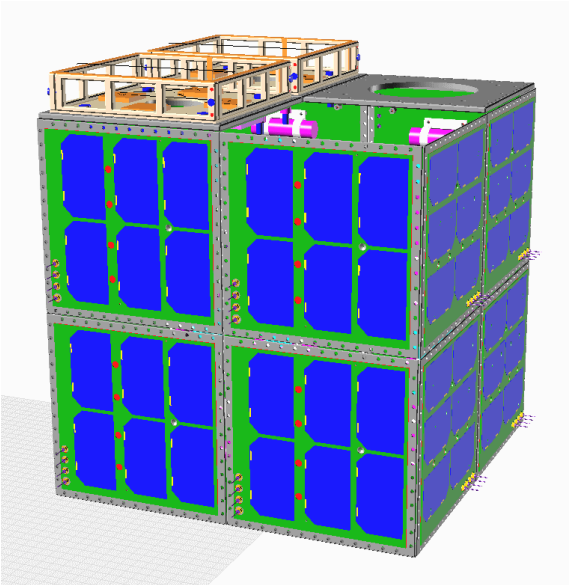


Figura 1.7: Struttura cubica con due moduli per lato [8].

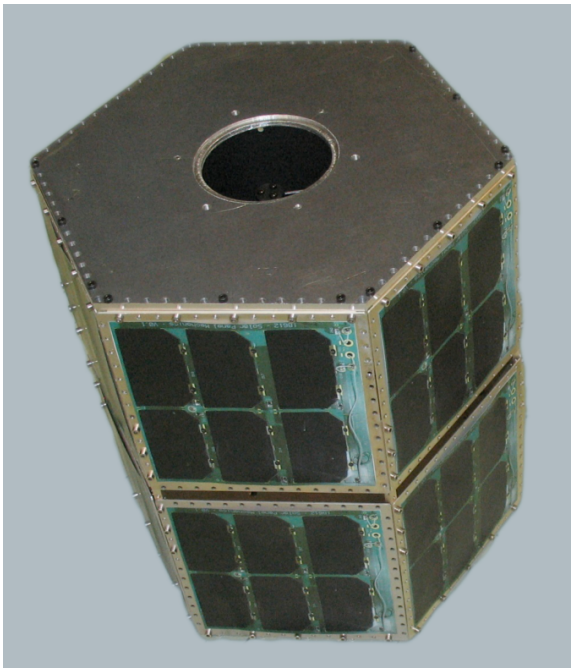


Figura 1.8: Struttura a prisma esagonale per payload tipo telescopio ottico [8].

Al loro interno sono contenuti moduli diversi, che hanno il compito di garantire l'operatività del satellite e del payload trasportato per il corretto completamento della missione. Le funzioni di cui si fa carico l'architettura AraMiS sono piuttosto

diversificate e possono essere riassunte tramite il diagramma UML delle classi di figura 1.9.

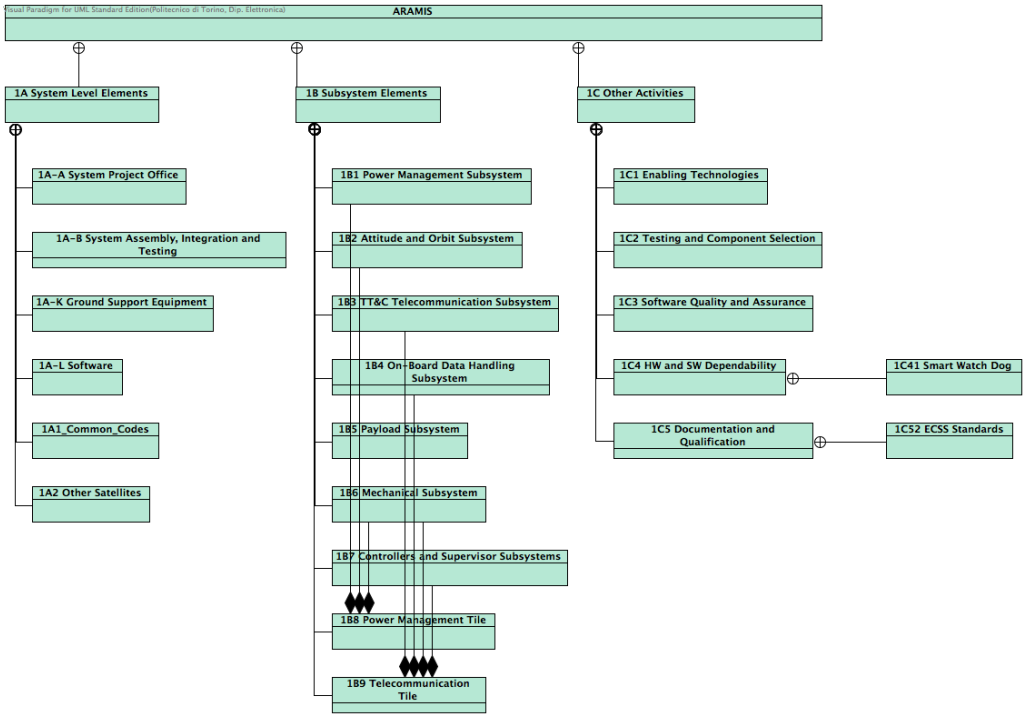


Figura 1.9: Diagramma delle classi del progetto AraMiS.

Il diagramma non racchiude al suo interno le sole funzioni svolte dal satellite, ma copre un campo di attività più ampio svolte all'interno del progetto AraMiS. Sono ad esempio indicati gli elementi a livello sistema, come come le procedure per l'assemblaggio, l'integrazione e il testing (*1A-B System Assembly, Integration and Testing*) o i dispositivi necessari a terra per portarle a termine (*1A-K Ground Support Equipment*).

Scendendo però più nel dettaglio, il diagramma definisce anche le attività svolte dall'architettura modulare. Troviamo quindi, all'interno della classe *1B Subsystem Elements*, la definizione dei sottosistemi che fanno parte dell'architettura con le relazioni gerarchiche che li collegano.

Alcuni di questi elementi sono puramente *hardware*, come *1B6 Mechanical Subsystem*, che definisce le caratteristiche e le specifiche della struttura costruttiva del satellite. Altri rappresentano invece un'insieme di funzioni diverse che verranno raggruppate fisicamente in un'unica entità, come nel caso di *1B8 Power Management*

Tile e *1B9 Telecommunication Tile*.

Come si osserva dal diagramma, la power management tile contiene al suo interno *1B1 Power Management Subsystem* e *1B2 Attitude and Orbit Subsystem*, oltre a *1B6 Mechanical Subsystem* per ovvie ragioni costruttive.

Il sistema di power management è formato dai pannelli solari esterni alla mattonella e dai circuiti interni di controllo e conversione dell'energia, che si occupano della carica delle batterie e della distribuzione, alle altre mattonelle dell'architettura AraMiS e al payload, della potenza elettrica disponibile.

Il sistema di attitude and orbit control è formato invece da una serie di sensori, che permettono di determinare a bordo la posizione e l'assetto del satellite, e da una coppia di attuatori, che permettono entro certi limiti di effettuare correzioni dell'assetto stesso.

Entrambi i sistemi, presenti su ogni mattonella di power management, sono in grado di svolgere autonomamente le funzioni di base per cui sono stati costruiti. Con l'integrazione di più mattonelle all'interno di uno stesso satellite però, è necessario un coordinamento tra l'insieme dei sistemi di attitude control e di power management per sfruttare al meglio le risorse disponibili in ambiente spaziale.

È quindi necessaria la presenza di un controllore centrale, l'On-Board Computer (OBC), che regoli i diversi sistemi. Questo processore centrale, nell'architettura AraMiS, è localizzato nella telecommunication tile (*1B9 Telecommunication Tile*), all'interno di *1B4 On-Board Data Handling Subsystem*, di cui si può osservare il contenuto in figura 1.10.

Nel diagramma troviamo anche altri elementi, tra cui *1B43 Housekeeping Management*, *1B44 Telecommand Management* e *1B41 On-Board Data Bus*.

Il sistema di *housekeeping* si occupa del controllo di tutti i parametri interni del satellite e del controllo dello stato di funzionamento di ogni singola componente. Questo comprende il monitoraggio delle varie temperature interne, delle correnti dei pannelli solari, delle tensioni delle batterie e di ogni altra grandezza che possa determinare lo stato di *salute* del sistema. Queste informazioni vengono misurate da vari sensori posizionati nei punti più critici del sistema, raccolte periodicamente e inviate a terra per il monitoraggio remoto dello stato del satellite.

Il telecommand management si occupa invece di *ricevere* istruzioni da terra che controllano il comportamento del satellite. Il controllo può riguardare ad esempio

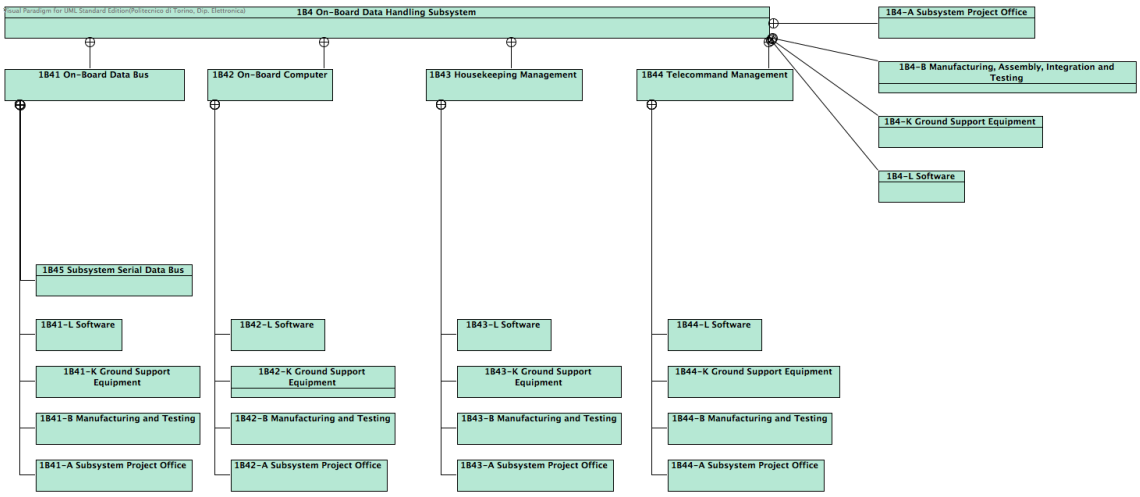


Figura 1.10: Diagramma delle classi di 1B4 On-Board Data handling Subsystem

l'attivazione dei sistemi di controllo d'assetto o l'esecuzione di procedure che, per la loro criticità, non possono essere attivate autonomamente dai controllori a bordo. Procedure, anche queste, che vanno comunque ad intervenire su sistemi diversi all'interno del satellite e comportano uno scambio di informazione tra moduli diversi.

L'on-board data bus è, infine, l'entità che consente tutte le interazioni fin qui elencate. Considerando però che le funzioni delle classi *1B43* e *1B44* verranno svolte internamente all'OBC, l'on-board data bus dovrà, in definitiva, supportare le seguenti comunicazioni:

- invio di comandi dall'OBC alle power management tile e ricezione delle risposte;
- invio dalle power management tile all'housekeeping management dei dati di housekeeping.

Gli ulteriori requisiti imposti dall'architettura AraMiS sono la flessibilità, la modularità e la semplicità dell'intero sistema. Questo pone vincoli sulla complessità delle topologie di interconnessione utilizzabili e sul mezzo di comunicazione utilizzato. Per garantire poi una maggiore flessibilità consentendo il collegamento in serie dei bus di potenza, potrebbe essere desiderabile un qualche livello di isolamento galvanico tra i vari nodi della comunicazione.

Per quanto riguarda specifiche di sistema più generali, nell'architettura AraMiS è previsto l'utilizzo quasi esclusivo di processori MSP430 della Texas Instruments. Questi processori sono stati scelti principalmente per la presenza in letteratura di

dati di affidabilità che ne certificano il buon funzionamento fino a Total Ionizing Dose (TID) di 30 krad(Si) [**?**]. L'architettura MSP presenta inoltre interessanti caratteristiche volte alla riduzione dei riduzione dei consumi e alla flessibilità, con la disponibilità di un vasto insieme di periferiche integrate nei vari modelli di processore. I test si riferiscono alla prima serie dei processori (MSP430x1xx), ma nell'architettura AraMiS si prevede di utilizzare principalmente processori della seconda e la quarta serie (MSP430x2xx, MSP430x4xx) , in quanto più veloci e contenenti periferiche più avanzate. Come previsione ad una possibile tolleranza inferiore alle radiazioni, si è svolto una tecnica per mitigare la sensibilità alle radiazioni di un qualsiasi processore, avendo la possibilità anche di scegliere altri μ-controllore nel caso che si sia dimostrato il bisigno.

Le caratteristiche ambientali in cui i circuiti si troveranno a doperare sono invece quelle tipiche dell'ambiente spaziale, con assenza di atmosfera (e quindi di trasferimento di calore per convezione) e elevate dosi radiazioni e particelle ad alta energia. Da alcune prime stime, la temperatura interna del satellite non dovrebbe discostarsi di molto dal range $0 \div 70\,^{\circ}\mathrm{C}$ mentre una valutazione della total dose annuale è osservabile nel grafico di figura 1.11.

Le curve sono ricavate mediante i modelli del programma di simulazione SPENVIS considerando un'orbita polare di inclinazione 97° e in funzione di diversi valori di quota e di spessore della struttura meccanica. Come si può osservare, la total dose varia di molto a seconda dello spessore della struttura esterna e con i moduli dell'architettura AraMiS (considerato non solo l'alluminio, ma anche i circuiti stampati e i pannelli solari) la curva effettiva dovrebbe posizionarsi tra le due indicate. Per una quota di 800 km, la TID annuale dovrebbe quindi essere compresa tra 1.2 e 2.0 krad(Si).

1.3 Tecnica di mitigazione degli effetti radioattivi

Smart Wacth Dog

Esistono diverse tecniche a livello commerciale per mitigare gli effetti alle radiazione nei dispositivi elettronici, ma solitamente sono tecniche applicate nei processi d'integrazione e di produzione, e che implicano forti incrementi nel costo finale dei dispositivi o del sistema complessivo. Lo scopo principale del progetto AraMiS è quello di creare soluzioni di basso costo, mantenendo sempre qualità ed efficienza. Sono questi i motivi che ci hanno portato a produrre, attraverso la progettazione ingegneristica, nuove e innovative soluzioni che abbiano elevati standard; per un

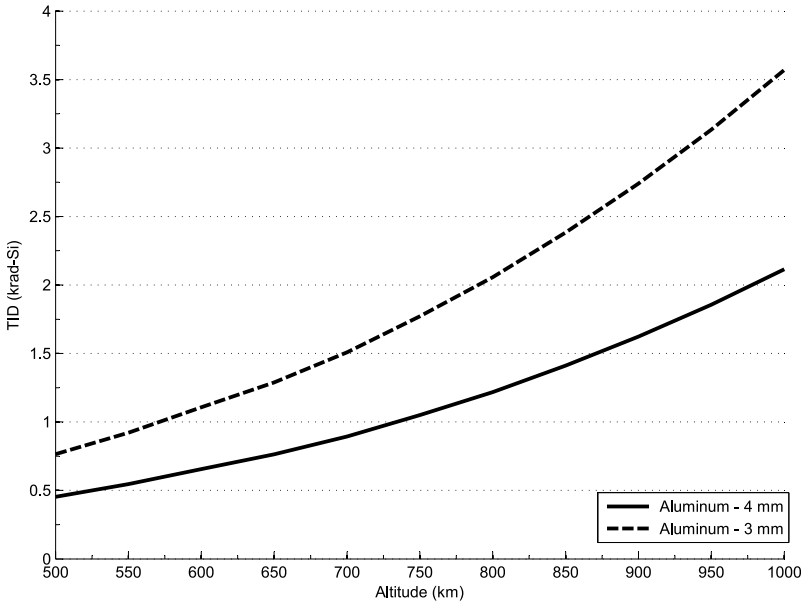


Figura 1.11: Livello di Total Ionizing Dose annuale in funzione della quota per un'orbita di inclinazione 97° e per due diversi spessori della struttura esterna del satellite. I modelli utilizzati tengono conto delle radiazioni provocate da elettroni, protoni e Bremsstrahlung.

progetto importante come quello di AraMiS.

La tecnica proposta permette l'utilizzo di un qualsiasi μ-processore commerciale, è stata sviluppata per dare più robustezza al sistema. Fa uso di componenti di basso costo ed è molto flessibile, è stata progettata come un sistema modulare. Tramite la conformazione di classi che possono essere modificate secondo le esigenze di futuri missioni.

Il sistema finale fa uso di un insieme di tool software e hardware, sviluppati tramite i linguaggi UML, C++ e VHDL. LA Smart Wacth Dog è stata chiamata così perché non è la semplice 'watch dog' che è contenuta all'interno di quassi tutti i processori disponibili in mercato, ma va oltre, facendo un controllo accurato dello stato della memoria operativa del μ-controllore a proteggere. Non si limita ad aspettare la comunicazione da parte del μ-processore, che solitamente non porta informazione del suo funzionamento ma soltanto azzera un timer per evitare il reset delle operazione. La Smart Wacth Dog è in grado di riprogrammare la memoria operativa, blocchi a blocchi e tutta complessa. Dovuta alla sua struttura permette l'aggiornamento del firmware del processore una volta che questo sia nello spazio, con l'unica condizione che al prodursi una catena di eventi sfavorevoli che possano modificare il contenuto

della memoria operativa, questa sara ripristinata col firmware originale, con il quale è stato messo in orbita, lasciando il lavoro di aggiornare nuovamente il software del μ-controllore alla stazione di terra, tramite i sistemi di comunicazioni a disposizione.

In figura 1.12 si può osservare il principio alla basse della tecnica sviluppata. In questo diagramma si evidenza la interazione tra il processore e la Smart Wacth Dog, la quale avviene periodicamente. In questo processo si controlla il blocco di memoria operativa che contiene il codice che sarà eseguito subito dopo. La comunicazione si produce tramite due unità che fanno uso del protocollo standard RS-232, il primo (a destra) serve per la ricezione ed invio di dati, il secondo serve per riprogrammare la memoria operativa nel caso che fosse necessario. Tramite la connessione diretta tra FPGA e CPU si può fare un reset nel caso che sia bisogno.

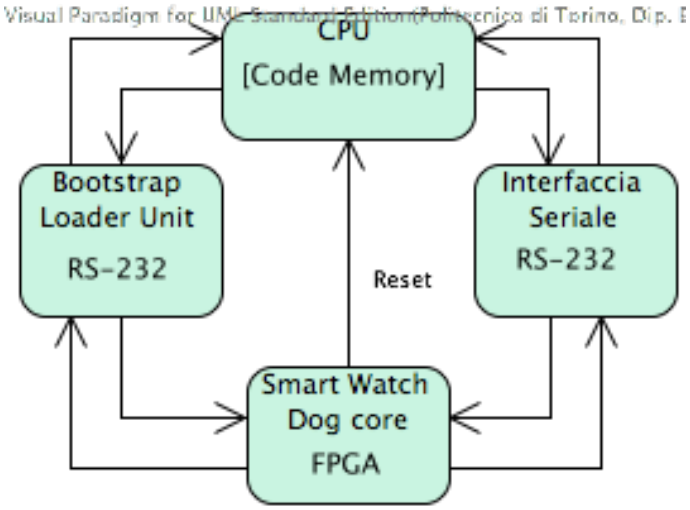


Figura 1.12: ***Principio di funzionamento della tecnica Smart Wacth Dog***

Capitolo 2

Effetti delle radiazioni sui componenti elettronici

Nella attualità i circuiti integrati, in sigla ICs (dal corrispondente termine inglese integrated circuits) sono presenti nella maggior parte delle attrezzature utilizzate nel mondo spaziale, aeronautico, automobilistico e delle telecomunicazioni, tutte queste applicazioni hanno bisogno di una accuratezza elevata nella risposta che offrono all'utente finale, in questi ambiti sono gli ICs quelli che implementano la logica ed il controllo, quindi sono essi che devono avere una accuratezza molto elevata nel loro funzionamento.

2.1 Le Radiazione nello spazio Circumterrestre

In tutti gli ambienti ci sono presenti le radiazioni, ma in particolare nelle applicazioni spaziali o aeronautiche (vuole dire quelle applicazioni a gran altezza oppure sopra l'atmosfera) sono più elevate e continue, invece quelle applicazioni a quota più bassa sono meno irradiate, perché l'atmosfera terrestre filtra molte delle radiazioni presenti nello spazio. Gli errori accadono quando particelle caricate di energia si scontrano con gli ICs, queste particelle solitamente provengono dai raggi cosmici che si generano all'interno del universo o nel nostro sole.

Nella figura 2.1 si presentano le cinture di radiazione di Van Allen della Terra, queste cinture indicano i tipi e la quantità delle radiazioni alle quali può essere sotto posto un oggetto nello spazio circumterrestre, vuole dire, fuori dell'atmosfera ma ancora sotto l'effetto della gravità del nostro pianeta.

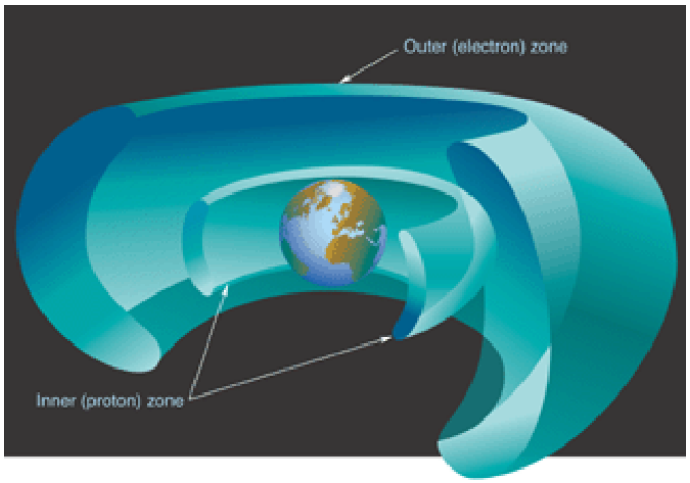


Figura 2.1: ***Diagramma delle cinture di radiazione di Van Allen della Terra [2, 12]***

Tra gli ICsche vengono più influenzate ci sono: le memorie SDRAM, i transistor NMOS, i circuiti basati sulle tecnologie bipolari e Schottky bipolari a bassa potenza; tra quelli meno suscettibile ci sono: i circuiti CMOS/SOS e i CMOS standard. Gli effetti vengono studiati separatamente e prendono il nome di "Effetti di Singoli Eventi",in acronimo SEE proveniente dall'inglese Single Event Effects, a loro volta sono suddivisi in Single Event Upset (SEU), Single Event Latch-up (SEL) e Single Event Burnout (SEB).

2.2 Single Event Efects

Un Single Event Effects (SEE) è il risultato della interazione di una singola particella di energia con un circuito. La possibilità di che accada un singolo evento è stata postulata per Wallmark e Marcus nel 1962[10]. La prima anomalia su un satellite moderno è stata segnalata da Binder nel 1975[3].

Ci sono dei SEE che si producono dovuti a particelle che non provengono dallo spazio, ma da lo interno del package del ICs, le particelle Alpha provengono di tràcce lasciate da concentrazioni di uranio e di torio presenti nel processo di packaging del ICs. Sono stati May e Woods pionieri delle ricerche sugli effetti che producono questo tipo di particelle nelle memorie dinamiche[6].

2.2.1 Sigle Event Upset

Questi eventi sono definiti per la NASA come "errori indotti dalle radiazioni nei circuiti microelettronici, causati quando le particelle cariche (di solito da fasce di radiazione o dai raggi cosmici) perdono energia ionizzanti nel mezzo attraverso il quale passano, lasciando dietro di sè una scia di coppie elettrone-lacuna." I Single Event Upset (SEU) sono errori software transitori e non sono distruttivi per i circuiti.

Un SEU può accadere in componenti analogici, digitali e ottici oppure può avere effetti nel dintorno dei circuiti di interfaccia. I singoli eventi di upset appaiono tipicamente come impulsi transitori in circuiti logici o di sostegno, oppure come bit flip nelle celle di memorie o dei registri. Anche è possibile avere un multiple SEU bit in cui un singolo ione colpisce due o più bit causano errori simultanei. I multiple SEU bit sono un problema per le strategie di rilevamento e correzione di singoli errori, dal fatto che risulta impossibile assegnare i bit al interno di una parola in diversi chips. Un SEU grave è il Single-Event Functional Interrupt (SEFI), è quello nel quale il SEU addivenne nel circuito di controllo del dispositivo portando il sistema in uno stato di collaudo, di stop o peggio ancora in uno stato sconosciuto. I SEFI alterano la normale operazione del sistema, e richiede un riavvio della potenza del sistema per ricuperarlo.

2.2.2 Singel Event Latchup

Questo è un evento che causa una condizione dove si perdono funzionalità del dispositivo a causa di un singolo evento indotto nello stato di funzionamento del dispositivo. Il primo in osservare un Single Event Latch-up (SEL) è stato Kolasinski nel 1979 durante il collaudo in terra dei dispositivi[5]. I SELs sono errori hardware, e sono potenzialmente distruttivi, vuole dire che possono generare guasti permanenti.

Il SEL risulta in una corrente di operazione alta, sopra le specifiche del dispositivo. La condizione di Latch può distruggere il dispositivo, trascinando verso il baso la tensione del bus, oppure generando danni irreversibili sul generatore di potenza.

Originalmente, la preoccupazione sulla presenza del Latchup è stata causata dagli ioni pesanti, tuttavia, il Latchup può essere generato dai protoni. Un SEL è risolto con un riavvio off-on della potenza d'alimentazione. Però se la potenza non è rimossa immediatamente, possono accadere fallimenti catastrofici nel dispositivo,

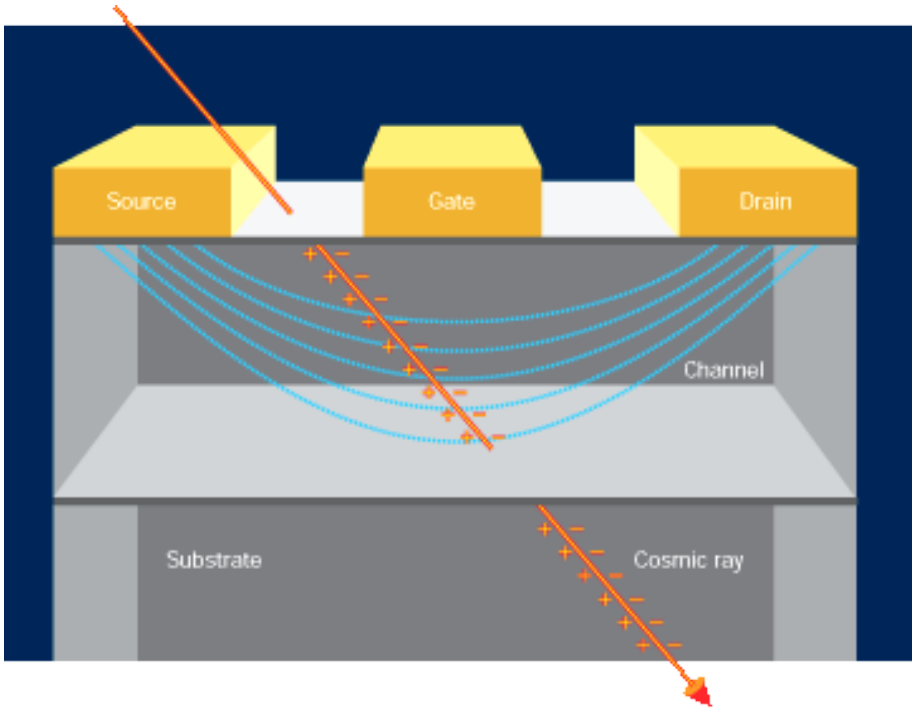


Figura 2.2: ***Ionizzazione a traverso un dispositivo microelettronico prodotto di un raggio cosmico***

dovuti al calore eccessivo o metallizzazione oppure fallimenti nei cavi di collegamento. Il SEL è molto dipendente della temperatura, dato che la soglia del avvenimento del Latchup diminuisce con il crescere della temperatura, allo stesso tempo che cresce la sezione trasversale.

2.2.3 Single Event Burnout

Il Single Event Burnout (SEB) è una condizione che può causare la distruzione del dispositivo, dovuta a uno stato prolungato di alta corrente in un transistore di potenza. Un SEB genera un fallimento permanete nel dispositivo. I SEBs includono i burnout di MOSFETs, rotture di porte logiche, bits congelati, e rumore nei Charge-Copuled Devices (CCD). Il primo SEB riportato è stato su i MOSFETs di potenza nel 1986 per Waskiewicz[11]. Soltanto si sono rilevati finora SEBs sui MOSFETs di potenza di canale *n*.

Un SEB può attivare un MOSFET di potenza disposto nello stato OFF, quando uno ione pesante attraversa un deposito e quello è abbastanza carico come per

accendere il dispositivo. Si è osservato che la suscettibilità ai SEBs diminuisce col crescere della Temperatura.

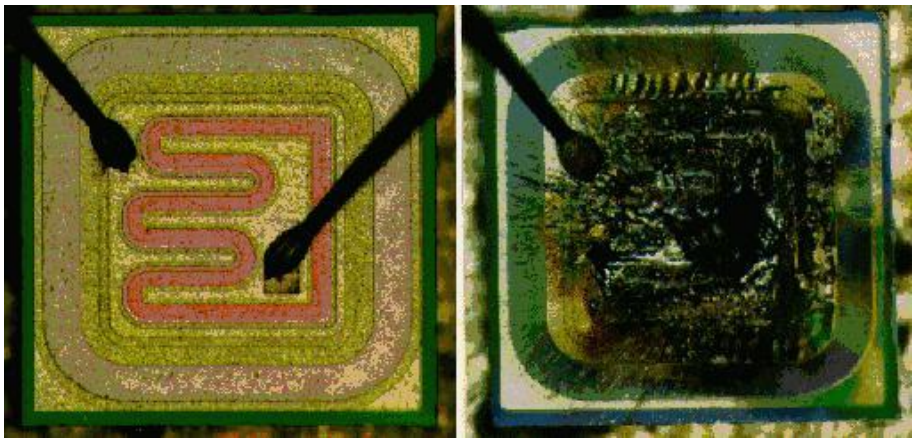


Figura 2.3: ***Circuito prima e dopo un SEB***

2.3 Modo di affrontare gli effetti delle radiazioni

Ci sono diversi approcci per affrontare e ridurre gli effetti dei SEE nei dispositivi, quei metodi sono scelti a secondo delle applicazioni (a livello di applicazioni spaziali dipende fortemente dell'orbita), del tipo di dispositivo, del rischio che rappresenta al sistema un errore in quel dispositivo, e a le condizione di progetto; come possono essere le condizioni economiche, di energia, di capacità di processamento oppure di spazio hardware.

Tra le tecniche software, si trovano diversi tipi, tutte che cercano di mitigare gli effetti dei SEUs, ci sono a livello commerciale molti alternative, tra quelle più utilizzate ce la perdizione degli errori, e successivamente la sua correzione; questi predittori per essere accurati e veloci implicano un compromesso software importante, dal fatto che consumano molte risorse per fare delle previsione, poi a seconda della complessità del algoritmo predittivo si può avere bisogno di un sistema che consuma molta potenza per riuscire a girare il software. per questo motivo ci sono anche soluzioni hardware per questo tipo di problemi, solitamente queste approcci vengono applicati sulle Field-Programmable Gate Array (FPGA), le quale permettono di avere una maggior versatilità che la implementazione diretta del circuito sul silicio; anche qua ci sono dei compromessi, tanto economici come di consumo di potenza e spazio.

Una combinazione tra tecniche software e hardware sarebbe la soluzione più consigliabile, per qualsiasi applicazione di alto rischio, comunque, sempre dipenderà dalle risorse a disposizione.

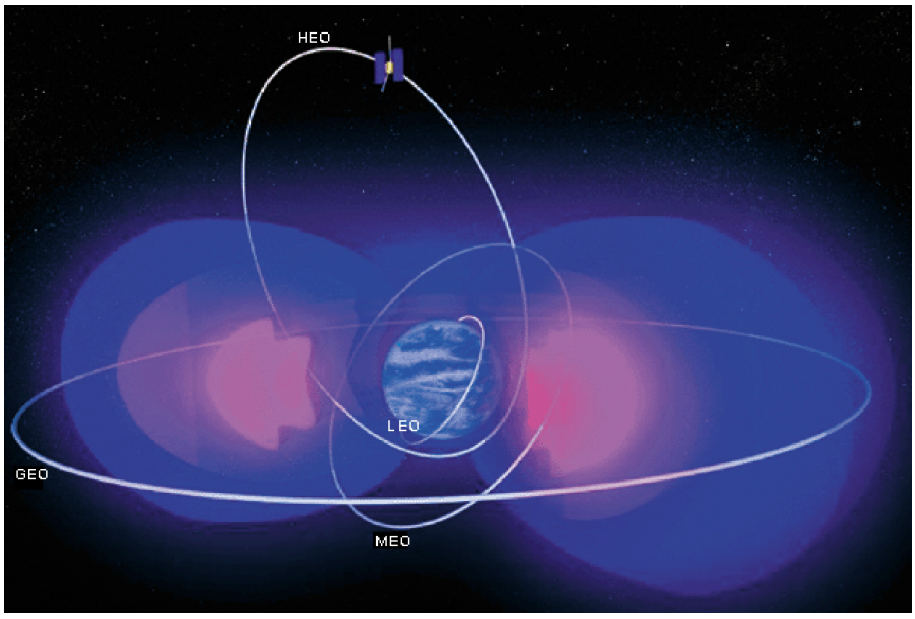


Figura 2.4: ***Cinture di radiazione di Van Allen e tipiche orbite dei satelliti,Illustration by B. Jones, P. Fuqua, J. Barrie, The Aerospace Corporation [12]***

Per l'orbita LEO, la più vicina alla terra (400-800Km di quota sopra il livello del mare) che si vede nella figura 2.4, abbiamo che i protoni possono essere i responsabili della maggior quantità dei guasti presenti provocando dei SEUs significativi, per questo motivo bisogna sviluppare delle tecniche adatte oppure utilizzare hardware e software certificati per applicazioni spaziali. Tra i componenti certificati abbiamo i Rad Hard e i Radietion Tolerance, come ultima alternativa resta utilizzare componenti Low Cost facendo delle opportune adattamenti e protezioni.

2.3.1 Componenti rad hard certificati

I dispositivi considerati Rad-Hard certificati, sono quelli che sottoposti a prove di radiazione compiono con le caratteristiche elencate nella tabella 2.1. Questo tipo di componenti sono solitamente collaudati dalle organizzazioni spaziali riconosciute come la NASA, l'ESA, oppure direttamente per la azienda che gli produce, e anche gli possono certificare altri istituzioni tanto pubbliche come private; Sempre che quelli componenti mantengano le specifiche richieste nelle normative e standard

pubblicate per gli organismi competenti [4, 7].

I dispositivi Rad-Hard sono costosi, dovuto alla alta qualità dei materiali con i quali vengono fatti, la accuratezza dei processi tecnologici coinvolti e di assemblaggio, ed a tutti i processi collaudo che devono eseguirsi per la sua certificazione. Per questo la sua scelta viene sempre condizionata dal fondo di progetto, è solitamente sono utilizzati unicamente nei componenti critici della missione, oppure nel caso di missioni militari o di viaggi inter-plnetari vengono utilizzati in tutto il complesso del progetto. Nel caso della missione AraMiS, data la sua filosofia, questi componenti possono essere usati unicamente quando le soluzione e tecniche di ingegneria no sono sufficienti per assicurate il perfetto funzionamento del satellite in orbita.

CARATTERISTICA	RAD HARD
Total Dose	10^5–10^6 rads
Dose-Rate Upset	$> 10^9$ rads (Si)/s
Dose-rate- Induced Latchup	$> 10^{12}$ rads (Si)/s
Neutrons	10^{14} – 10^{15} n/cm^2
Single Event Upset (SEU)	10^{-8} – 10^{-10} errors/bit-day
Single Event Latch-up (SEL)	37-80 MeV-cm^2/mg (LET)
Single Event Burnout (SEB)	

Tabella 2.1: Specificazioni Rad-Hard [7, 4, 9]

2.3.2 Componenti low cost e low power

Dispositivi commerciali di basso consumo di potenza e di basso costo, usati spesso nella attualità in quasi tutti i progetti elettronici a livello mondiale, sono stati introdotti nelle applicazioni spaziali negli ultimi anni, nel marco delle iniziative di ricerca universitarie che si sono aderite a lo standard Cubesat [13] questi componenti sono stati anche introdotti nelle applicazioni spaziali; ambito nel quale erano sempre stati di uso molto ridotto, non che nullo.

La problematica di questi tipo di componenti in applicazioni spaziali, è che hanno una forte variazione delle sue caratteristiche da un lotto di produzione ad un altro ed invero nello stesso lotto. Per questi prodotti è richiesto un maggior margine di progettazione e un collaudo molto più accurato da parte dei progettisti; però presentano il vantaggio che hanno costi decisamente inferiori ai componenti Rad-Hard, che come abbiamo detto in precedenza sono prodotti per avere prestazioni superiori.

Tra questi tipo di componenti si trovano i Commercial Off-The-Shelf (COTS). Un prodotto COTS può essere adoperato in alternativa a componenti sviluppati internamente all'azienda. Nell'ambito di progetti di sviluppo hardware e software, questa pratica è spesso una strategia volta a contenere i costi di sviluppo e manutenzione. Nel caso di componenti hardware, spesso anche i costi di produzione unitari del prodotto finale sono ridotti, dato che i componenti COTS sono ottimizzati e prodotti su scala più vasta rispetto a componenti equivalenti dedicati e sviluppati internamente. In realtà si tratta di funzionalità e/o applicativi pronti per l'uso. Questi tipi di dispositivi sono stati allo stesso modo scartati per molto tempo dalle applicazioni spaziali. Per non avere, come tutti i componenti commerciali, un capacita di ripetere accuratamente le sue caratteristiche, e per avere margini di funzionamenti più bassi dei dispositivi che rispondono a gli standard militari.

Tutte queste particolarità possono essere superate con tecniche di ingegneria applicate nella fase di progettazione e sviluppo del prodotto finale; nel caso di AraMiS i COTS vengono usati spesso con le apposite tecniche di protezione e collaudo che una applicazione spaziale richiede, nella tabella 2.2 si elencano le caratteristiche nominali dei componenti COTS.

CARATTERISTICA	COTS
Total Dose	10^3–10^4 rads
Dose-Rate Upset	10^6–10^8 rads (Si)/s
Dose-rate- Induced Latchup	10^7–10^9 rads (Si)/s
Neutrons	10^{11}–10^{13} n/cm^2
Single Event Upset (SEU)	10^{-3}–10^{-7} errors/bit-day
Single Event Latch-up (SEL)	< 20 MeV-cm^2/mg (LET)
Single Event Burnout (SEB)	

Tabella 2.2: Specificazioni dei dispositivi COTS [7, 4, 9]

Capitolo 3

Progetto della Smart Watch Dog

L'Unified Model Language (UML), è un linguaggio universale per rappresentare qualunque tipo di sistema, sia esso software, hardware e anche organizzativo. Il suo obiettivo è specificare e documentare le caratteristiche di un sistema.
Modellare il sistema per mitigare gli effetti alle radiazione per un processore commerciale con l'aiuto degli strumenti forniti dall'UML consente di rendere la scrittura del codice più agevole ed efficiente oltre al fatto che è più semplice scrivere un codice riutilizzabile in futuro, prevedere ed anticipare eventuali carenze del sistema.

Nell'ambito di un progetto come AraMiS, nel quale professori, dottorandi e studenti si avvicendano nella stesura del software e anche si impegnano nella realizzazione dei diversi moduli hardware, l'utilizzo dei diagrammi UML permette di fornire una chiara idea, a chiunque sia coinvolto nello sviluppo, di tutto l'insieme che costituisce il sistema.

Per fare ciò l'UML si avvale di strumenti grafici, per lo più diagrammi, attraverso i quali esplicare le funzioni strutturali e comportamentali del sistema. I principali diagrammi sono:

- Diagramma dei casi d'uso: rappresentano i "modi" in cui il sistema può essere utilizzato;

- Diagramma delle classi: specifica le classi e gli oggetti che fanno parte del progetto e le loro interazioni;

- Diagramma di sequenze: mostra le interazioni che avvengono tra gli oggetti che partecipano a una situazione specifica, mettendo in primo piano le relazioni tra gli oggetti e la loro topologia. I diagrammi di sequenze sono specialmente adatti a mostrare un particolare flusso o situazione di programma e sono

uno dei migliori tipi di diagramma per dimostrare o spiegare rapidamente un processo nella logica del programma.

3.1 Diagrammi di Casi d'Uso

Spesso il primo passo per modellizzare un sistema è definirne i casi d'uso. Questi diagrammi rappresentano le funzionalità che il sistema mette a disposizione dei suoi utilizzatori.

L'utilizzatore del sistema è chiamato attore. I diagrammi descrivono l'interazione tra attore e sistema senza prendere in esame la struttura interna di quest'ultimo *(modello black box)*. È importante comprendere che la figura di attore può essere ricoperta non solo da esseri umani, ma anche da altre applicazioni, sistemi o in casi più generale enti oppure organizzazioni.

Nella figura 3.1 si può vedere i primi casi d'uso per la Smart Wacth Dog nella quale possiamo identificare un attore la *"CPU"*, che richiede al sistema di fare un controllo dello stato della memoria (di un blocco specifico) tramite il metodo "Check". Poi in figura 3.2 possiamo individuare un altro caso d'uso *"WatchdogTimer"* (WDT), il quale può scatenare il processo di *"Reset* oppure di *"Riprogrammazione"* dell'intera memoria operativa della CPU o di un blocco specifico della stessa.

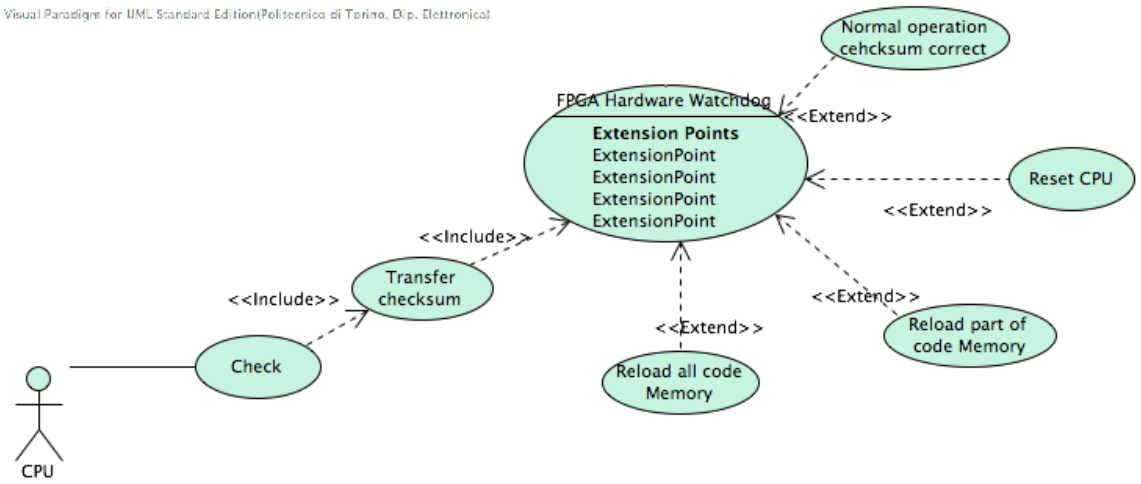


Figura 3.1: ***Caso d'uso della Smart Wacth Dog, per l'attore 'CPU'***

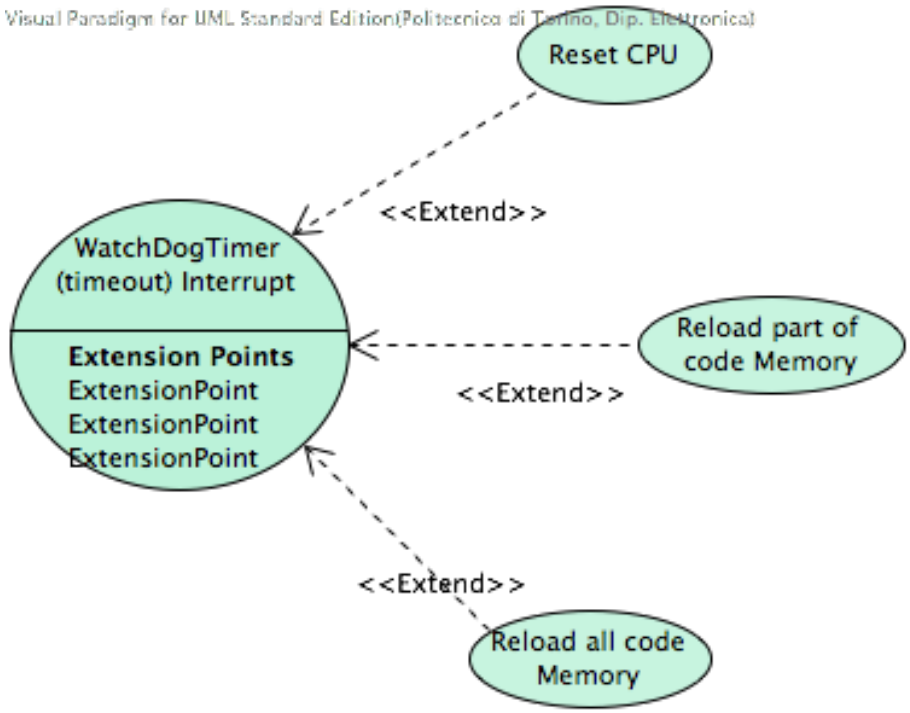


Figura 3.2: ***Caso d'uso della Smart Wacth Dog , per l'attore 'WatchdogTimer'***

3.1.1 Corsi d'azione base

Al fine di rendere più chiara l'interpretazione dei diagrammi dei casi d'uso ognuno di essi è accompagnato dalla descrizione del proprio *corso d'azione base*. In questi testi attore e sistema dialogano ed il corso d'azione base termina quando sono state fornite tutte le risposte necessarie a soddisfare l'obiettivo dell'attore. Il sistema, cioè la Smart Wacth Dog, viene denominato per brevità Smart Watch Dog (SWD).

Check:

CPU Chiede di fare il controllo di un blocco di memoria;

SWD Riceve l'ordine, e calcola il "Checksum" (parola clave di 16 bit), dal contenuto del blocco di memoria segnalato dal contatore che all'interno della fase software;

SWD Trasmette via porta seriale il "Checksum" alla fase hardware della SWD, in modo che sia questa fase l'incaricata di fare il confronto tra il checksum previamente calcolato e quello che viene generato nella FPGA;

SWD In base al risultato dal confronto tra i "Checksum" delle entrambi fasi la SWD, scatena uno dei casi d'uso descritti in seguito.

Normal Operation:

SWD In questo caso, il checksum della fase software è arrivato in tempo ed è corretto, si azzera il timer della Smart Wacth Dog, si resta in attesa di un nuovo evento.

Reset CPU:

SWD In questo caso, il checksum della fase software è arrivato in tempo, ma è incorretto e negli ultimi 10 secondi non si sono prodotti più di due errori (stato del 'rebootCounter'). Si da l'ordine di fare il reset della CPU.

Reload Part of code Memory:

SWD In questo caso, il checksum della fase software è arrivato in tempo, ma è incorretto e negli ultimi 10 secondi si sono prodotti più di due errori (stato del 'rebootCounter'). Si ordina inviare la Password di acceso alla unità di controllo di programmazione del μ-processore. Se la password viene accetta correttamente si da l'ordine di fare la riprogrammazione del ultimo blocco di memoria operativa controllato della CPU.

Reload All code Memory:

SWD In questo caso, il checksum della fase software è arrivato in tempo, ma è incorretto e negli ultimi 10 secondi si sono prodotti più di due errori (stato del 'rebootCounter'). Si ordina inviare la Password di acceso alla unità di controllo di programmazione del μ-processore. Se la password non viene accetta correttamente si da l'ordine di fare la riprogrammazione di tutta la memoria operativa della CPU.

WatchdogTimer interrupt:

SWD Quando arriva al massimo del tempo permesso dal progettista, senza ricevere informazione del μ-processore, invia la richiesta al sistema di incrementare il "reboot counter" il quale ha registro di quante volte si è realizzato un "reset" o una riprogrammazione della memoria in un periodo di tempo stipulato (10 secondi per questo lavoro di tesi).

SWD In base allo stato del "reboot counter",decide di esegue uno dei casi d'uso elencati precedentemente, "Reset CPU", "Reload Part of code Memory", "Reload All code Memory" con il particolare che non viene più presso in considerazione lo stato del checksum, dovuto al fatto che questo caso d'uso viene messo in funzionamento precisamente per la mancanza di un checksum in arrivo da parte della CPU. Fa il "Reset" del WDT.

3.2 Diagrammi di Classi

Il Diagramma delle Classi fornisce una chiara rappresentazione degli oggetti che compongono il sistema e le interazioni fra di essi. Per la natura del progetto, sarebbe più corretto parlare di Diagramma degli Oggetti, in quanto di ogni classe è presente una sola istanza, ovvero l'oggetto corrispondente (salvo in contate eccezioni).

In questo progetto abbiamo due diagrammi delle classi, il primo che descrive le classi che appartengono alla fase software della SWD (figura 3.3), implementata all'interno delle routine della CPU; ed un secondo diagramma che raduna le classi che appartengono alla fase hardware, ovvero quella implementata su FPGA(figura 3.4).

3.2.1 SW_SmartWatchdog

Questa classe è la principale della fase software, ed è incaricata di interfacciare la CPU con il resto della Smart Wacth Dog. I codici '.cpp' e '.h' di questa classe le possiamo trovare in 1 e 2 dell'appendice A. La classe "SW_SmartWatchDog" contiene un unico metodo:

`check():bool` Quando è chiamato, richiede al "ChecksumGeneraitor" (CG), tramite il suo metodo `createChecksum():ushort`, di creare il checksum apposito per il blocco di memoria che indica il "Counter", salva il valore ritornato dal metodo di CG nel attributo privato `checksum`. Questo ultimo en inviato tramite la "Serial_Interface" alla fase hardware della Smart Wacth Dog.

3.2.2 ChecksumGenerator

Questa classe si incarica di generare il "checksum" corrispondente al blocco di memoria che indica il "Counter", a come unico metodo `createChecksum():ushort`. I codici di questa classe gli troviamo in 3 e 4 dell'appendice A

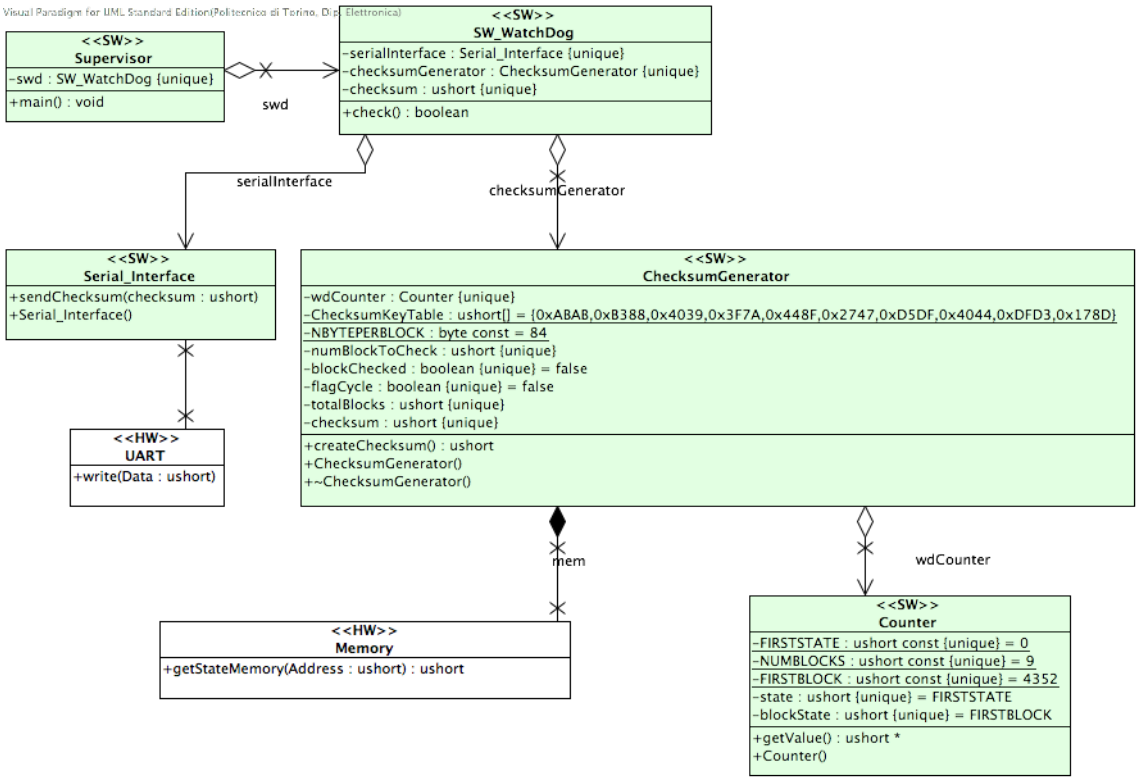


Figura 3.3: ***Diagramma delle classe del Software della Smart Wacth Dog***

`createChecksum():ushort` Al essere chiamato, legge dal posto di memoria indicato da `wdCounter.getValue():blockState` fino ad arrivare a leggere `NBYTEPERBLOCK` celle di memoria, ogni cella di memoria rappresenta un byte, e sono lette da due byte in due byte(**Parola**). Questo metodo per generare il `checksum` realizza la operazione *xor* tra tutte le parole lette dalla memoria, e alla fine della lettura si realizza un'ultima operazione *xor* con il valore che corrisponde al blocco di memoria del `checksumKeyTable:ushort[]`, la quale è una Look-Up Table (LUT) che contiene le parole chiavi necessarie per uguagliare il checksum generato dai dati contenuti dei blocchi di memoria con quello di controllo generato dalla Smart Wacth Dog nella fase hardware.

La scelta di avere un LUT che permetta questo aggiustamento è dovuta a la necessità di avere la libertà di modificare a piacere il "Firmware" complessivo della CPU quando la missione sia già nello spazio; e dal fatto che i checksum generati nella fase hardware sono fissi per ogni singolo blocco, bisogna avere la maniera di aggiustare i risultati dei checksum provenenti da ogni blocco di memoria della CPU, con i suoi corrispondenti nella fase hardware;

`ChecksumGenerator()` Costruttore che inizializza le variabili (attributi) della classe, quando si fa la istanza da parte de la classe che la dichiara;[1]

`~ChecksumGenerator()` Distruttore, il quale è chiamato nel momento che deve essere rimossa la istanza fatta della classe che ha il suo stesso nome. [2]

3.2.3 Counter

La classe "Counter", come il suo nome lo indica conta i blocchi di memoria che dovranno essere controllati. Contiene un unico metodo `getValue: ushort*` che ha come valore di ritorno un vettore che contiene il numero del blocco a controllare ed il indirizzo di memoria della prima cella di quel blocco. Questa classe contiene due attributi costanti `NUMBLOCKS` e `FIRSTBLOCK`, i quali sono molto importanti, già che loro indicano il numero di blocchi e l'indirizzo della prima cella del primo blocco di memoria, a controllare rispettivamente. I codici di questa classe gli troviamo in 5 e 6 dell'appendice A

`getValue():ushort*` Questo metodo, d'implementazione molto semplice, ogni volta che è chiamato ritorna il valore corrente dello stato del contatore "`state`, insieme al indirizzo della prima cella di memoria del corrispondente stato, allo stesso tempo che incrementa lo stato del contatore, per prepararsi alla prossima chiamate.

3.2.4 Serial_Interface

Questa ha al suo interno la configurazione specifica della interfaccia seriale del μ-Processore scelto ed i comandi necessari per inviare tramite porta seriale il dato di 16 bit che riceve tramite il suo unico metodo `sendChecksum(checksum:ushort):void`. I codici di questa classe gli troviamo in 7 e 8 dell'appendice A

[1] Tutti i costruttori hanno lo stesso comportamento, si identificano per avere lo stesso nome della classe alla quale appartengono, da questo momento en poi sarano elencati unicamente quando ci sia bisogno di approfondire nelle inizializzazioni che devono fare.

[2] Tutti i distruttori si comportano dello stesso modo, si indentificano per avere lo stesso nome della classe alla quale appartengono ed a che sono preceduti dal simbolo "~"; da questo momento en poi non saranno più elencati nelle descrizioni delle successive classi.

`sendChecksum(checksum:ushort):void` Questo metodo riceve il dato che gl'invia il `SW_SmartWatchDog` e lo colloca nel buffer di uscita del μ-Processore, nel caso dello sviluppo di questa tesi il processore scelto è il MSP430FG439, per il quale nel suo datasheet[15] e nel suo User's Guide[16], ci sono tutti i requisiti di configurazione, per un funzionamento ottimo della sua interfaccia seriale.

`Serial_Interface()` In questo costruttore si trovano tutti i comandi di configurazioni necessari per il funzionamento della interfaccia seriale del MSP430FG439 con il protocollo UART, con 8 bit di informazione, un bit di start, un bit di stop y un bit di parità; a una velocità di 115200 baud per trasmissione e ricezione.

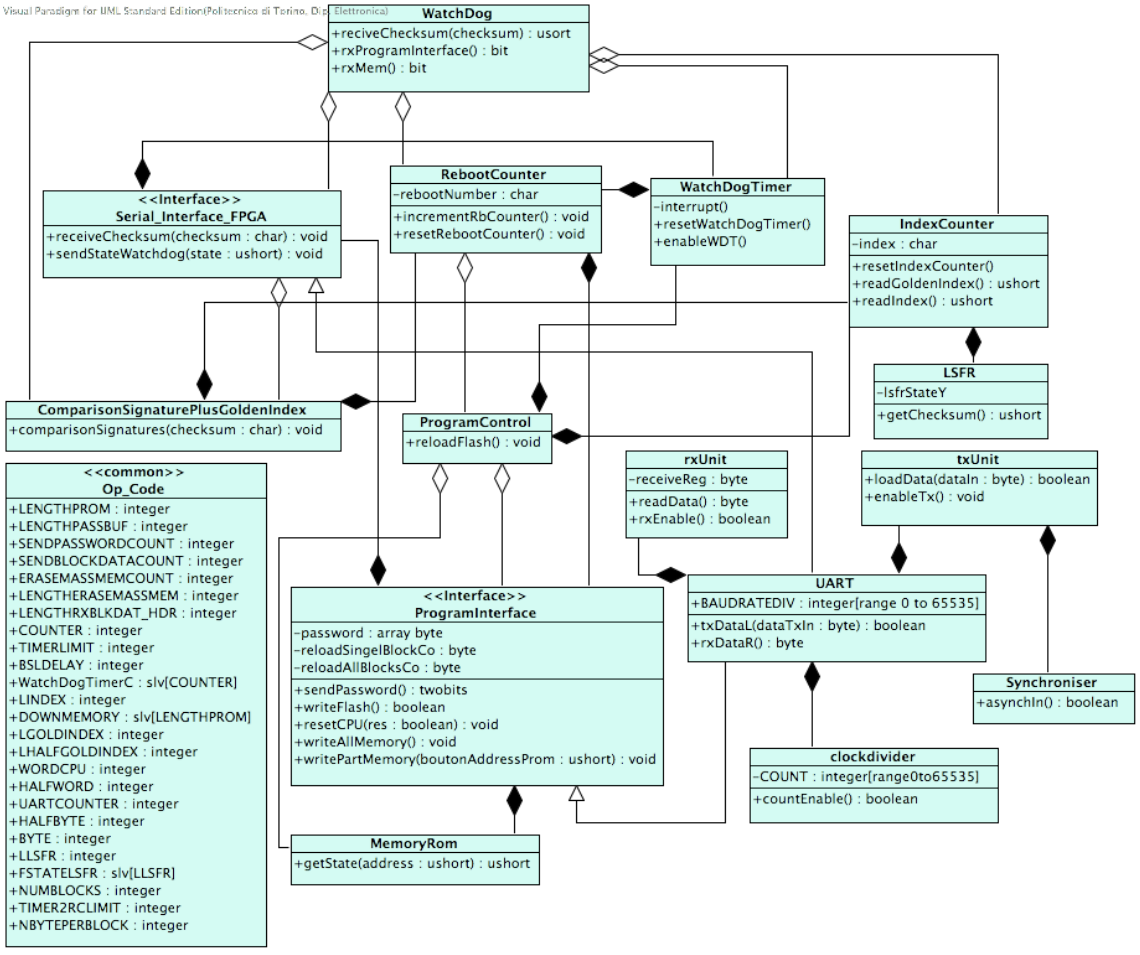


Figura 3.4: ***Diagramma delle classe del Hardware della Smart Wacth Dog***

3.2.5 Watchdog

Questa classe —Chiamata soltanto "Watchdog" anzi che HW_SmartWatchdog unicamente per semplicità— contiene tutte le classe, ovvero oggetti che servono a implementare la Smart Wacth Dog nella FPGA. Il codice VHDL di questa classe lo troviamo in 11 dell'appendice A

`reciveChecksum(checksum:ushort):void` Questo metodo trasmette (collega) la chiamata ricevuta al metodo dello stesso nome della classe "Serial_Interface_FPGA";

`rxProgramInterface():void` Questo metodo trasmette (collega) la chiamata ricevuta al metodo dello stesso nome della classe "ProgramInterface";

`rxMem()` Questo metodo trasmette (collega) la chiamata ricevuta al metodo dello stesso nome della classe "MemoryRom".

3.2.6 Op_Code

Questa classe contiene tutte le costanti che servono a tutte gli altre classe della fase hardware. Tutte sono attributi pubblici. Il codice VHDL di questa classe lo troviamo in 10 dell'appendice A

Lunghezza in bit/byte:

`LENGTHPROM` : per il indirizzo della memoria non-volatile, valore costante (VC)= 16, tipo di dato (TD)= integer;

`LENGTHPASSBUF` : del buffer della password, VC= 30 TD=integer;

`LENGTHERASEMASSMEM` : del buffer per la cancellazione della memoria, VC=10, TD=integer;

`LENGTHRXBLKDAT_HDR` : del `rxBlockData` 'header buffer', VC=8, TD=integer;

`COUNTER` : del `WDTCounter`, VC=8, TD=integer;

`LINDEX` : del Index Counter, VC=16, TD=integer;

`LGOLDINDEX` : del `goldenIndex` e del `checksum`, VC=16, TD=integer;

`LHALFGOLDINDEX` : di metà del `goldenIndex` e del `checksum`, VC=8, TD=integer;

`WORDCPU` : della parola della CPU, VC=16, TD=integer;

`HALFWORD` : di metà della parola della CPU, VC=8, TD=integer;

`UARTCOUNTER` : del contatore di la UART, VC=4, TD=integer;

`HALFBYTE` : di metà di byte, VC=4, TD=integer;

`TWOBIT` : di un vettore di due bit, VC=2, TD=integer;

`BYTE` : di un byte, VC=8, TD=integer;

`LLSFR` : dello stato del LSFR, VC=5, TD=intger;

`TOP64KB` : di 64 chilo byte, VC=65536, TD= integer;

`MEMSIZE` : della memoria "ROM", delle stesse dimensioni della memoria operativa della CPU, VC=TOP64KB-FIRSTADDMEM, TD=integer;

Numero di byte a inviare al Bootstrap Loader (BSL) quando si trasmette:

`SENDPASSWORDCOUNT` : la password, VC=31, TD=integer;

`SENDBLOCKDATACOUNT` : un blocco di data, VC=216, TD=integer;

`ERASEMASSMEMCOUNT` : il commando di cancellazione della memoria, VC=11,TD=integer;

Limite di conto per aver un tempo: ($f_{clk} = 20\,MHz$)

`TIMERLIMIT` : di 10ms nel timer 1 del `RebootCounter`, VC= 200000, TD=integer;

`BSLDELAY` : di 1,2ms nel timer del `ProgramInterface`, VC= 24000, TD=integer;

`WatchDogTimerC` : di To Be Define (TBD) ms nel `WatchdogTimer`, VC=x"TBD", TD=std_logic_vector;

`TIMER2RCLIMIT` : di 10 secondi nel timer 2 del `RebootCounter`, VC=200M, TD=integer;

Altre costanti:

`FSTATELSFR` : primo stato del LSFR, VC=x"16", TD=std_logic_vector (LLSFR -1 downto 0);

`NUMBLOCKS` : numero di blocchi a controllare della memoria operativa del μ-processore, VC= TBD, TD=integer;

`NBYTEPERBLOCK` : numero di byte per blocco a controllare, VC=250, TD=integer;

`FIRSTADDMEM` : indirizzo della prima locazione della memoria operativa del Processore= h1100 , VC=4352 , TD=integer.

3.2.7 WatchdogTimer

Questa classe come il suo nome lo indica è un "Timer" che al arrivare al suo valore massimo `WatchDogTimerC`, auto-chiama il suo metodo privato `interrupt`, genera un segnale che incrementa il contatore della classe "RebootCounter". Nella figura 3.5 si vede il top rtl della classe, generato tramite il codice VHDL presente in 12 dell'appendice A

`interrupt():bool` Quando il "timer" arriva a `WatchDogTimerC` attiva questo metodo privato, il quale indica tramite un segnale di uscita —Attivo in basso (0)— che il tempo predefinito, nel quale la CPU deve riuscire a girare tutto un blocco di memoria di dimensione `NBYTEPERBLOCK` in byte è scaduto, senza ricevere il checksum corrispondente da parte del processore; lo quale può indicare un guasto nel blocco di memoria operativa in esecuzione. Per tanto bisogna andare a fare un "Reset" oppure nel caso di ripetuti "interrupt" un ripristino di quel blocco di memoria, o di tutta nella sua complessità;

`resetWatchdogTimer()` Questo metodo a come scopo fare il reset del registro del "Timer" per impedirle di arrivare al suo valore massimo.

`enableWDT()` Questo metodo è un metodo 'On-Off', non ritorna nessun valore, ma permette di abilitare o disabilitare la WDT, vuole dire che, quando è attivo (Livello Alto) permette alla WDT di attendere il metodo `interrupt()` e di incrementare il contatore del timer, in caso contrario blocca il contatore e la chiamata della interruzione. Continua ad attendere il metodo `resetWatchdogTimer()`

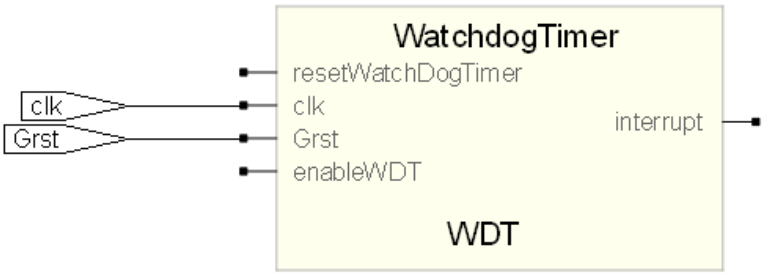


Figura 3.5: ***Top della classe Watchdog Timer***

3.2.8 Serial_Interface_FPGA

Questa classe implementa la interfaccia tra il μ-processore e la FPGA, fa uso di una istanza della classe "UART". Nella figura 3.6 si vede il top rtl della classe, generato tramite il codice VHDL presente in 13 dell'appendice A

`reciveChecksum(checksum:ushort):void` Questo metodo implementa il ciclo necessario per memorizzare la parola del checksum, la quale arriva divisa in due byte, il primo la metà bassa, poi la metà alta. Al ricevere l'intera parola, genera un segnale —Attivo in basso (0)— che indica che a ricevuto con successo il checksum, allo stesso tempo che pone a disposizione nel bus di uscita il valore ricevuto;

`sendStateWatchdog(state:ushort):void` In vecce questo permette di riportare alla CPU il numero di volte, in tutta la vita di funzionamento della Smart Wacth Dog, che è stata fatta un riprogrammazione della memoria, se trasmettono due valori di stato ("state") il primo —la parte bassa della intera parola— è il conto di quante volte si è ripristinato un singolo blocco di memoria (qualsiasi), il secondo —parte alta— notifica le volte che si ha fatto una riprogrammazione completa della memoria operativa della CPU.

Figura 3.6: ***Top della classe Serial Interface***

3.2.9 ComparetionSignaturePlusGoldenIndex

È incaricato di comparare il checksum arrivato dalla CPU, con il goldenIndex della FPGA. Il goldenIndex si richiede alla classe `IndexCounter`, ed è un valore pseudo aleatorio di 16 bit. Tutto questo tramite il suo unico metodo. Nella figura **??** si vede il top rtl della classe, generato tramite il codice VHDL presente in 14 dell'appendice A

`comparisonSignatures(checksum:char):bool` Qui si implementa la comparazione tra il checksum (CPU) e il goldenIndex (SWD), se la comparazione avviene positiva, il metodo segnala al sistema che quel blocco di memoria è in ordine; e con questa azione finisce il processo di controllo del blocco di memoria indicato dal `IndexCounter`. Nel caso che la comparazione sia negativa, il metodo indica che c'è al meno un errore nella memoria operativa del μ-processore e lo segnala incrementando il `RebootCounter`, lo qual scatena gli ulteriori processi, i quali possono essere il "Reset" della CPU, la riprogrammazione del blocco di memoria appena controllato o nel caso peggiore la riprogrammazione di tutta la memoria del μ-controllore.

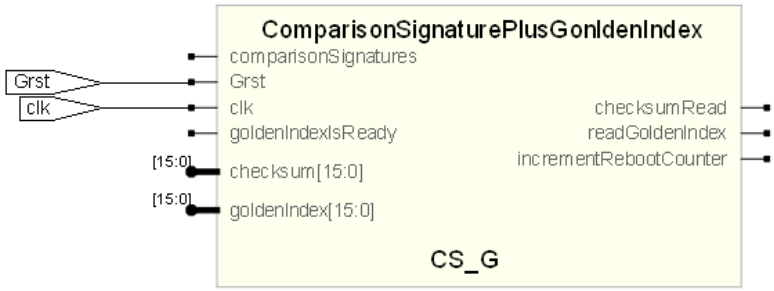


Figura 3.7: ***Top della classe Comparison Signature plus Golden Index***

3.2.10 RebootCounter

Questa classe ha conto delle volte che si è realizzato un "Reset" della CPU o un qualsiasi ripristino di memoria, entro 10 secondi[3], al scadere di questo tempo il contatore viene azzerato e si inizia a contare quei 10 secondi dalla prossima richiesta d'incremento. Il `RebootCounter` è munito da due metodi. Nella figura 3.8 si vede il top rtl della classe, generato tramite il codice VHDL presente in 17 dell'appendice A

`incrementRbCounter():void` La prima volta che è chiamato attiva il Timer —Al interno della classe— che alla scadenza del tempo prescelto azzera il contatore. Per ogni richiesta entro questo tempo di validità il metodo controlla il valore del contatore, nel caso che questo abbia un valore minore di due il metodo pone a disposizione i segnali necessari per fare il "Reset" della CPU; in caso

[3]Tempo imposto a livello di progettazione, può essere aggiustato a piacere da ogni progettista modificando nel Op_Code il valore della costante `TIMER2RCLIMIT`

contrario segnala al sistema che deve esistere al meno un errore persistente, e scatena il processo di ripristino della memoria;

`resetRebootCounter():void` Questo semplice metodo è l'incaricato di azzerare, in maniera asincrona, tutti i registri del `RebootCounter`, compresso il contatore.

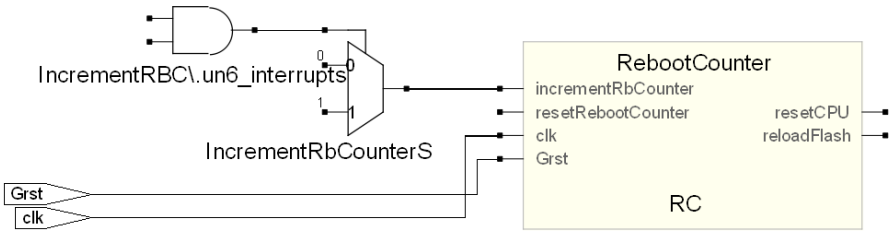


Figura 3.8: ***Top della classe Reboot Counter***

3.2.11 IndexCounter

Questa classe, contiene il contatore di blocchi, il quale indica il blocco e controllare in ogni ciclo di operazione, mete a disposizione tre metodi. Nella figura 3.9 si vede il top rtl della classe, generato tramite il codice VHDL presente in 15 dell'appendice A

`readIndex():ushort` Questo metodo ritorna il valore del contatore del Index, che indica il numero del blocco che si sta controlando;

`readGoldenIndex():ushort` Ritorna il valore del `goldenIndex` corrispondente al blocco di memoria che si sta controllando, al tempo che incrementa il contatore e calcola il `goldenIndex` per il prossimo ciclo;

`resetIndexCounter():void` Questo metodo azzera il contatore di blocchi e allo stesso tempo il generatore dei rispettivi `goldenIndex`.

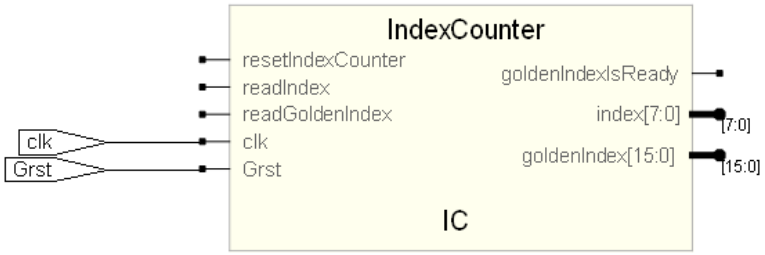


Figura 3.9: ***Top della classe Index Counter***

3.2.12 LSFR

Questa classe è un semplice generatore pseudo casuale di numeri a 5 bit, chiamato anche generatore numerico basato su registri di scorrimento; che genera 31 combinazioni. Conoscendo il valore iniziale del registro del LSFR si può determinare tutta la serie di combinazioni possibili e il ordine nel quale saranno generati. Il valore iniziale è fisso, per ottenere il massimo numero di combinazioni possibili (31), ed è "10000", il polinomio omogeneo e primitivo anche per permettere di avere il massimo numero di combinazioni ed è $f(x) = x^5 + x^4 + x^2 + x + 1$. Se vogliamo approfondire sulla teoria dei LSFR posiamo vedere [18]. Nella figura 3.10 si vede il top rtl della classe, generato tramite il codice VHDL presente in 16 dell'appendice A

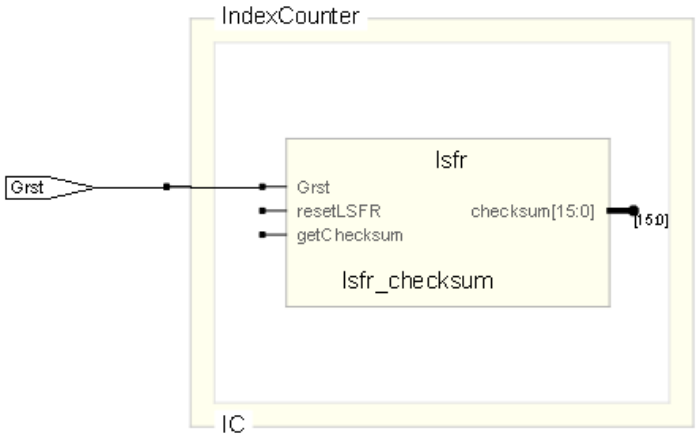


Figura 3.10: ***Top della classe LSFR***

`getChecksum():ushort` Ritorna come valore di checksum una parola di 16 bit, che contiene lo stato del LSFR ripetuto tre volte dal bit più significativo al meno

significativo, il bit zero ha come valore fisso '1', come si vede nella figura 3.11, al essere di 5 bit il polinomio generatore il numero di combinazioni rimane sempre a 31, però al far si che la parola di uscita sia di 16 bit si riesce ad avere 31 numeri con una codifica in bit molto lontana tra di loro.

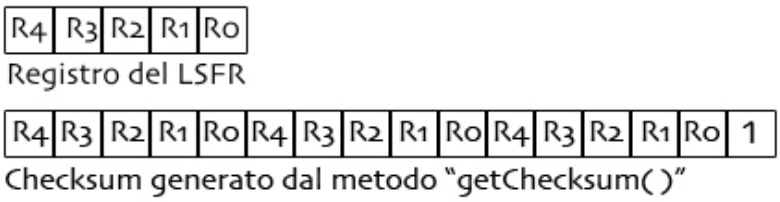


Figura 3.11: ***Modo di generare un goldenIndex (checksum) di 16 bit a partire di un LSFR di 5 bit***

3.2.13 ProgramControl

Questa classe riceve l'ordine di riprogrammare la memoria operativa, ed implementa il processo necessario per prendere la decisione su che si deve ripristinare se il singolo blocco che si sta controllando oppure tutta la memoria. Per questo si fa uso d'un unico metodo. Nella figura 3.12 si vede il top rtl della classe, generato tramite il codice VHDL presente in 18 dell'appendice A

`reloadFlash():void` Quando si chiama questo metodo, dalla classe si invia una richiesta al sistema di inviare la password al BSL del MSP430 e prendere il suo controllo, come risposta deve ricevere la conferma di che si è inviata la password, che si ha possesso del controllo della unità BSL e finalmente se la password fu accettata, in caso positivo il metodo da l'ordine al sistema di riprogrammare il singolo blocco che ha fallito, in caso negativo, vuol dire la password no fu riconosciuta, il metodo da l'ordine di ripristinare la intera memoria. Il metodo resta in attesa della conferma del successo della scrittura in memoria, in caso che questo segnale no arrivi genera una bandiera che indica che il μ-processore non risponde ed deve essere messo fuori di funzionamento.

3.2.14 ProgramInterface

Questa è la classe più specializzata di tutta la SWD, dal fatto che deve essere adatta a programmare il specifico processore a proteggere, se questa tecnica vuole

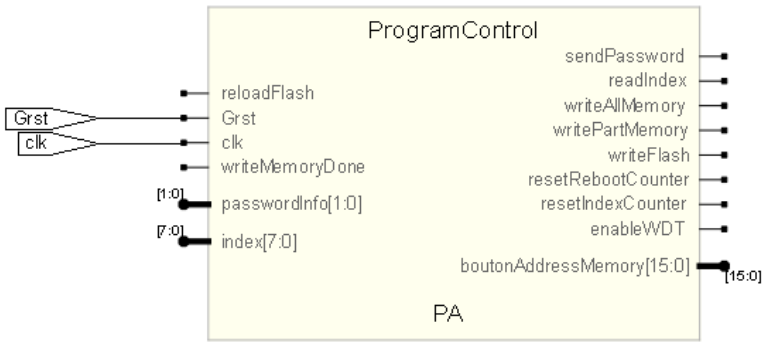


Figura 3.12: ***Top della classe ProgramControl***

essere utilizzata, per controllare il buon funzionamento di un diverso tipo di processore, i metodi di questa classe devono essere adattati al nuovo μ-controllore. Per implementare l'interfaccia tra il sistema SWD, la memoria non volatile (che contiene il firmware del processore) y la unita BSL del MSP430FG439, la classe mete a disposizione 5 metodi. Nella figura 3.13 si vede il top rtl della classe, generato tramite il codice VHDL presente in 19 dell'appendice A

`resetCPU():void[On-Off]` il primo metodo, del tipo On-Off deve essere mantenuto in On—Attivo in basso (0)— il tempo necessario per realizzare il "reset" di un Processore standard, nel quale genererà i segnali necessari per realizzare il reset dipendendo del processore a proteggere. Una volta messo in Off disattiva i segnali di uscita;

`sendPassword():twobits` Questo metodo si incarica di prendere il controllo della unità di programmazione e di successivamente inviare la password (PW) a quella unità, e tradurre il risultato al sistema con la codifica "01" se è corretta la (PW) e "10" se è incorretta;

`writeAllMemory():void` Questo metodo non può essere attivato in qualsiasi istante, il sistema deve prima dare l'ordine di prendere controllo della unità di programmazione. —tramite il metodo precedente— Questo metodo prepara tutti i comandi di programmazione e lettura della memoria non volatile per ripristinare la intera memoria;

`writePartMemory(boutonAddressProm:ushort):void` Al uguale che il metodo precedente questo può essere chiamato soltanto dopo aver richiesto il controllo dell'unità di programmazione tramite il `sendPassword()`. Implementa la preparazione dei comandi per programmare il blocco che si ha revisionato;

`writeFlash():bool` In vece questo può essere chiamato soltanto quando si ha chiamato uno dei due metodi su spiegati, prende i comandi pre-definiti e inizia il processo di programmazione.[4]. Ritorna un segnale indicando il successo della operazione di scrittura sulla memoria.

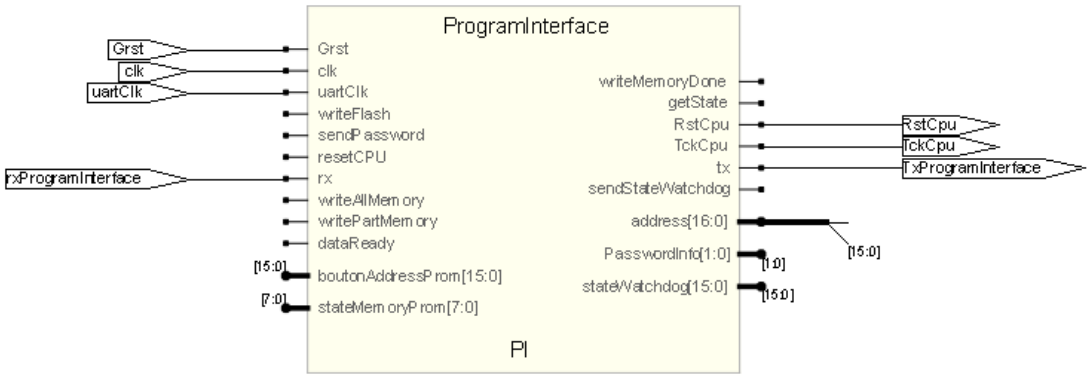


Figura 3.13: ***Top della classe ProgramInterface***

3.2.15 MemoryRom

Questa classe serve a implementare i protocolli di comunicazione con la memoria non volatile scelta per il progettista, per tanto il suo metodo sarà specializzato alla memoria. In qualsiasi caso al essere una interfaccia semplifica le richieste dei dati in memoria dal sistema. Nella figura 3.14 si vede il top rtl della classe, non si presenta il codice VHDL generale per questa classe, ma si presenta il codice utilizzato per le simulazioni e le prove in 24, dove questa classe si adattata come una memoria "Rom" all'interno della FPGA nell'appendice B.

[4]Nel caso specifico di questo lavoro di tesi, riferirsi ahi documenti [15, 16, 17] per una spiegazione approfondita del processo di programmazione tramite la unità BSL del MSP430FG439 ed i suoi requisiti.

`getState(address:ushort):byte` Implementa il protocollo di comunicazione necessario per interfacciare la memoria[5] con il sistema SWD, riceve l'indirizzo virtuale di memoria e lo trasforma in un indirizzo fisico per poi ritornare il valore che legge della memoria, insieme ad un segnale che indiche che il dato è pronto.

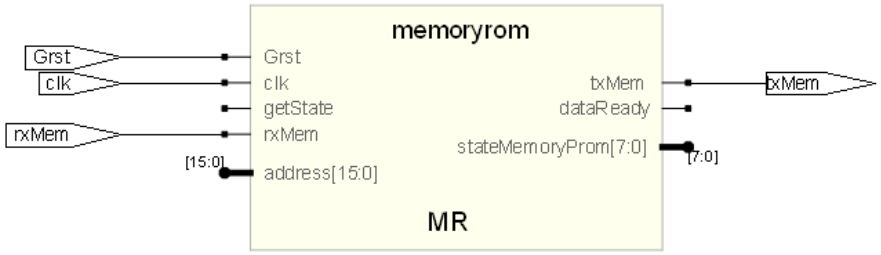


Figura 3.14: ***Top della classe MemoryRom***

3.2.16 UART

Questa classe è l'unica che si istanza più di una volta, ci saranno due, una che permette comunicare alla Smart Wacth Dog con il μ-controllore e l'altra che interfaccia il BSL del MSP430 con il `ProgamInterface` della SWD. Dal suo nome si puo inferire che implementa il protocollo di comunicazione seriale UART. Mete a disposizione due metodi. Nella figura 3.15 si vede il top rtl della classe, generato tramite il codice VHDL presente in 20 dell'appendice A

`txDataL(dataTxIn:byte):bool` Il dato che riceve "`dataTxIn`" lo pone a disposizione della `txUnit` e non attende nessuna altra chiamata fino che la unità di trasmissione indica il successo, dopo di che ritorna un valore bool per indicare che è pronto ad attendere nuovi chiamate;

`rxDataR():byte` Metodo che è chiamato dopo che la propria classe indica che c'è un dato nel bus di ingresso in attesa di essere letto, allo che ritorna il dato memorizzato nel suddetto bus.

[5]La memoria suggerita da questo lavoro di tesi è una "F-Ram", che ha delle caratteristiche positive nella resistenza alle radiazioni[19], che può trovarsi con bus seriale I^2C o IPS

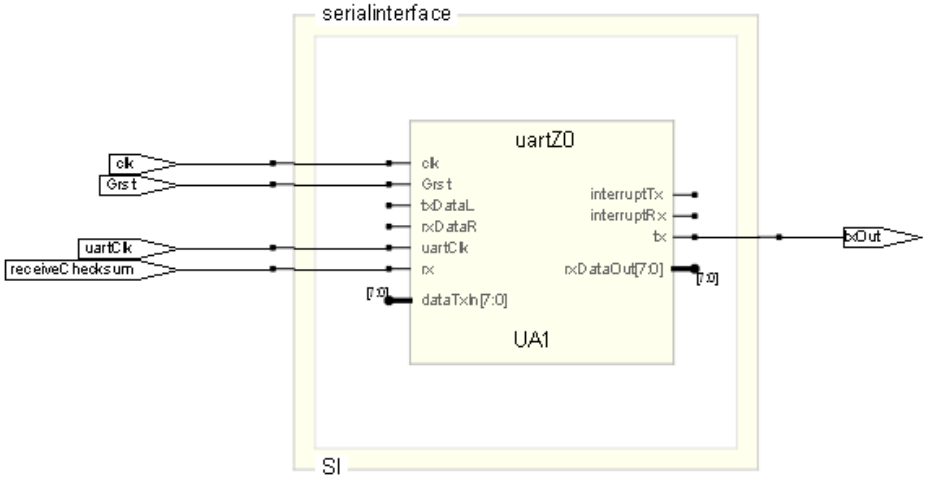


Figura 3.15: ***Top della classe UART, come istanza della classe Serial Interface***

3.2.17 txUnit

Questa classe tramite sui due metodi implementa il protocollo de trasmissione seriale asincrona di 8 bit di data, 1 bit di start, 1 bit di stop e un bit di parità. Nella figura 3.16 si vede il top rtl della classe, generato tramite il codice VHDL presente in 21 dell'appendice A

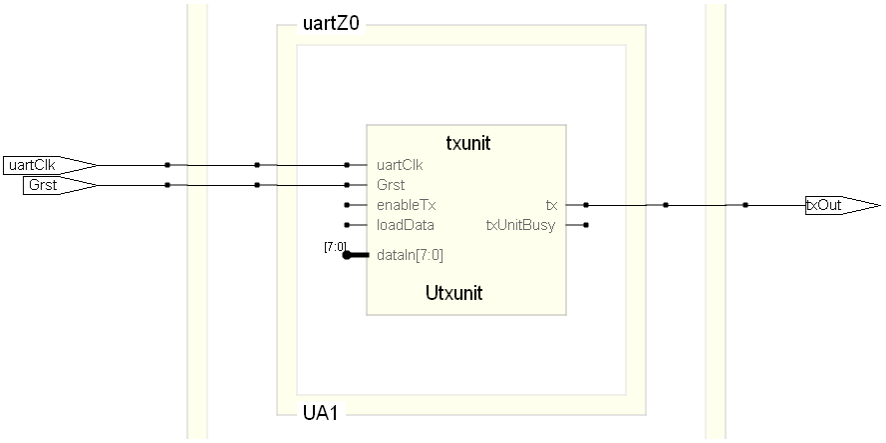


Figura 3.16: ***Top della classe txUnit***

`enableTx():void` Una volta chiamato attiva la unità per attendere richieste di invio di dati;

`loadData(dataIn:byte):bool` Questo metodo implementa il protocolo UART RS-232 per invio di dati, ritorna un segnale di un bit una volta inviato il dato.

3.2.18 rxUnit

Questa classe tramite sui due metodi implementa il protocollo de ricezione seriale asincrona di 8 bit di data, 1 bit di start, 1 bit di stop e un bit di parità. Nella figura 3.17 si vede il top rtl della classe, generato tramite il codice VHDL presente in 22 dell'appendice A

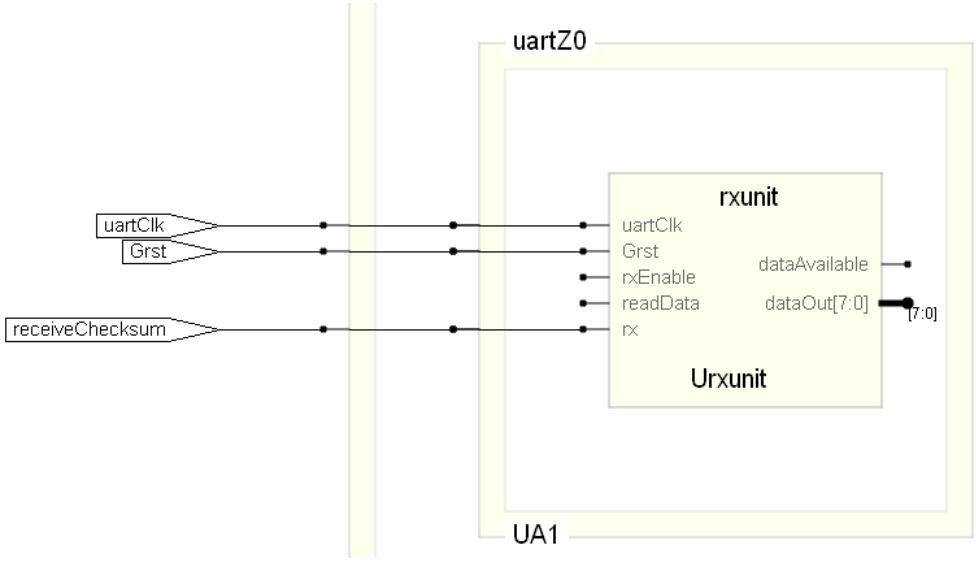


Figura 3.17: ***Top della classe rxUnit***

`rxEnable():bool` Una volta chiamato attiva la unità per attendere arrivi di dati al suo bus di ingresso, implementa il protocolo UART RS-232 per la ricezione di dati, ritorna un segnale di un bit una volta ricevuto un dato valido;

`readData():byte` Questo metodo ritorna il valore presente nel bus di ingresso.

3.2.19 ClockDivider

Come il nome l'indica questa classe divide (in tempo) il segnale di orologio al suo ingresso. Al dire dividere il "Clock" si intende allungare il periodo. Nella figura 3.18 si vede il top rtl della classe, generato tramite il codice VHDL presente in 23 dell'appendice A.

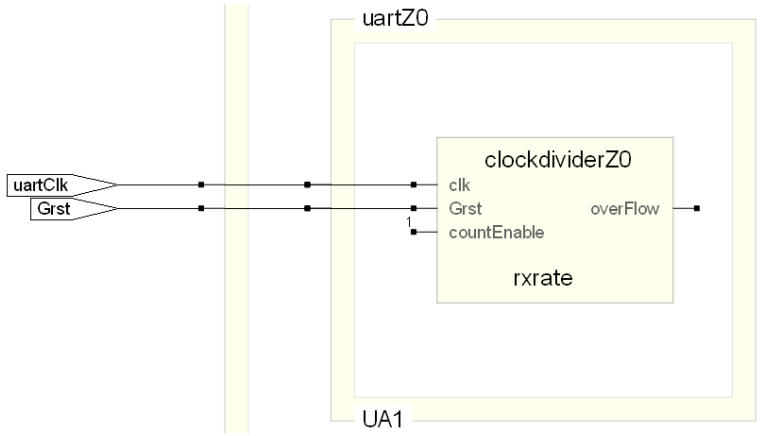


Figura 3.18: ***Top della classe ClockDivider***

`countEnable():bool` Metodo del tipo "On-Off", implementa la divisione del "Clk" d'ingresso e ritorna un "clock" di uscita di una frequenza uguale `COUNT` volte minore, quando è On. Se si mete in Off ritorna un zero costante.

3.2.20 Synchroniser

Un oggetto semplice che sincronizza un segnale aleatorio con un il fianco di salita di il "Clock" d'ingresso. Nella figura 3.19 si vede il top rtl della classe, generato tramite il codice VHDL presente in 23 dell'appendice A

`asynchIn():bool` Metodo che ritorna un bit attivo, quando arriva il fianco di salita del "Clock" d'ingresso successivo alla chiamata la quale e un segnale aleatorio.

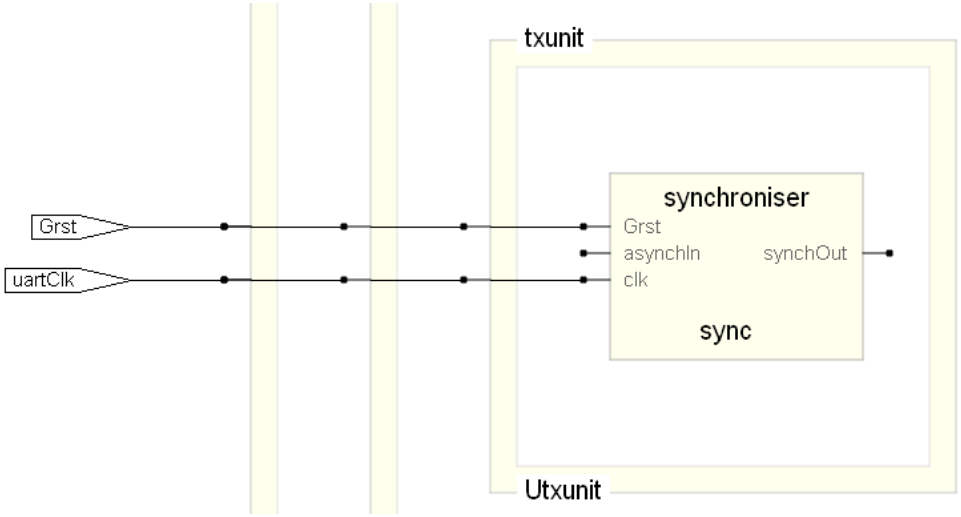


Figura 3.19: ***Top della classe Synchroniser***

3.3 Diagrammi di Sequenze

A continuazione si presentano le diverse possibilità di svolgimento dei flussi di programma e segnali, all'interno del sistema Smart Wacth Dog. Ogni una di queste possibilità attendono ai casi d'uso che può svolgere il sistema ed anche alle particolarità di ogni uno di loro.

3.3.1 Funzionamento normale

Il diagramma di funzionamento normale, svolge le sequenze di comunicazione tra la CPU e la FPGA, quando questa accade in tempo utile, vuole dire quando ancora non si è prodotta una interrupt da parte della `WatchdogTimer`. Nella figura 3.20 si osserva quando l'attività 'supervisor' chiama il metodo `check()`, il quale a sua volta chiama `createChecsum()`, per la creazione della 'checksum' (crc); poi nel diagramma vediamo un riassunto delle operazioni che esegue il metodo `createChecsum` per ricavare il crc, chiedendo lo stato al `Counter` e andando a leggere le celle di memoria che conformano il blocco a controllare, una volta calcolato il crc, lo ritorna all'attività `SW_WatchDog`. Consecutivamente si vede nella linea di tempo 7, la chiamata al metodo `sendChecksum()` per l'invio del crc attraverso del rs-232, per ogni chiamata del metodo `reciveChecksum()` si presenta un riassunto del processo del invio dello stato del buffer di serializzazione della `UART` del μ-processore (soltanto si elencano 4 bit), si fanno due chiamate di questo metodo perché si inviano due byte; dopo di che la `Serial_Interface_FPGA`, azzera la `WatchDogTimer`, e richiede il confronto del crc in arrivo e quello generato dalla FPGA, come si considera per

questo diagramma il caso normale, i checksum sono identici. Quindi la sequenza finisce con la comparazione.

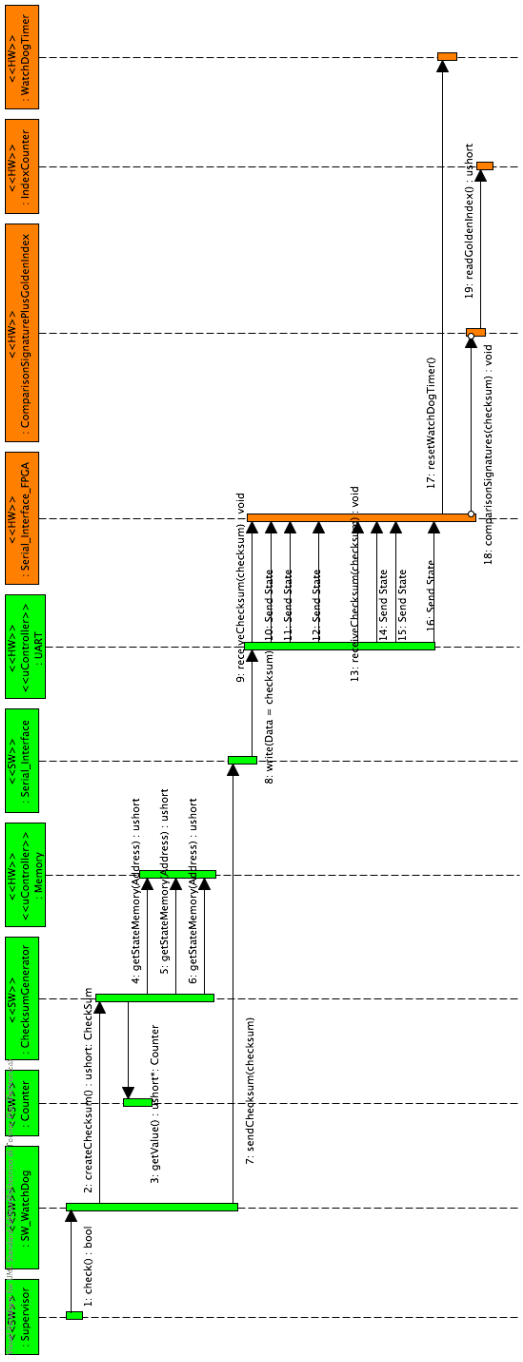


Figura 3.20: ***Diagramma delle sequenze del funzionamento normale del sistema***

3.3.2 Interrupt per timeout della WatchdogTimer

La sequenza della figura 3.21 si osserva la chiamata del metodo `interrupt()` della `WatchdogTi- mer` (wdt), questo accade unicamente quando non se produce un `resetWatchDogTimer` in un tempo utile; il tempo nel quale la wdt fa la chiamata della interruzione viene calcolato come nella equazione 3.1, il valore 'WatchDogTimerC' è una costante che viene aggiustata per avere un tempo di timeout bilanciato. Il metodo `interrupt()` genera la chiamata del metodo `incrementRbCounter()`. Le azione successive sono svolte nei diagrammi dei 3.3.4, 3.3.5 e 3.3.6 dal fatto che dipendono dallo stato del contatore del `RebootCounter`.

$$T_{(clk)} * WatchDogTimerC = timeout \tag{3.1}$$

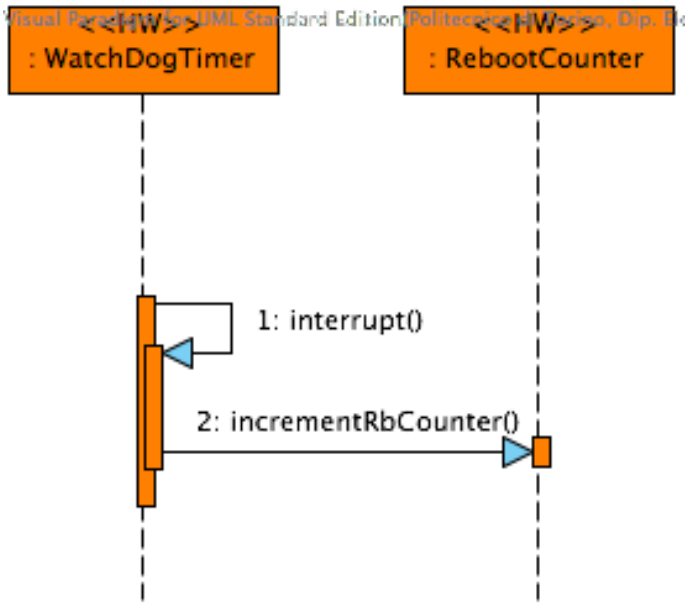


Figura 3.21: ***Sequenze del timeout della WatchdogTimer***

3.3.3 Checksum Incorretto

Nella figura 3.22 si evidenziano le sequenze che si producono quando un checksum arriva un tempo utile, ma presenta un errore ed è diverso da quello generato all'interno della fase hardware della Smart Wacth Dog, come nel caso del normale funzionamento, viene chiamato il metodo che fa la comparazione, come i checksum sono diversi si chiama il metodo `incrementRbCounter()`. Le azione successive sono svolte nei diagrammi dei 3.3.4, 3.3.5 e 3.3.6.

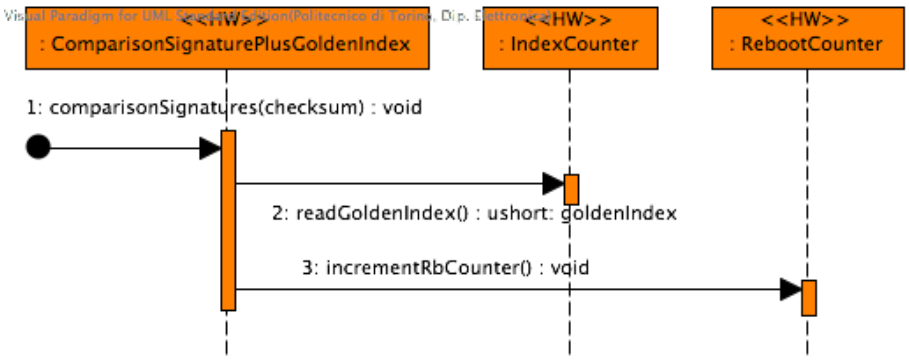


Figura 3.22: ***Sequenze per l'errore in un checksum***

3.3.4 Reset CPU

Nella figura 3.23 si osserva la sequenza della prima alternativa dopo una chiamata al metodo `incrementRebootCounter`, la quale si produce in un tempo tale che la `WatchdogTimer` non riesce a produrre la sua interruzione. Le sequenze che si svolgono sono le necessarie per fare il reset della CPU, si producono quando lo stato del rebootCounter '`rebootNumber`' è minore o uguale a due (2). Dopo che si attiva il metodo `resetCPU` si azzera il contatore del 'IndexCounter' tramite il metodo `resetIndexCounter`.

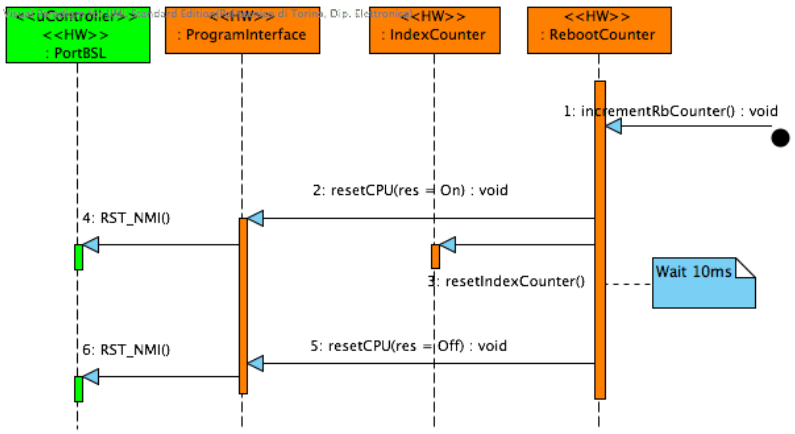


Figura 3.23: ***Sequenza per il reset della CPU***

3.3.5 Ripristino di un blocco di Memoria

Nel diagramma della figura 3.24 si osservano le sequenze che si producono per la riprogrammazione di un blocco della memoria operativa della CPU. Questa si

produce quando lo stato del `RebootCounter` e maggiore di 2, si da l'ordine al `ProgramControl` (PC) di fare il `reloadFlash()`; a sua volta il PC ferma il WDT tramite il metodo `enableWDT(off)`, dopo di che scatena l'invio della password attraverso la `ProgramInterface` (PI) che prende il controllo della CPU tramite l'unita BSL del MSP430, nella figura 3.25 si vede il protocollo che svolge la PI. Considerando che i processi di possesso del controllo della BSL e di invio della password siano fatti con successo, si decide di ripristinare unicamente il blocco di memoria che indica il `IndexCounter` come ritorno del metodo `readIndex()`, al interno della PC si calcola il primo indirizzo di memoria dal quale si deve iniziare la riprogrammazione, e lo invia come parametro alla PI quando chiama il metodo `writePartMemory()`; successivamente fa il reset di tutti i contatori e da l'ordine definitiva di scrivere la memoria tramite il metodo `writeFlash()`. Quando il processo di invio dei datti finisce la PI ritorna un valore 'bool' `writeMemoryDone` che indica la fine della scrittura, il PC riavvia il WDT inviando il 'on' tramite il suo metodo di 'enable'. Il protocollo dell'invio dei dati alla BSL dopo che si il controllo della suddetta è quello standard UART con un Baudrate di 9600, un bit di parità pare, en un bit de star e di stop; la sequenza di dati che devono essere inviati per ogni caso sono presenti nella figura 3.26; in cuesto caso vengono utilizati i comandi 'RX Password' e 'RX Data Block'.

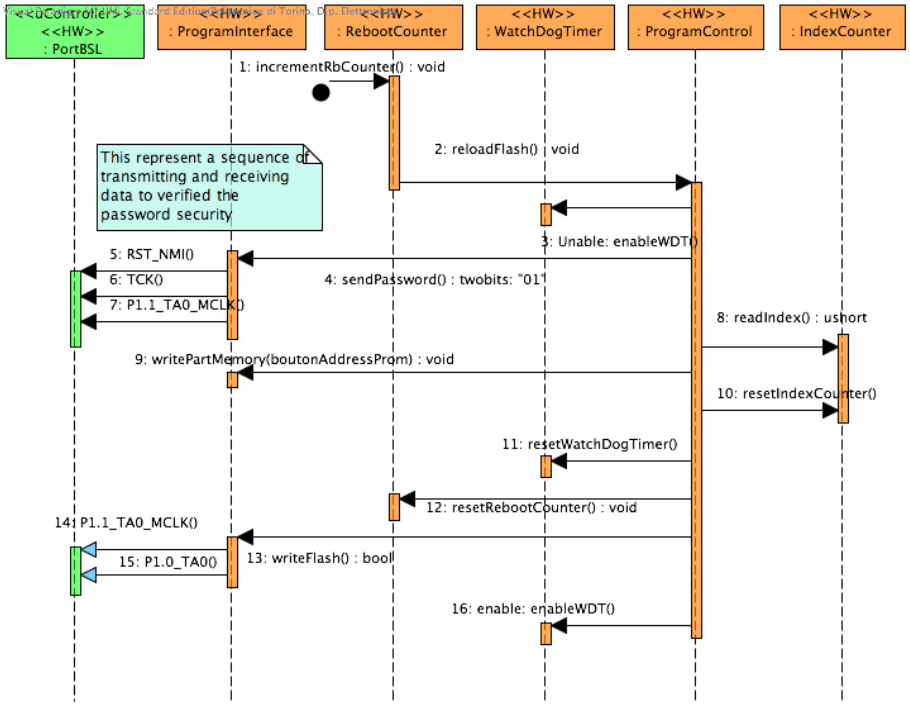


Figura 3.24: ***Diagramma di sequenze per il ripristino di un singolo blocco di memoria operativa***

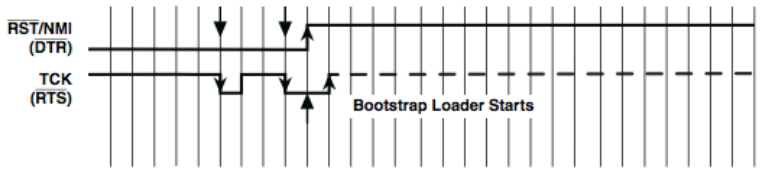


Figura 3.25: ***Protocollo di ingresso alla Bootstrap Loader (BSL)***

Received BSL Command	HDR	CMD	L1	L2	AL	AH	LL	LH	D1	D2...Dn	CKL	CKH	ACK
RX data block	80	12	n	n	AL	AH	n–4	0	D1	D2 ... Dn–4	CKL	CKH	ACK
RX password	80	10	24	24	xx	xx	xx	xx	D1	D2 ... D20	CKL	CKH	ACK
Erase segment	80	16	04	04	AL	AH	02	A5	–	– – –	CKL	CKH	ACK
Erase main or info	80	16	04	04	AL	AH	04	A5	–	– – –	CKL	CKH	ACK
Mass erase	80	18	04	04	xx	xx	06	A5	–	– – –	CKL	CKH	ACK
Erase check	80	1C	04	04	AL	AH	LL	LH	–	– – –	CKL	CKH	ACK
Change baud rate	80	20	04	04	D1	D2	D3	xx	–	– – –	CKL	CKH	ACK
Set mem offset	80	21	04	04	xx	xx	AL	AH	–	– – –	CKL	CKH	ACK
Load PC	80	1A	04	04	AL	AH	xx	xx	–	– – –	CKL	CKH	ACK
TX data block	80	14	04	04	AL	AH	n	0	–	– – –	CKL	CKH	–
BSL responds	80	xx	n	n	D1	D2 ...	...	...	...	... Dn	CKL	CKH	–
TX BSL version	80	1E	04	04	xx	xx	xx	xx	–	– – –	CKL	CKH	–
BSL responds	80	xx	10	10	D1	D2 ...	...	...	...	... D10	CKL	CKH	–

Figura 3.26: ***Data Frame della Bootstrap Loader (BSL)***

3.3.6 Ripristino di tutta la Memoria

Nel diagramma di sequenze della figura 3.27 si fa vedere le sequenze che producono il ripristino di tutta la memoria operativa, fino alla azione “8” (esclusa) le sequenze sono le estesse che in 3.3.5, in questo caso si considera che la password non è stata accettata, con lo che se presume che tutta la memoria o gran parte di questa è stata modificata dovuto alle radiazioni. Questa affermazione si realizza basati sulla certezza che la zona di memoria dove è presente la password del BSL, non può essere modificata dalla CPU in nessun modo, quindi questa situazione erronea si considera un segale che la memoria operativa è in uno stato sfavorevole per il buon funzionamento del μ-processore; l'unica sequenza diversa con 3.3.5 è la chiamata al metodo `writeAllMemory()` anzi che il metodo `writePartMemory`. I comandi che vengono utilizzati del elenco della figura 3.26 sono ‘Mass erase’, ‘RX Password’ e ‘RX Data Block’; dove l'ultimo comando è inviato tante volte come blocchi di memoria abbia la CPU.

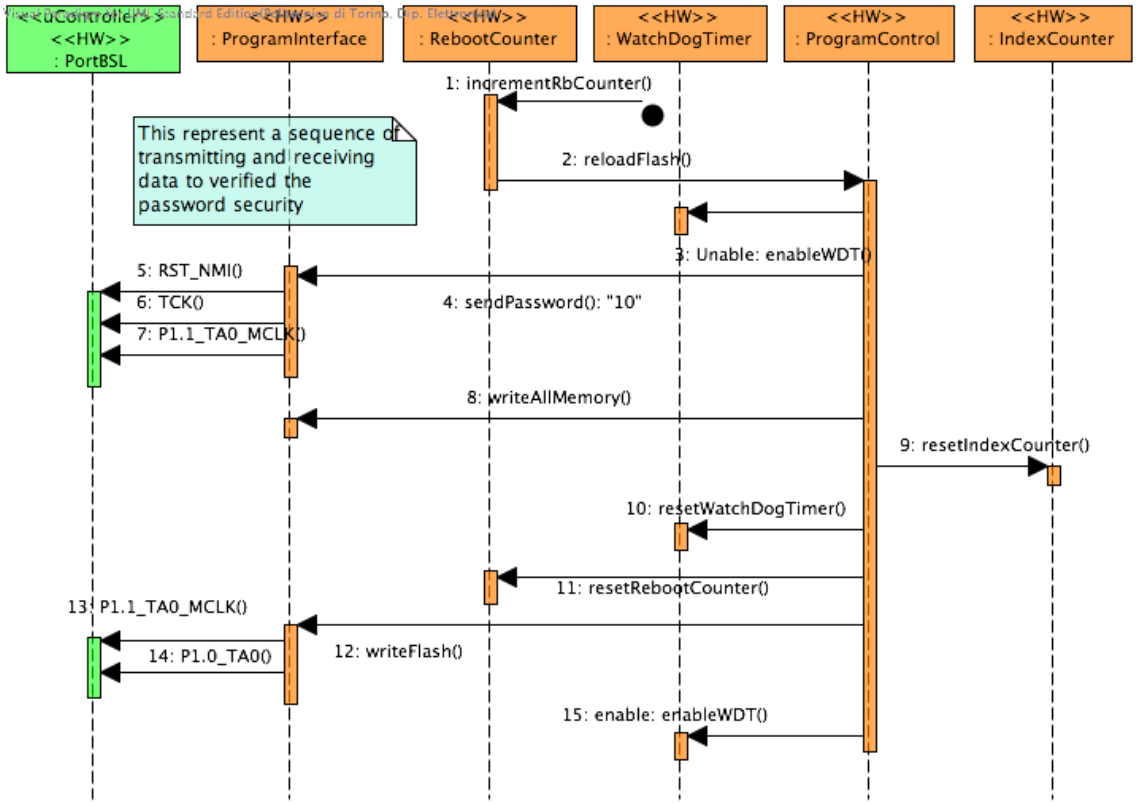


Figura 3.27: ***Diagramma di sequenze per il ripristino dell'intera memoria operativa***

Capitolo 4

Risultati ed Analisi

4.1 Software Smart Watchdog

Come primi risultati abbiamo quegli delle simulazioni e diversi collaudi fatti sul software, prima classe per classe e poi di tutto il complesso. Facendo uso del Identifier Extension (IDE) *IAR Embedded Workbench* si è provato tramite simulazione oggetto per oggetto il sistema `SmartWatchDog_CPU`, per comprovare il suo funzionamento. I risultati sono stati ottimi e d'accordo con le previsioni progettuali, perché esercitano la loro funzione senza registrare errori. Nell'appendice B in 25 si può osservare il codice del main usato per le prove, dove l'errore introdotto per le prove finali fu cambiare diverse istruzioni che in questo compaiono in modo che la Smart Wacth Dog gl'individuasse come cambiamenti dei dati della memoria.

Si è riuscito ad ottimizzare la struttura del codice che è stato scritto nella fase de progetto, riducendo il numero di operazioni assembler generate dal compilatore provenienti dal codice C++. Comprimendo lo spazio occupato dal software della Smart Wacth Dog all'interno della memoria operativa della CPU, fino a 589 Byte che equivalgono a 0,575 kB.

Dopo aver comprovato il singolo funzionamento per ogni classe, si sono messe assieme in unico sistema, e tramite simulazione si è collaudato, che:

1. Il software allocasse tutte le variabili e costanti in maniera corretta;
2. la lettura degli celle di memoria si facessero dal primo indirizzo della memoria flash en poi, senza andare a controllare zone di memoria che possono cambiare grazie al corretto funzionamento della CPU (e. zona della RAM);
3. la creazione del `checksum` si producesse unicamente con i datti contenuti nella memoria operativa;

4. la struttura logica e le istanze delle classe fossero corrette ed uniche, vuol dire che ogni classe fosse istanzata una volta sola in tutta la vita di funzionamento della CPU, non facendo differenza quali o quanti programmi girassero al interno del μ-processore;

5. per ogni blocco collaudato di memoria, anche effettuando molteplici controlli i checksum fossero rilevati sempre giusti.

Dopo aver fatto il processo di collaudo tramite simulazione, si è continuato con il 'Debug' sul processore di prova (MSP430FG439) faccendo uso del develoment tool MSP-FET430U80 della *'Texas Instrument'* che si può vedere nella figura 4.1, questo tool insieme al software *IAR Embedded Workbench* permettono di corroborare il buon funzionamento del software svilupato per la Smart Wacth Dog, permettendo di realizzare un collaudo approfondito in tempo reale del comportamento del codice nel processore fisico.

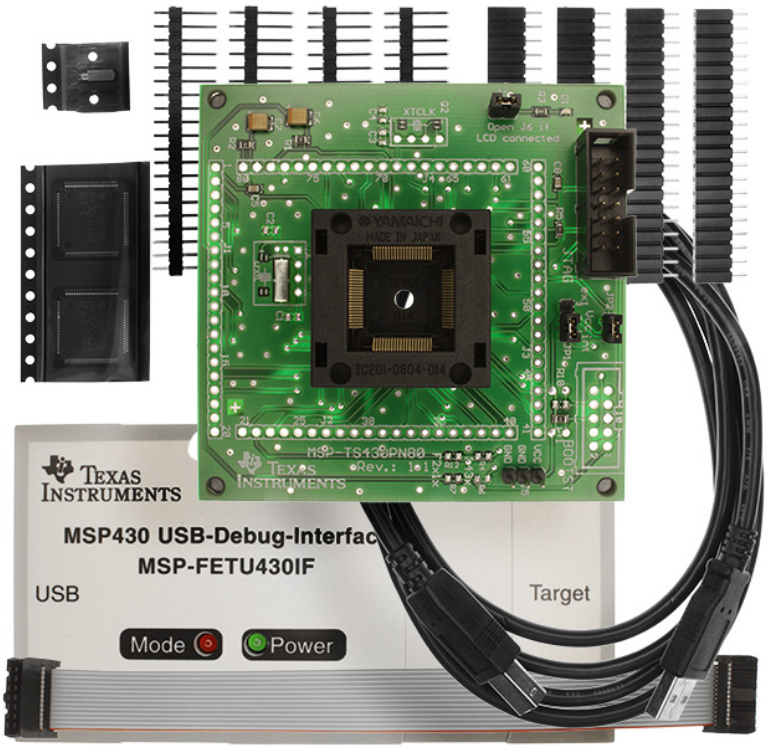


Figura 4.1: ***Develoment tool MSP-FET430U80***

Il collaudo realizzato sul tool MSP-FET430U80, ha permesso di comprovare che:

1. Il software alloca tutte le variabili e costanti in maniera corretta;
2. la lettura degli celle di memoria è fatta dal primo indirizzo della memoria flash in poi, senza andare a controllare zone di memoria che possono cambiare grazie al corretto funzionamento della CPU;
3. il `checksum` si genera unicamente con i datti contenuti nella memoria operativa;
4. la struttura logica e le istanze delle classi sono corrette ed uniche per ogni oggetto, non fa differenza quali o quanti programmi girano all'interno del MSP430FG439;
5. anche se si effettuano molteplici controlli di memoria i checksum sono rilevati sempre giusti e unici per ogni blocco.

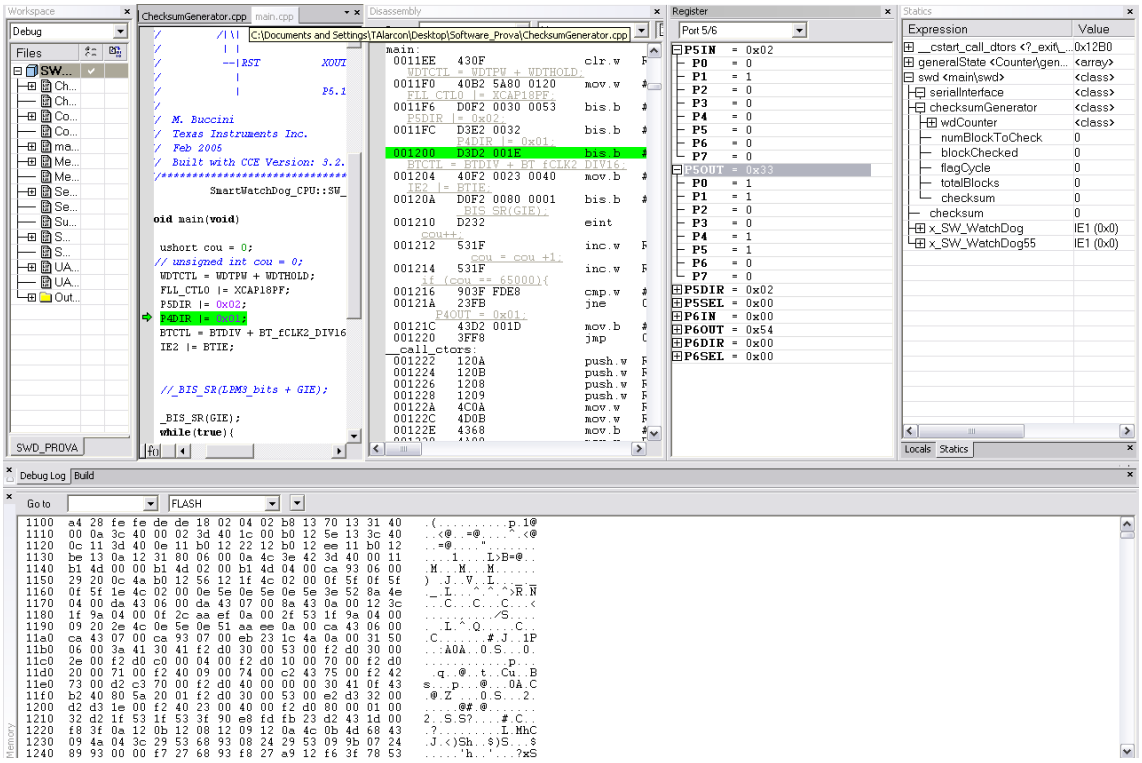


Figura 4.2: ***IAR Emdedded Workbench IDE, vista dei processi al interno del μ-processore durante il Debug***

4.2 Hardware Smart Watchdog

Per la fase hardware il processo di simulazione fa uso dei software SynaptiCAD e ModelSim, i quali permettono di generare i file di stimolo o 'testbench' e di simulare rispettivamente. I processi di simulazione del hardware si sono effettuati in diverse fasi, nello stesso modo che per il software, prima per ogni classe e dopo per tutto l'insieme degli oggetti; il ModelSim ci permette di osservare di maniera molto precisa i possibili comportamenti di tutti i segali all'interno di ogni componente sviluppato, consentendo di controllare l'assenza o presenza di errori o di stati non contemplati nella fase di progetto.

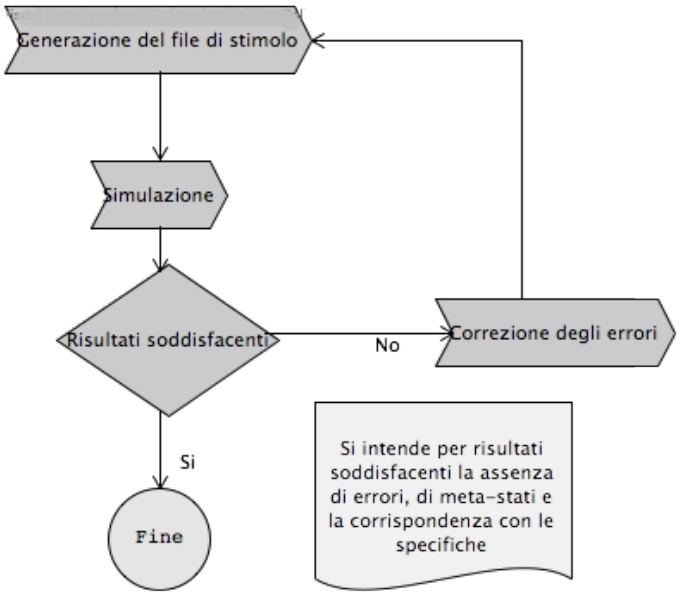


Figura 4.3: ***Ciclo per le simulazione di tutte le classi e del sistema***

Tramite le simulazioni si è verificato che:

1. Gli oggetti avessero tutti i possibili stati definiti e stabili;
2. tutte i segnali fossero inizializzati con il suo corretto valore;
3. se segue la logica definita a livello progettuale per ogni classe e per il sistema nel suo insieme;
4. i protocolli di comunicazione e programmazione (specifici al μ-processore usato in questa tesi) concordassero con le specifiche[17, 16] del componente a controllare;

5. al essere messi insieme tutti gli oggetti della Smart Wacth Dog di questa fase la stabilita degli stati e la accuratezza dei risultati fossero mantenute.

Il processo di simulazione è consistito da un ciclo il quale viene rappresentato dal diagramma della figura 4.3, questo processo è stato applicato a ogni singola classe e poi al sistema, permettendo di ottenere un codice robusto, ottimo ed efficiente a livello logico. Le simulazioni danno la possibilità di provare gli scenari di funzionamento in tutta la sua estensione, anche su quelle che non sono previste dal funzionamento normale del circuito, concedendo la opportunità di prevenire stati indesiderati.

Nelle figure 4.4 e 4.5 si osserva il risultato della simulazione di una richiesta di 'reset' da parte del `RebootCounter` al `ProgramInterface`, in questa simulazione si realizzano molteplici incrementi del RebootCounter questo è un caso improbabile ma si è fatto con lo scopo di vedere simultaneamente il reset per il caso nel quale `rebootNumber` è minore di 3, ed il caso nel quale questo stato è anche maggiore di 3, per il quale si produce la chiamata del `reloadFlash()` —Si considera uno stato improbabile dal fatto che mentre si fa il reset della CPU nn si poso ricevere più checksum e la WDT è stata azzerata un instante prima—, nella figura 4.6 si vede invece il protocollo d'ingresso del BSL e l'inizio dell'invio della password da parte del PI per il caso nel quale si richiede il ripristino della memoria operativa.

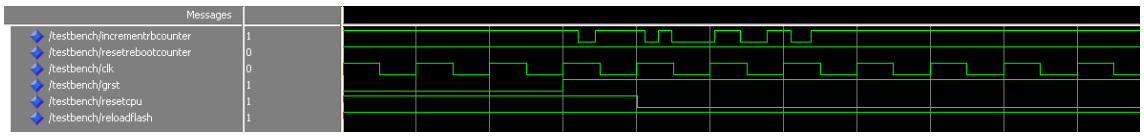


Figura 4.4: ***Diagramma di tempo di una simulazione del RebootControl***

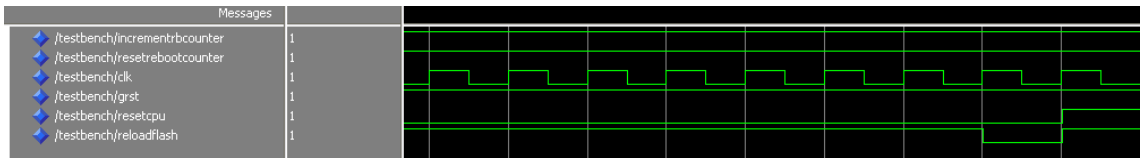


Figura 4.5: ***Diagramma di tempo di una simulazione del RebootControl***

Il passo consecutivo è stato la compilazione del codice tramite il software Symplify Pro Actel Edition, il quale ha permesso di comprovare l'efficienza del progetto ed allo stesso tempo di ottimizzare i componenti a livello fisico. Questo è un processo diverso a quello delle simulazione perché la ottimizzazione del codice in questo caso non è riferita alla logica che deve sviluppare il circuito. In questo caso quello

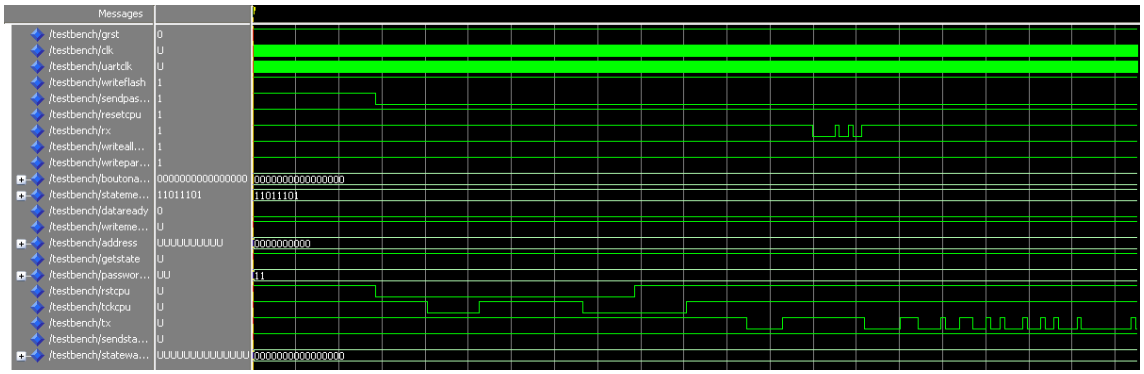


Figura 4.6: ***Diagramma di tempo di una simulazione della ProgramInterface***

che viene ottimizzato sono le caratteristiche proprie del circuito come velocità di risposta, consumo di potenza e spazio occupato all'interno della FPGA. Così come i componenti che il compilatore riesce a generare a partire della descrizione data tramite il codice VHDL.

La Smart Wacth Dog pronta può essere martirizzata in una qualsiasi FPGA, nello specifico di questa tesi in una qualsiasi FPGA della Actel di più di 60 chilo porte, vuol dire che può essere martirizzata in una FPGA della serie nano della Actel, che hanno dimensioni di anche '*10mmx10mm* '. Tramite il software 'Designer' si è generato il Layout finale di un prototipo di Smart Wacth Dog, il quale posiamo osservare nella figura 4.7 dove si vede che soltanto il 80% del nucleo della FPGA viene utilizzato, lo che può permettere applicare ulteriori tecniche di progettazione (e. TMR).

Nelle figure successive osserviamo i risultati per le previsioni dei consumi di potenza negli scenari più rilevanti del possibile funzionamento della Smart Wacth Dog nella fase hardware, questi risultati sono molto soddisfacenti dal fatto che il massimo consumo di potenza che si può avere in stato 'attivo' è di soltanto 23,04 mW, ed in stato 'sleep' è di 0,104 mW. Nella tabella della figura 4.8 si elencano i modi di funzionamento previsti negli analisi di potenza.

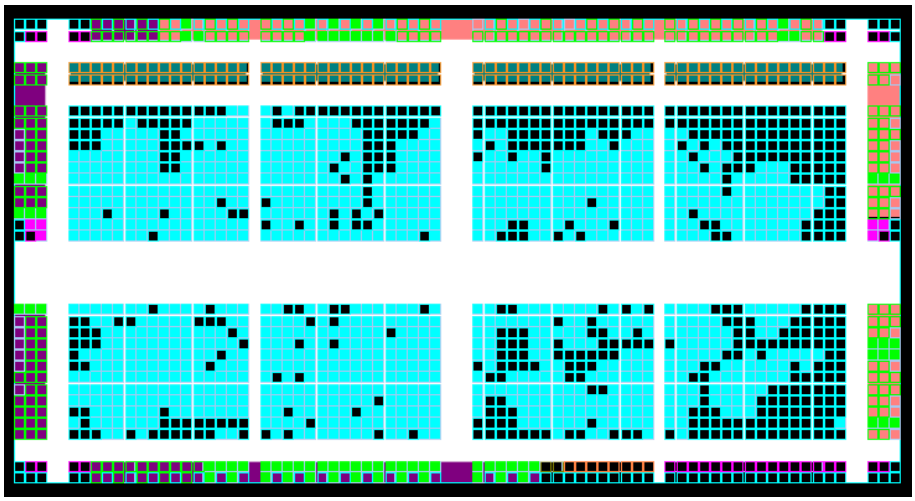


Figura 4.7: ***Layout di prototipo di Smart Wacth Dog su una FPGA Actel AP3N060 100VQFP***

Mode	On	Off
Active	All Vccs & All Clocks	None
Static (Idle)	All Vccs	All Clocks
Sleep	Vcci	Vcca (Core Voltage)

Figura 4.8: ***Tabella dei modi di funzionamento predefiniti per le previsioni di consumo di potenza***

Insieme ai risultati di potenza ed spazio ci sono quelli di velocità ($f_{requenze}[MHz]$), nel processo di compilazione si e configurato il compilatore per ottenere il circuito più veloce possibile, come risultato abbiamo ottenuto le frequenze della figura 4.13 delle quale possiamo estrarre la frequenza massima per il 'clk' e per il 'uartClk', che sono 120,8 e 108,8 rispettivamente. Queste sarebbero le frequenze che ci producono i massimi di potenza assorbita nello stato attivo del sistema, in vece di utilizzare queste frequenze di orologio andremo a utilizzare un 15% delle stesse, col quale si spera di avere un consumo di potenza di massimo **"9,178 mW"**.

In oltre si sono realizzate simulazioni e prove sul circuito come prodotto finale, vuole dire, che si è comprovato il funzionamento del circuito definitivo grazie al tool di sviluppo 'ARM Cortex-M1-Enable ProASIC3L', questi processi di simulazione e prove si differenziano dei precedenti, nel fatto che sono utilizzati come componenti quelli che vengono fuori delle compilazioni e sintetizzazione realizzate. Questo è un processo molto importante dovuto a che permette di comprovare la compatibilità del dispositivo con la tecnologia di implementazione della FPGA, oltre che permettere

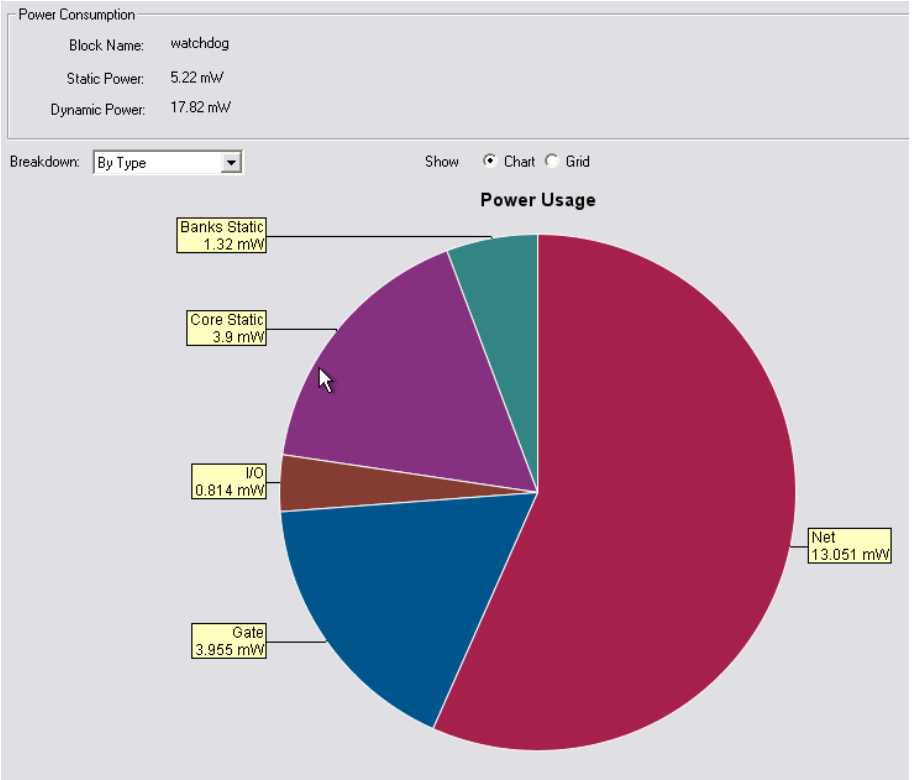


Figura 4.9: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo di ogni area della FPGA in stato attivo***

di realizzare un ulteriore collaudo del funzionamento del sistema.

Il circuito realizzato per le prove finali del sistema completo si presenta nella figura 4.16, lo schematico di collegamento si presenta nella figura 4.17, nello schematico si indicano in piedini utilizzati di ogni dispositivo dopo del nome della segnale che si sta collegando (es. rxProgramInterface_(piedino nella scheda di sviluppo)_(piedino della fpga)). Nell'appendice E si possono osservare i "pinout" dei due dispositivi (MSP430FG439 e ARM Cortex-M1-Enable ProASIC3L).

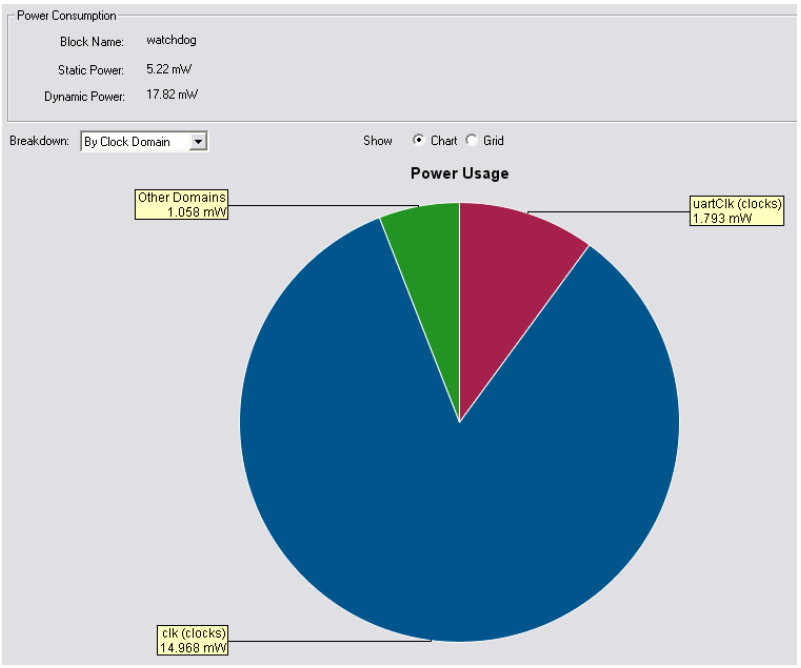


Figura 4.10: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo dovuto ai diversi orologi in stato attivo***

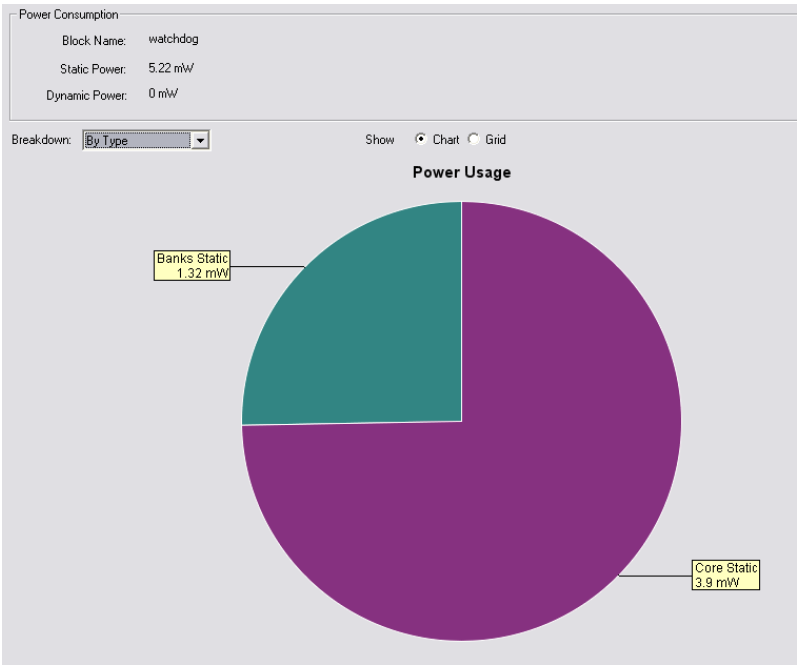


Figura 4.11: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo di ogni area della FPGA in stato 'static'***

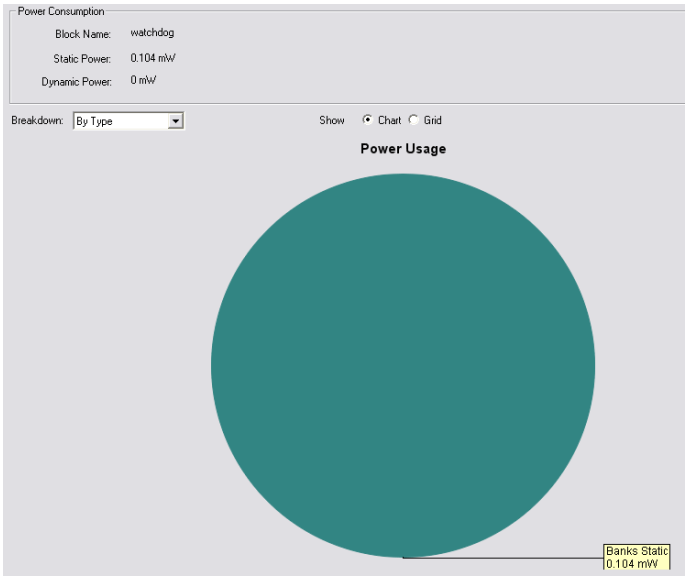


Figura 4.12: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo di ogni area della FPGA in stato 'sleep'***

	Name	Period (ns)	Frequency (MHz)
	CS_G/readGoldenInde x-0	6.643	150.534
ⓘ	PI/UA1/loadData:Q	0.694	1440.922
ⓘ	RC/reloadFlash:Q	0.744	1344.086
ⓘ	SI/UA1/Urxunit/dataA vailable:Q	0.696	1436.782
ⓘ	SI/UA1/Urxunit/receiv eLoad:Q	0.728	1373.626
ⓘ	SI/UA1/loadData:Q	0.778	1285.347
	clk	8.277	120.817
	uartClk	9.191	108.802

Figura 4.13: ***Tabella dei segnali che possono essere considerate come orologi o di sincronia nella Smart Wacth Dog***

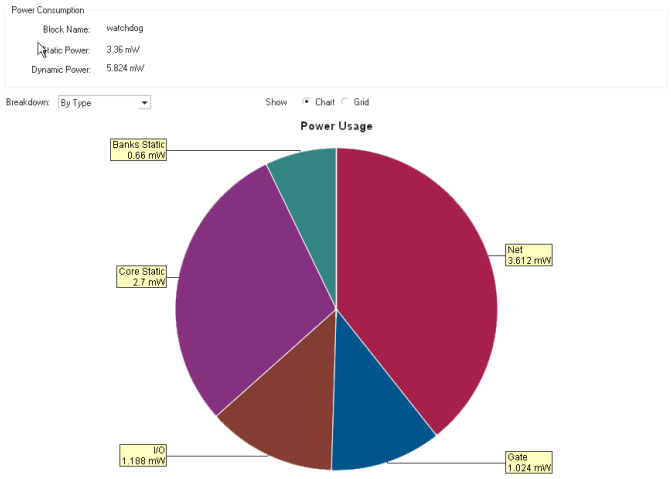


Figura 4.14: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo di ogni area della FPGA in stato attivo con gli orologi di 20 MHz***

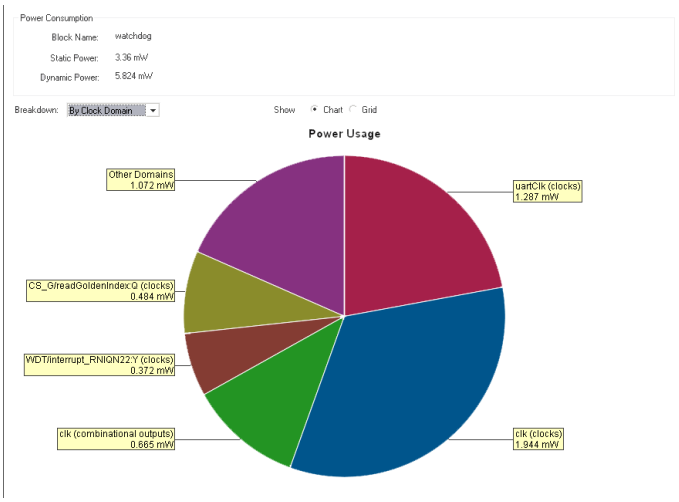


Figura 4.15: ***Grafico di consumo di potenza per tutto il sistema, considerando il consumo dovuto ai diversi orologi in stato attivo con una frequenza di 20 MHz***

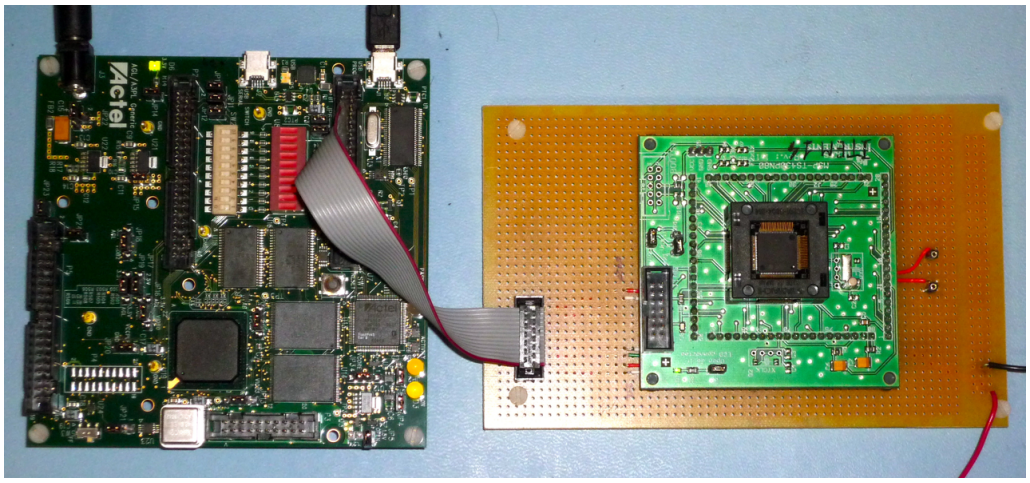


Figura 4.16: ***Circuito FPGA e CPU inter-connessi***

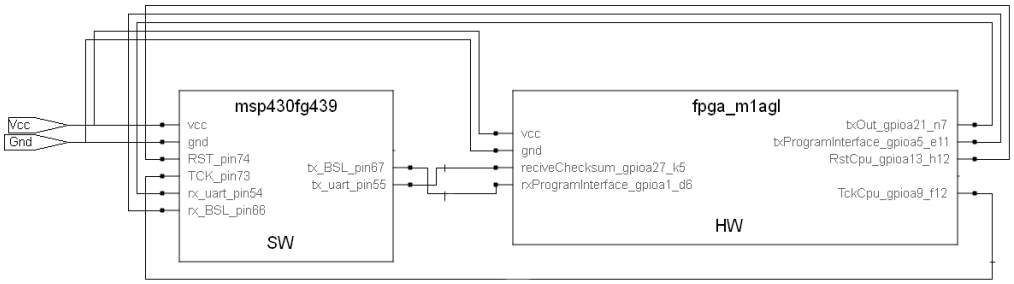


Figura 4.17: ***Schematico del Circuito FPGA e CPU inter-connessi***

Capitolo 5

Conclusioni

Per chiudere la discussione di questo lavoro di tesi, si può affermare che l'oggettivo principale di sviluppare una tecnica per mitigare gli effetti delle radiazioni su un processore commerciale è stato raggiunto soddisfacentemente.

Si è elaborato un sistema che è in grado di determinare se si sono prodotti errori all'interno della memoria operativa del processore, di contabilizzarli e correggerli. Si può aggiungere che è un sistema preciso e veloce; che ha un consumo di potenza molto basso e che può essere implementato in un qualsiasi ambito.

La tecnica sviluppata risponde alle caratteristiche di basso costo e basso consumo di potenza, consone con quelle del progetto AraMiS per il quale è stato svolto.

Nello sviluppo della tecnica per mitigare gli effetti delle radiazioni sono stati ricercati, utilizzati e collaudati componenti che anche essendo COTS possono essere introdotti nelle applicazioni rad-hard.

Questa tecnica può essere arricchita facendo uso delle tecniche di progettazione conosciute per i sistemi microelettronici (es. TMR).

Un lavoro futuro e quello di sviluppare un dispositivo che permetta di avere un sistema ridondante conformato di un μ-processore e di una Smart Wacth Dog, che possano essere attivati nel caso in cui i danni prodotti dalle radiazioni siano irreparabili tramite la riprogrammazione. (es. SEB).

Appendice A

Firmware Smart Wacth Dog

In questo capitolo si elencano i codici sorgenti dei firmware del μ-processore e della FPGA, le spiegazioni del codice sono all'interno dello stesso, in lingua inglese per standard.

A.1 Per il MSP430

Questo codice è stato generato automaticamente tramite i tool di progettazione del Visual Paradigm dalle condizioni e istruzioni fatte in ambiente UML.

Sorgente 1: Classe SW_WatchDog del μ-controllore. file:`./Codice/SWWatchDog.cpp`

```
1  #include "SW_WatchDog.h"
2
3  #include "Serial_Interface.h"
4  #include "ChecksumGenerator.h"
5
6
7
8
9  bool SmartWatchDog_CPU::SW_WatchDog::check() {
10         checksum = checksumGenerator.createChecksum();
11         serialInterface.sendChecksum(checksum);
12 }
```

Sorgente 2: Classe SW_WatchDog del μ-controllore. file:`./Codice/SWWatchDog.h`

```
1  #pragma diag_suppress=Pa050
2  #ifndef __SW_WatchDog_h__
3  #define __SW_WatchDog_h__
4
5  #include "platform.h"
6
7  #include "Serial_Interface.h"
8  #include "ChecksumGenerator.h"
9
10 namespace SmartWatchDog_CPU
11 {
12     class Serial_Interface;
13     class ChecksumGenerator;
14     class SW_WatchDog;
15 }
16
17 namespace SmartWatchDog_CPU
18 {
19     class SW_WatchDog
20     {
21         private: SmartWatchDog_CPU::Serial_Interface serialInterface;
22         private: SmartWatchDog_CPU::ChecksumGenerator checksumGenerator;
23         private: ushort checksum;
24         private: SmartWatchDog_CPU::SW_WatchDog* x_SW_WatchDog;
25
26
27         public: bool check();
28     };
29 }
30
31 #endif
```

Sorgente 3: Classe ChecksumGenerator del μ-controllore. file:`./Codice/ChecksumGenerator.cpp`

```
1  #include "ChecksumGenerator.h"
2
3  #include "Counter.h"
4
5
6
7  ushort* SmartWatchDog_CPU::ChecksumGenerator::checksumKeyTable;
8
9  ushort SmartWatchDog_CPU::ChecksumGenerator::createChecksum() {
10         unsigned short *stateWdCounter;
11         unsigned short *isBeenCheck;
12         const ushort NBYTEPERBLOCK = 86;
13             const ushort checksumKeyTable[]= {0xabab,0xd8a1,0x6de7,0x8421,0x335b,0
                   xfeec,0xa6f8,0xb2b9,0x6343,0x8905};
14
15         if (blockChecked == false) {
16           stateWdCounter = wdCounter.getValue();
17           isBeenCheck = (unsigned short *)(stateWdCounter[1]);
18           numBlockToCheck = stateWdCounter[1] + NBYTEPERBLOCK;
19           blockChecked = true;
20           flagCycle = true;
21           checksum = 0;
22          }
23          while (flagCycle == true){
24              if (isBeenCheck < (unsigned short *)(numBlockToCheck)){
25                 checksum ^= *isBeenCheck;
26                 isBeenCheck++;
27                 if (isBeenCheck == (unsigned short *)(numBlockToCheck)){
28                 checksum ^= checksumKeyTable[stateWdCounter[0]];
29                 blockChecked = false;
30                 flagCycle = false;
31               }
32          }
33         }
34         return checksum;
35 }
36
37 SmartWatchDog_CPU::ChecksumGenerator::ChecksumGenerator() {
38     blockChecked = false;
39     flagCycle = false;
40         blockChecked = false;
41         flagCycle = false;
42 }
43
44 void SmartWatchDog_CPU::ChecksumGenerator::~ChecksumGenerator() {
45 //  throw "Not yet implemented";
46 }
```

Sorgente 4: Classe ChecksumGenerator del μ-controllore. file:`./Codice/ChecksumGenerator.h`

```
1  #pragma diag_suppress=Pa050
2  #ifndef __ChecksumGenerator_h__
3  #define __ChecksumGenerator_h__
4
5  #include "platform.h"
6
7  #include "Counter.h"
8
9  namespace SmartWatchDog_CPU
10 {
11     class Counter;
12     class ChecksumGenerator;
13 }
14
15 namespace SmartWatchDog_CPU
16 {
17     class ChecksumGenerator
18     {
19         private: SmartWatchDog_CPU::Counter wdCounter;
20         /**
21          * This is a constant "look up table"
22          */
23         private: static ushort* checksumKeyTable;
24         private: static byte const NBYTEPERBLOCK = 84;
25         private: ushort numBlockToCheck;
26         private: bool blockChecked;
27         private: bool flagCycle;
28         private: ushort totalBlocks;
29         private: ushort checksum;
30
31         /// <summary>
32         /// Generates checksum of sequence specified by sequence.
33         /// </summary>
34
35         public: ushort createChecksum();
36
37
38         public: ChecksumGenerator();
39
40
41         public: void ~ChecksumGenerator();
42     };
43 }
44
45 #endif
```

Sorgente 5: Classe Counter del μ-controllore. file:./Codice/Counter.cpp

```
1  #include "Counter.h"
2
3
4
5  ushort* SmartWatchDog_CPU::Counter::getValue() {
6       ushort const FIRSTSTATE = 0;
7          ushort const NUMBLOCKS = 9;
8          ushort const FIRSTBLOCK = 4352;
9              const ushort NBYTEPERBLOCK = 86;
10         static ushort generalState[2];
11         if(state <= NUMBLOCKS) {
12            generalState[0] = state;
13            generalState[1] = blockState;
14           ++state;
15            blockState += NBYTEPERBLOCK;
16         }
17         else {
18            state = FIRSTSTATE;
19            blockState = FIRSTBLOCK;
20         }
21         return (generalState);
22 }
23
24 SmartWatchDog_CPU::Counter::Counter() {
25     state = FIRSTSTATE;
26     blockState = FIRSTBLOCK;
27     ushort const FIRSTSTATE = 0;
28              ushort const FIRSTBLOCK = 4352;
29         state = FIRSTSTATE;
30         blockState = FIRSTBLOCK;
31         //these values are the start of the blocks counter (state)
32         //and the address to the first memory place to check (blockState)
33 }
```

Sorgente 6: Classe Counter del μ-controllore. file:`./Codice/Counter.h`

```
1  #pragma diag_suppress=Pa050
2  #ifndef __Counter_h__
3  #define __Counter_h__
4
5  #include "platform.h"
6
7  namespace SmartWatchDog_CPU
8  {
9      class Counter;
10 }
11
12 namespace SmartWatchDog_CPU
13 {
14     class Counter
15     {
16         private: static ushort const FIRSTSTATE = 0;
17         private: static ushort const NUMBLOCKS = 9;
18         private: static ushort const FIRSTBLOCK = 4352;
19         private: ushort state;
20         private: ushort blockState;
21
22         /// <summary>
23         /// Returns counter value and Increment counter
24         /// </summary>
25
26         public: ushort* getValue();
27
28
29         public: Counter();
30     };
31 }
32
33 #endif
```

Sorgente 7: Classe Serial_Interface del μ-controllore. file:./Codice/SerialInterface.cpp

```
1  #include "Serial_Interface.h"
2
3
4  void SmartWatchDog_CPU::Serial_Interface::sendChecksum(ushort checksum) {
5      TXBUF0 = checksum;
6  }
7
8  SmartWatchDog_CPU::Serial_Interface::Serial_Interface() {
9          //  Baud rate divider with 1048576hz = 1048576Hz/9600 = ~109 (06Dh|08h)
10         //  ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = default DCO = 32 x ACLK =
               1048576Hz
11         //  //* An external watch crystal between XIN & XOUT is required for ACLK
               *//
12
13           FLL_CTL0 |= XCAP18PF;                  // Configure load caps
14           P2SEL |= 0x30;                             // P2.4,5 = USART0 TXD/RXD
15           ME1 |= UTXE0 + URXE0;                 // Enable USART0 TXD/RXD
16               UCTL0 &= 0x00;                       // Clear register control
17           UCTL0 |= CHAR;                          // 8-bit character
18               UCTL0 |= PENA;                         // Parity Enable
19               UCTL0 |= PEV;                          // Parity even
20               UCTL0 |= SPB;                          // One Stop Bit
21           UTCTL0 |= SSEL1;                       // UCLK = SMCLK
22           UBR00 = 0x6D;                           // 1MHz 115200
23           UBR10 = 0x00;                           //
24           UMCTL0 = 0x04;                        // Modulation
25           UCTL0 &= ~SWRST;                     // Initialize USART state machine
26           IE1 |= URXIE0;                             // Enable USART0 RX interrupt
27 }
```

Sorgente 8: Classe Serial_Interface del μ-controllore. file:`./Codice/SerialInterface.h`

```
1  #pragma diag_suppress=Pa050
2  #ifndef __Serial_Interface_h__
3  #define __Serial_Interface_h__
4
5  #include "platform.h"
6
7  namespace SmartWatchDog_CPU
8  {
9      class Serial_Interface;
10 }
11
12 namespace SmartWatchDog_CPU
13 {
14     class Serial_Interface
15     {
16
17         /// <summary>
18         /// It sends checksum to the FPGA.
19         /// </summary>
20
21         public: void sendChecksum(ushort checksum);
22
23
24         public: Serial_Interface();
25     };
26 }
27
28 #endif
```

Sorgente 9: Configurazioni per la piattaforma specifica del MSP430FG439. file:./Codice/platform.h

```
#pragma diag_suppress=Pa050
#define MSP430 defined (__MSP430F149__) || defined (__MSP430FG439__)
#define CHIPCON false

//#include <intrinsics.h>
//#include <msp430.h>
#include "io430xG43x.h"
#define byte unsigned char
#define ushort unsigned short
#define ulong unsigned long
#define uint unsigned int
#define __abstract
#define volatile_byte volatile byte
#define port_address volatile unsigned char * const

#define def_CMD_WRITE_DATA
#define def_CMD_SET_CONFIGURATION

//#define def_CMD_READ_DATA
//#define def_CMD_GET_HOUSEKEEPING
//#define def_CMD_GET_HISTORY
//#define def_CMD_GET_STATISTICS
//#define def_CMD_GET_CONFIGURATION
//#define def_CMD_SET_CONFIGURATION

//#define def_CMD_STANDBY
//#define def_CMD_WAKEUP
//#define def_CMD_RESET_STATISTICS

//#define NULL            0       /* 0L if pointer same as long */

#define TBD 4

#define __DEBUG

namespace std;//{};//SmartWatchDog_CPU{};//std{};
```

A.2 Per la FPGA

Questo codice è stato scritto in base alle condizione ed istruzioni fatte in ambiente UML, sfortunatamente ancora non abbiamo a disposizione un generatore automatico di codice VHDL abbastanza robusto come per implementare di maniera automatizzata il codice dai diagrammi UML.

Sorgente 10: Classe Op_Code della FPGA.

```
1  library ieee;
2      use ieee.std_logic_1164.all;
3      use ieee.std_logic_arith.all;
4     -- use ieee.std_logic_unsigned.all;
5      use ieee.std_logic_signed.all;
6
7  package op_code is
8
9    constant LENGTHPROM              : integer := 16;   -- Length in bits of the prom
         memory address
10   constant LENGTHPASSBUF           : integer := 30;  -- Length in bits of the
         Password Buffer
11   constant SENDPASSWORDCOUNT       : integer := 31;  -- Number of byte to send the
         password to the BSL unit + 1
12   constant SENDBLOCKDATACOUNT      : integer := 261;  -- Number of byte to send the
         data block to the BSL unit + 1
13   constant LENGTHBLOCKDATA         : integer := 262;  -- Lenght in bytes of th
         blockData buffer
14   constant ERASEMASSMEMCOUNT       : integer := 11;  -- Number of byte to send the
         mass erase memory comand to the BSL unit + 1
15   constant LENGTHERASEMASSMEM      : integer := 10;  -- Length in bits of the erase
         mass memory buffer
16   constant LENGTHRXBLKDAT_HDR      : integer := 8;  -- Length in bits of the
         rxBlockData header buffer
17   constant COUNTER                 : integer := 21;          -- Length in bits of the
          WDTCounter
18   constant TIMERLIMIT              : integer := 200000;  -- Limit of reboot counter
         timer to count 10ms whit a clk of 20MHz
19   constant BSLDELAY                : integer := 24000;  -- number to have a delay of
         1.2ms for each bls action whit a clk of 20MHz
20   constant WatchDogTimerC          : std_logic_vector(COUNTER - 1 downto 0) := "
         11111111111111111111";  -- Time that the Watchdog Timer must count to
         generate an interrupt
21   constant LINDEX                  : integer := 16;       -- Length in bits of the
         Index counter
22   constant LGOLDINDEX              : integer := 16;  -- Length in bits of the golden
         index
23   constant LHALFGOLDINDEX          : integer := 8;  -- Length in bits of the half
         golden index
24   constant WORDCPU                 : integer := 16;      -- legnth in bits of cpu word
25   constant HALFWORD                : integer := 8;      -- Length in bits of cpu half
         word
26   constant UARTCOUNTER             : integer := 4;  -- Number of uart counter bits
27   constant HALFBYTE                : integer := 4;
28   constant TWOBITS                 : integer := 2;
29   constant BYTE                    : integer := 8;
30   constant LLSFR                   : integer := 5;         -- Legnth in bits of LSFR
         state
31   constant FSTATELSFR              : std_logic_vector (LLSFR -1 downto 0) := "10000";
          -- First state of LSFR
32   constant NUMBLOCKS               : integer := 244;  -- Number of blocks to check
```

```
33   constant TIMER2RCLIMIT          : integer := 200000000; -- limit of reboot
         counter's timer2 to count 10 seconds with a clk of 20MHz
34   constant NBYTEPERBLOCK          : integer := 250; -- Byte per block numbers
35   constant TOP64KB                : integer := 65536; -- Number of bytes in 64kB
36   constant MEMSIZE                : integer := TOP64KB;     -- Number of memory bytes
37   constant FIRSTADDMEM            : integer := 4352; -- First address of CPU's
         operative
38                                               -- memory h(1100)
39   end package;
```

Sorgente 11: Classe Watchdog della FPGA.

```
1  ----------------------------------------------------------------------
2  -- Title      : WatchDog
3  -- Project    : Aramis
4  ----------------------------------------------------------------------
5  -- File       : WatchDog.vhd
6  -- Author     : Tomas Antonio Alarcon Rivera
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-08-27
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------
13 -- Description: It's the Package than contains all subclass than belong to the
14 -- Watch dog
15 ----------------------------------------------------------------------
16 -- Copyright (c) 2009
17 ----------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-08-27  1.0      Tomas Alarcon   Created
21 ----------------------------------------------------------------------
22
23 library ieee;
24 use ieee.std_logic_1164.all;
25
26 library ieee;
27 library proasic3;
28 use proasic3.all;
29
30 library work;
31 use work.Op_Code.all;
32
33
34 entity watchdog is
35
36   port (
37     receiveChecksum     : in  std_logic;  --rx serial interface
38     rxProgramInterface  : in  std_logic;
39     clk                 : in  std_logic;
40     uartClk             : in  std_logic;
41     Grst                : in  std_logic;
42     rxMem               : in  std_logic;
43     txMem               : out std_logic;
44     RstCpu              : out std_logic;
45     TckCpu              : out std_logic;
46     txOut               : out std_logic;  -- tx serial interface
47     TxProgramInterface  : out std_logic);
48
49 end watchdog;
50
51 architecture aramis of watchdog is
52 --  attribute syn_preserve : boolean;
53 --  attribute syn_preserve of aramis: architecture is true;
54   component serialinterface
55     port (
56       receiveChecksum             : in std_logic;  -- rx
57       clk                         : in std_logic;
58       uartClk                     : in std_logic;
59       Grst                        : in std_logic;
60       checksumRead                : in std_logic;
```

```
61      sendStateWatchdog           : in std_logic;
62      stateWatchdog               : in std_logic_vector(WORDCPU - 1 downto 0);
63      txOut                       : out std_logic;
64      resetWatchDogTimer          : out std_logic;
65      comparisonSignatures        : out std_logic;
66      checksum                    : out std_logic_vector( LGOLDINDEX - 1 downto 0))
            ;
67  end component;
68
69  component WatchdogTimer
70    port (
71      resetWatchDogTimer : in  std_logic;  -- Make to zero the watchdog timer
72      clk                : in  std_logic;
73      Grst               : in  std_logic;  -- General Reset of all system
74      enableWDT          : in  std_logic;  -- Enable de counter of WDT (active high
            )
75      interrupt          : out std_logic); -- Signal that requests the interrupt
            and the reset the CPU
76  end component;
77
78  component ComparisonSignaturePlusGonIdenIndex
79    port (
80      comparisonSignatures          : in std_logic;
81      Grst                          : in std_logic;
82      clk                           : in std_logic;
83      checksum                      : in std_logic_vector (LGOLDINDEX -1 downto 0);
84      goldenIndex                   : in std_logic_vector (LGOLDINDEX -1 downto 0);
85      goldenIndexIsReady            : in std_logic;
86      checksumRead                  : out std_logic;
87      readGoldenIndex               : out std_logic;
88      incrementRebootCounter        : out std_logic);
89  end component;
90
91  component IndexCounter
92    port (
93      resetIndexCounter     : in  std_logic;
94      readIndex             : in  std_logic;
95      readGoldenIndex       : in  std_logic;
96      index                 : out integer range 0 to NUMBLOCKS;
97      clk                   : in  std_logic;
98      goldenIndex           : out std_logic_vector (LGOLDINDEX-1 downto 0);
99      goldenIndexIsReady    : out std_logic;
100     Grst                  : in  std_logic);
101 end component;
102
103 component RebootCounter
104   port (
105     incrementRbCounter : in  std_logic;   -- Increment order
106     resetRebootCounter : in  std_logic;   -- Make to zero the reboot counter
107     clk                : in  std_logic;
108     Grst               : in  std_logic;   -- General reset of all system
109     resetCPU           : out std_logic;   -- Signal that requests the CPU reset
110     reloadFlash        : out std_logic);  -- Signal that requests the code reload
            on the memory flash
111 end component;
112
113 component ProgramControl
114   port (
115   reloadFlash                 : in  std_logic;
116   Grst                        : in  std_logic;
117   clk                         : in  std_logic;
118   passwordInfo                : in  std_logic_vector (TWOBITS - 1 downto 0);
```

```
119     writeMemoryDone             : in  std_logic;
120     sendPassword                : out std_logic;
121     readIndex                   : out std_logic;
122     writeAllMemory              : out std_logic;
123     writePartMemory             : out std_logic;
124     boutonAddressMemory         : out std_logic_vector (WORDCPU - 1 downto 0);
125     writeFlash                  : out std_logic;
126     resetRebootCounter          : out std_logic;
127     resetIndexCounter           : out std_logic;
128     enableWDT                   : out std_logic;
129     index                       : in integer range 0 to NUMBLOCKS);
130   end component;
131
132   component ProgramInterface
133     port (
134     Grst                : in  std_logic;
135     clk                 : in  std_logic;
136     uartClk             : in  std_logic;
137     writeFlash          : in  std_logic;
138     sendPassword        : in  std_logic;
139     resetCPU            : in  std_logic;  -- Active low
140     rx                  : in  std_logic;
141     writeAllMemory      : in  std_logic;
142     writePartMemory     : in  std_logic;
143     boutonAddressProm   : in  std_logic_vector (WORDCPU -1 downto 0);
144     stateMemoryProm     : in  std_logic_vector (BYTE -1 downto 0);
145     dataReady           : in  std_logic;
146
147     writeMemoryDone     : out std_logic;
148     address             : out integer range 0 to MEMSIZE -1;
149     getState            : out std_logic;
150     PasswordInfo        : out std_logic_vector (TWOBITS - 1 downto 0);
151     RstCpu              : out std_logic;
152     TckCpu              : out std_logic;
153     tx                  : out std_logic;
154     sendStateWatchdog   : out std_logic;
155     stateWatchdog       : out std_logic_vector(WORDCPU -1 downto 0));
156   end component;
157
158   component memoryrom
159     port (
160     Grst            : in  std_logic;
161     clk             : in  std_logic;
162     getState        : in  std_logic;
163     address         : in  integer range 0 to MEMSIZE - 1;
164     rxMem           : in  std_logic;
165     txMem           : out std_logic;
166     stateMemoryProm : out std_logic_vector(BYTE -1 downto 0);
167     dataReady       : out std_logic);
168   end component;
169
170
171
172
173   -- Serial Interface's signals
174   signal  checksumReadS               : std_logic;
175   signal  sendStateWatchdogS          : std_logic;
176   signal  stateWatchdogS              : std_logic_vector(WORDCPU - 1 downto 0);
177   signal  resetWatchDogTimerS         : std_logic;
178   signal  comparisonSignaturesS       : std_logic;
179   signal  checksumS                   : std_logic_vector(LGOLDINDEX - 1 downto 0);
180   -- Watchdog Timer's signals
```

```
181   signal  interruptS                    : std_logic;
182   signal  enableWDTS                    : std_logic;
183   -- ComparisonSignature&gonIdenindex's signals
184   signal  goldenIndexS                  : std_logic_vector (LGOLDINDEX -1 downto 0);
185   signal  goldenIndexIsReadyS           : std_logic;
186   signal  readGoldenIndexS              : std_logic;
187   signal  incrementRebootCounterS       : std_logic;
188   -- Index Counter's signals
189   signal  resetIndexCounterS            : std_logic;
190   signal  readIndexS                    : std_logic;
191   signal  indexS                        : integer range 0 to NUMBLOCKS;
192   -- RebootCounter's signals
193   signal  IncrementRbCounterS           : std_logic;
194   signal  resetRebootCounterS           : std_logic;
195   signal  resetCPUS                     : std_logic;
196   signal  reloadFlashS                  : std_logic;
197   -- PromAccess's signals
198   signal  passwordInfoS                 : std_logic_vector (TWOBITS - 1 downto 0);
199   signal  writeMemoryDoneS              :  std_logic;
200   signal  sendPasswordS                 :  std_logic;
201   signal  writeAllMemoryS               :  std_logic;
202   signal  writePartMemoryS              :  std_logic;
203   signal  boutonAddressMemoryS          :  std_logic_vector (WORDCPU - 1 downto 0);
204   signal  writeFlashS                   :  std_logic;
205   -- ProgramInterface's signals
206   signal  stateMemoryPromS              : std_logic_vector(BYTE - 1 downto 0);
207   signal  addressS                      : integer range 0 to MEMSIZE - 1;
208   signal  dataReadyS                    : std_logic;
209   signal  getStateS                     : std_logic;
210
211 begin  -- aramis
212
213   SI:serialinterface
214     port map (
215       receiveChecksum       => receiveChecksum,
216       clk                   => clk,
217       uartClk               => uartClk,
218       Grst                  => Grst,
219       checksumRead          => checksumReadS,
220       sendStateWatchdog     => sendStateWatchdogS,
221       stateWatchdog         => stateWatchdogS,
222       txOut                 => txOut,
223       resetWatchDogTimer    => resetWatchDogTimerS,
224       comparisonSignatures  => comparisonSignaturesS,
225       checksum              => checksumS);
226   WDT:WatchdogTimer
227     port map (
228       resetWatchDogTimer => resetWatchDogTimerS,
229       clk                => clk,
230       Grst               => Grst,
231       enableWDT          => enableWDTS,
232       interrupt          => interruptS);
233   CS_G:ComparisonSignaturePlusGonIdenIndex
234     port map (
235       comparisonSignatures  => comparisonSignaturesS,
236       Grst                  => Grst,
237       clk                   => clk,
238       checksum              => checksumS,
239       goldenIndex           => goldenIndexS,
240       goldenIndexIsReady    => goldenIndexIsReadyS,
241       checksumRead          => checksumReadS,
242       readGoldenIndex       => readGoldenIndexS,
```

```
243         incrementRebootCounter=> incrementRebootCounterS);
244     IC:IndexCounter
245       port map (
246         resetIndexCounter     => resetIndexCounterS,
247         readIndex             => readIndexS,
248         readGoldenIndex       => readGoldenIndexS,
249         index                 => indexS,
250         clk                   => clk,
251         goldenIndex           => goldenIndexS,
252         goldenIndexIsReady    => goldenIndexIsReadyS,
253         Grst                  => Grst);
254     RC:RebootCounter
255       port map (
256         incrementRbCounter => IncrementRbCounterS,
257         resetRebootCounter => resetRebootCounterS,
258         clk                => clk,
259         Grst               => Grst,
260         resetCPU           => resetCPUS,
261         reloadFlash        => reloadFlashS);
262     PA:ProgramControl
263       port map (
264         reloadFlash          => reloadFlashS,
265         Grst                 => Grst,
266         clk                  => clk,
267         passwordInfo         => passwordInfoS,
268         writeMemoryDone      => writeMemoryDoneS,
269         sendPassword         => sendPasswordS,
270         readIndex            => readIndexS,
271         writeAllMemory       => writeAllMemoryS,
272         writePartMemory      => writePartMemoryS,
273         boutonAddressMemory => boutonAddressMemoryS,
274         writeFlash           => writeFlashS,
275         resetRebootCounter   => resetRebootCounterS,
276         resetIndexCounter    => resetIndexCounterS,
277         enableWDT            => enableWDTS,
278         index                => indexS);
279     PI:ProgramInterface
280       port map (
281         Grst               => Grst,
282         clk                => clk,
283         uartClk            => uartClk,
284         writeFlash         => writeFlashS,
285         sendPassword       => sendPasswordS,
286         resetCPU           => resetCPUS,
287         rx                 => rxProgramInterface,
288         writeAllMemory     => writeAllMemoryS,
289         writePartMemory    => writePartMemoryS,
290         boutonAddressProm => boutonAddressMemoryS,
291         stateMemoryProm    => stateMemoryPromS,
292         dataReady          => dataReadyS,
293         writeMemoryDone    => writeMemoryDoneS,
294         address            => addressS,
295         getState           => getStateS,
296         passwordInfo       => passwordInfoS,
297         RstCpu             => RstCpu,
298         TckCpu             => TckCpu,
299         tx                 => TxProgramInterface,
300         sendStateWatchdog => sendStateWatchdogS,
301         stateWatchdog      => stateWatchdogS);
302     MR:memoryrom
303       port map (
304         Grst             => Grst,
```

```
305         clk              => clk,
306         getState         => getStateS,
307         address          => addressS,
308         rxMem            => rxMem,
309         txMem            => txMem,
310         stateMemoryProm  => stateMemoryPromS,
311         dataReady        => dataReadyS);
312
313
314
315     -- purpose: generate the increment the reboot counter siganl's
316     -- type    : combinational
317     -- inputs : interruptS, incrementRebootCounterS
318     -- outputs: IncrementRbCounterS
319     IncrementRBC: process (interruptS, incrementRebootCounterS, Grst)
320     begin  -- process IncrementRBC
321       if interruptS = '0' or incrementRebootCounterS = '0' then
322         IncrementRbCounterS <= '0';
323       elsif (interruptS ='1' and incrementRebootCounterS = '1') or Grst = '0' then
324         IncrementRbCounterS <= '1';
325       end if;
326     end process IncrementRBC;
327
328
329
330 end aramis;
```

Sorgente 12: Classe WatchdogTimer della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : WatchdogTimer
3  -- Project    : Aramis satellite
4  ----------------------------------------------------------------------------
5  -- File       : WatchdogTimer.vhd
6  -- Author     : Tomas A. Alarcon R.
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-04-02
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Timer should generate an interrupt
14 ----------------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author  Description
19 -- 2009-04-02  1.0      TAlarcon    Created
20 ----------------------------------------------------------------------------
21 library ieee;
22 use ieee.std_logic_arith.all;
23 use ieee.std_logic_1164.all;
24 use ieee.std_logic_unsigned.all;
25
26 library work;
27 use work.Op_Code.all;
28
29
30 entity WatchdogTimer is
31
32   port (
33     resetWatchDogTimer : in  std_logic;  -- Make to zero the watchdog timer
34     clk                : in  std_logic;
35     Grst               : in  std_logic;  -- General Reset of all system
36     enableWDT          : in  std_logic;  -- Enable de counter of WDT (active high)
37     interrupt          : out std_logic); -- Signal that requests the interrupt and
           the reset the CPU
38 end WatchdogTimer;
39
40 architecture Aramis of WatchdogTimer is
41   signal WDTCounter : std_logic_vector(COUNTER-1 downto 0);  -- Signal for counter
42
43 begin  -- Aramis
44
45   -- purpose: Counter from zero to WatchDogTimerC
46   -- type   : sequential
47   -- inputs : clk, Grst, WDTCounter
48   -- outputs: Interrupt
49   Counter: process (clk, Grst)
50   begin  -- process Counter
51     if Grst = '0' then                  -- asynchronous reset (active low)
52       WDTCounter <= (others => '0');
53       interrupt <= '1';
54     elsif clk'event and clk = '1' then  -- rising clock edge
55       if enableWDT = '1' then
56         if resetWatchDogTimer = '0' then
57           WDTCounter <= (others => '0');
58           interrupt <= '1';
59         elsif WDTCounter = WatchDogTimerC then
```

```
60            interrupt <= '0';
61            WDTCounter <= (others => '0');
62          else
63            WDTCounter <= WDTCounter + 1;
64            interrupt <= '1';
65          end if;
66        end if;
67      end if;
68    end process Counter;
69  end Aramis;
```

Sorgente 13: Classe Serial_Interface_FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title       : SerialInterface
3  -- Project     : Aramis
4  ----------------------------------------------------------------------------
5  -- File        : SerialInterface.vhd
6  -- Author      : Tomas Antonio Alarcon Rivera
7  -- Company     : Politecnico di Torino
8  -- Created     : 2009-08-25
9  -- Last update: 2009-11-02
10 -- Platform    : Actel
11 -- Standard    : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Iunit than contains the comand for the comunicatios whit the
14 -- CPU, using the UART protocol
15 ----------------------------------------------------------------------------
16 -- Copyright (c) 2009
17 ----------------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-08-25  1.0      Tomas Alarcon   Created
21 ----------------------------------------------------------------------------
22
23 library ieee;
24 use ieee.std_logic_1164.all;
25 use ieee.std_logic_arith.all;
26 use ieee.std_logic_signed.all;
27
28 library work;
29 use work.Op_Code.all;
30
31
32 entity serialinterface is
33
34   port (
35     receiveChecksum             : in std_logic;  -- rx
36     clk                         : in std_logic;
37     uartClk                     : in std_logic;
38     Grst                        : in std_logic;
39     checksumRead                : in std_logic;
40     sendStateWatchdog           : in std_logic;
41     stateWatchdog               : in std_logic_vector(WORDCPU - 1 downto 0);
42     txOut                       : out std_logic;
43     resetWatchDogTimer          : out std_logic;
44     comparisonSignatures        : out std_logic;
45     checksum                    : out std_logic_vector( LGOLDINDEX - 1 downto 0));
46
47 end serialinterface;
48
49
50 architecture Aramis of serialinterface is
51   attribute syn_preserve : boolean;
52 --  attribute syn_preserve of Aramis: architecture is true;
53   component uart
54     generic (
55       BAUDRATEDIV : integer range 0 to 65535 :=781);  -- Baud rate divisor
56                                                       -- f(uartClk=48MHz)
                                                            /781/4=~9600 Br
57
58     port (
59       clk          : in  std_logic;         -- Clock of general system
```

```
60        Grst         : in  std_logic;         -- Global reset
61        txDataL      : in  std_logic;         -- Async load transmit buffer active
62                                              -- high
63        rxDataR      : in  std_logic;         -- Async read receive buffer
64        uartClk      : in  std_logic;         -- Clock used for Transmit/Receive
65        rx           : in  std_logic;         -- Rx
66        dataTxIn     : in  std_logic_vector (BYTE - 1 downto 0); -- Data to transmit
67        interruptTx  : out std_logic; -- Transmit interrupt: indicate waiting for
              byte
68        interruptRx  : out std_logic; -- Recive interrupt: indicate Byte received
69        tx           : out std_logic;         -- TX
70        rxDataOut    : out std_logic_vector (BYTE -1 downto 0)); -- Byte received
71    end component;
72
73
74    signal  loadTxDataS           : std_logic;
75    signal  txDataS               : std_logic_vector (BYTE -1 downto 0);
76    signal  rxDataS               : std_logic_vector (BYTE - 1 downto 0);
77    signal  uLoadRxDataS          : std_logic;
78    signal  interruptTxS          : std_logic;
79    signal  interruptRxS          : std_logic;
80    signal  interruptRxS2         : std_logic;
81    signal  stateWatchdogS        : std_logic_vector (WORDCPU -1 downto 0);
82    attribute syn_preserve of stateWatchdogS : signal is true;
83    signal  sendStateWdCount      : integer range 4 downto 0;
84    attribute syn_preserve of sendStateWdCount : signal is true;
85    signal  checksumArriveCount   : integer range 0 to 1;
86  begin  -- Aramis
87
88    UA1:uart
89      port map (
90        clk          => clk,
91        Grst         => Grst,
92        txDataL      => loadTxDataS,
93        rxDataR      => uLoadRxDataS,
94        uartClk      => uartClk,
95        rx           => receiveChecksum,
96        dataTxIn     => txDataS,
97        interruptTx  => interruptTxS,
98        interruptRx  => interruptRxS,
99        tx           => txOut,
100       rxDataOut    => rxDataS);
101
102 -- purpose: This process descrive
103 -- type   : sequential
104 -- inputs : Grst, clk
105 -- outputs: comparisonsSignatures, checksum, resetWatchDogTimer
106   Serial_Inerface_Process: process (Grst, clk)
107
108   begin  -- process Serial_Inerface_Process
109     if Grst = '0' then
110       comparisonSignatures <= '1';
111       resetWatchDogTimer <= '1';
112       checksumArriveCount <= 0;
113       checksum <= (others => '0');
114       uLoadRxDataS <= '0';
115     elsif clk'event and clk = '1' then
116       if checksumRead = '1' then          -- Synchronous reset
117         if interruptRxS2 = '1' then
118           if checksumArriveCount = 0 then
119             uLoadRxDataS <= '1';
120             checksum(LHALFGOLDINDEX -1 downto 0) <= rxDataS;
```

```
121             checksumArriveCount <= 1;
122           elsif checksumArriveCount = 1 then
123             uLoadRxDataS <= '1';
124             checksum(LGOLDINDEX -1 downto LHALFGOLDINDEX) <= rxDataS;
125             comparisonSignatures <= '0';
126             resetWatchDogTimer <= '0';
127             checksumArriveCount <= 0;
128           end if;
129         else
130           uLoadRxDataS <= '0';
131         end if;
132       else
133         comparisonSignatures <= '1';
134         resetWatchDogTimer <= '1';
135         checksumArriveCount <= 0;
136         checksum <= (others => '0');
137         uLoadRxDataS <= '0';
138       end if;
139     end if;
140   end process Serial_Inerface_Process;
141
142 -- purpose: This process takes the stateWatchdog from the bus and it sends to the
      CPU
143 -- type    : sequential
144 -- inputs : sendStateWatchdog, Grst, stateWatchdog
145 -- outputs: txOut
146   sendStateWD_proc: process (clk, Grst)
147   begin  -- process sendStateWD_proc
148     if Grst = '0' then                    -- asynchronous reset (active low)
149       stateWatchdogS <= (others => '0');
150       sendStateWdCount <= 0;
151       loadTxDataS <= '0';
152       txDataS <= (others => '0');
153     elsif rising_edge(clk) then  -- rising clock edge
154       if sendStateWatchdog = '0' and sendStateWdCount = 0 then
155         stateWatchdogS <= stateWatchdog;
156         sendStateWdCount<= 1;
157       elsif sendStateWdCount = 1 and interruptTxS = '1' then
158         txDataS <= stateWatchdogS(HALFWORD -1 downto 0);
159         loadTxDataS <= '1';
160       elsif sendStateWdCount = 1 and interruptTxS = '0' then
161         sendStateWdCount<= 2;
162         loadTxDataS <= '0';
163       elsif sendStateWdCount = 2 and interruptTxS = '1' then
164         txDataS <= stateWatchdogS(WORDCPU -1 downto HALFWORD);
165         loadTxDataS <= '1';
166         sendStateWdCount <= 3;
167       elsif sendStateWdCount = 3 and interruptTxS = '0' then
168         loadTxDataS <= '0';
169         sendStateWdCount <= 4;
170       elsif sendStateWdCount = 3 and interruptTxS = '1' then
171         sendStateWdCount <= 0;
172       end if;
173
174     end if;
175   end process sendStateWD_proc;
176
177 -- purpose: generate signal for read data in just time
178 -- type    : sequential
179 -- inputs : interruptRxS, Grst, uLoadRxDataS
180 -- outputs: interruptRxS2
181   Inte: process (interruptRxS, Grst, uLoadRxDataS)
```

```
182   begin  -- process Inte
183     if Grst = '0' or uLoadRxDataS = '1' then                    -- asynchronous
          reset (active low)
184       interruptRxS2 <= '0';
185     elsif interruptRxS'event and interruptRxS = '1' then  -- rising clock edge
186       interruptRxS2<= '1';
187     end if;
188   end process Inte;
189
190
191
192
193 end Aramis;
```

Sorgente 14: Classe ComparisonSignaturePlusGoldenIndex della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : ComparisonSignaturePlusGoldenIndex
3  -- Project    : Aramis
4  ----------------------------------------------------------------------------
5  -- File       : ComparisonSignaturePlusGoldenIndex.vhd
6  -- Author     : Tomas Antonio Alarcon Rivera
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-08-27
9  -- Last update: 2009-10-20
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Iutit than doit the compararison beetwin the checksum and de
14 -- golden index
15 ----------------------------------------------------------------------------
16 -- Copyright (c) 2009
17 ----------------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-08-27  1.0      Tomas Alarcon   Created
21 ----------------------------------------------------------------------------
22 library ieee;
23     use ieee.std_logic_1164.all;
24     use ieee.std_logic_arith.all;
25     use ieee.std_logic_signed.all;
26
27 library work;
28       use work.Op_Code.all;
29
30 entity ComparisonSignaturePlusGonldenIndex is
31
32   port (
33     comparisonSignatures          : in std_logic;
34     Grst                          : in std_logic;
35     clk                           : in std_logic;
36     checksum                      : in std_logic_vector (LGOLDINDEX -1 downto 0);
37     goldenIndex                   : in std_logic_vector (LGOLDINDEX -1 downto 0);
38     goldenIndexIsReady            : in std_logic;
39     checksumRead                  : out std_logic;
40     readGoldenIndex               : out std_logic;
41     incrementRebootCounter        : out std_logic);
42 end ComparisonSignaturePlusGonldenIndex;
43
44
45 architecture Aramis of ComparisonSignaturePlusGonldenIndex is
46
47   signal readFlag : std_logic; -- It is the reset for the Serial Interface
48                                -- (Active high)
49
50 begin  -- Aramis
51
52   -- purpose: Develops the actions to do the comparison of the signatures
53   -- type   : sequential
54   -- inputs : comparisonSignatures, Grst, clk, goldenIndexIsReady
55   -- outputs: incrementIndexCounter, incrementRebootCounter, checksumRead
56   ComparisonSignatures_process: process (clk, Grst, comparisonSignatures)
57   begin  -- process ComparisonSignatures_process
58     if Grst = '0' or comparisonSignatures = '1' then
59       incrementRebootCounter <= '1';
60       checksumRead <= '1';
```

```
61        readGoldenIndex <= '1';
62        readFlag <= '0';
63      elsif rising_edge(clk) then
64        if comparisonSignatures = '0' then
65          readGoldenIndex <= '0';
66          if goldenIndexIsReady = '1' then
67            if readFlag = '0' then
68              if checksum = goldenIndex then
69                checksumRead <= '0';          -- It is the reset for the Serial
70                readFlag <= '1';              -- Interface (Active Low)
71              else
72                incrementRebootCounter <= '0';
73                checksumRead <= '0';
74                readFlag <= '1';
75              end if;
76            end if;
77          end if;
78        end if;
79      end if;
80    end process ComparisonSignatures_process;
81
82  end Aramis;
```

Sorgente 15: Classe IndexCounter della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : IndexCounter
3  -- Project    : Aramis
4  ----------------------------------------------------------------------------
5  -- File       : IndexCounter.vhd
6  -- Author     : Tomas Alarcon
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-04-28
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Is the index for sincronice the CPU whit the watch dog
14 ----------------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author  Description
19 -- 2009-04-28  1.0      TAlarcon    Created
20 ----------------------------------------------------------------------------
21
22 library ieee;
23 use ieee.std_logic_1164.all;
24 use ieee.std_logic_arith.all;
25 use ieee.std_logic_unsigned.all;
26 library work;
27 use work.Op_Code.all;
28
29 entity IndexCounter is
30
31   port (
32     resetIndexCounter     : in  std_logic;
33     readIndex             : in  std_logic;
34     readGoldenIndex       : in  std_logic;
35     index                 : out integer range 0 to NUMBLOCKS;
36     clk                   : in  std_logic;
37     goldenIndex           : out std_logic_vector (LGOLDINDEX-1 downto 0);
38     goldenIndexIsReady    : out std_logic;
39     Grst                  : in  std_logic);
40
41 end IndexCounter;
42
43 architecture Aramis of IndexCounter is
44
45   component lsfr
46     port (
47       Grst        : in  std_logic;        -- Global reset
48       resetLSFR   : in  std_logic;
49       getChecksum : in  std_logic;
50       checksum    : out std_logic_vector (LGOLDINDEX - 1 downto 0));
51   end component;
52
53   signal indexNumber : integer range 0 to NUMBLOCKS + 1;  -- The storage signal for
       the Index
54   signal indexNumberCopy : integer range 0 to NUMBLOCKS + 1;
55   signal goldenIndexS : std_logic_vector (LGOLDINDEX - 1 downto 0);
56   signal resetLSFRS : std_logic;
57
58 begin  -- Aramis
59
```

```
60    lsfr_checksum:lsfr
61      port map (
62        Grst         => Grst,
63        resetLSFR    => resetLSFRS,
64        getChecksum => readGoldenIndex,
65        checksum     => goldenIndexS);
66
67    -- purpose: Make zero the index number or Ask to the index number
68    -- type    : sequential
69    -- inputs : IncrementIndexCounter, Grst, ResetIndexCounter, ReadIndex
70    -- outputs: IndexNumber, Index
71    RI: process (clk, Grst, indexNumber, resetIndexCounter)
72    begin  -- process Counter
73      if Grst = '0' or resetIndexCounter = '0' then                    -- asynchronous
          reset (active low)
74        index <= 0;
75        indexNumberCopy <= 0;
76        goldenIndexIsReady <= '0';
77      elsif clk'event and clk= '1' then
78        if readIndex = '0' then
79          index <= indexNumber - 1;
80        end if;
81        if indexNumber /= indexNumberCopy then
82          goldenIndexIsReady <= '1';
83          indexNumberCopy <= indexNumber;
84        else
85          goldenIndexIsReady <= '0';
86        end if;
87      end if;
88    end process RI;
89    goldenIndex <= goldenIndexS;
90      -- purpose: Process for read the Golden Index and increment the counter
91      -- type    : sequential
92      -- inputs : readIndex
93      -- outputs: Index
94      ReadGIndex_P: process (readGoldenIndex, Grst, ResetIndexCounter)
95      begin  -- process ReadGINdex
96        if Grst = '0'  or resetIndexCounter = '0' then
97          indexNumber <= 0;
98          resetLSFRS <= '0';
99        elsif readGoldenIndex'event and readGoldenIndex = '0' then
100         resetLSFRS <= '1';
101         if indexNumber <= NUMBLOCKS then
102           indexNumber <= indexNumber + 1;
103         else
104           indexNumber <= 0;
105           resetLSFRS <= '0';          -- To reset LSFR (active low)
106         end if;
107       end if;
108     end process ReadGIndex_P;
109   end Aramis;
```

Sorgente 16: Classe LSFR della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title       : LSFR
3  -- Project     : Aramis
4  ----------------------------------------------------------------------------
5  -- File        : LSFR.vhd
6  -- Author      : Tomas Alarcon
7  -- Company     : Politecnico di Torino
8  -- Created     : 2009-09-30
9  -- Last update: 2009-11-02
10 -- Platform    : Actel
11 -- Standard    : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Checksum Generator, is a generator of pseudo casual ....
14 ----------------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author          Description
19 -- 2009-09-30  1.0      Tomas Alarcon   Created
20 ----------------------------------------------------------------------------
21 library ieee;
22 use ieee.std_logic_1164.all;
23
24 library work;
25 use work.Op_Code.all;
26
27 entity lsfr is
28
29   port (
30     Grst        : in  std_logic;        -- Global reset
31     resetLSFR   : in  std_logic;
32     getChecksum : in  std_logic;
33     checksum    : out std_logic_vector (LGOLDINDEX - 1 downto 0));
34
35 end lsfr;
36
37 architecture aramis of lsfr is
38
39   signal slfrStateY : std_logic_vector (LLSFR - 1 downto 0);
40
41 begin  -- aramis
42 checksum(LGOLDINDEX - 1 downto 11) <= slfrStateY;
43 checksum(10 downto 6) <= slfrStateY;
44 checksum(5 downto 1) <= slfrStateY;
45 checksum(0) <= '1';
46   lsfr_pro: process (getChecksum, Grst, resetLSFR)
47   begin  -- process lsfr_pro
48     if Grst = '0' or resetLSFR = '0' then                  -- asynchronous reset (
         active low)
49       slfrStateY <= FSTATELSFR;
50     elsif getChecksum'event and getChecksum = '0' then  -- rising clock edge
51       slfrStateY(4) <= slfrStateY(0) xor slfrStateY(1) xor slfrStateY(2) xor
           slfrStateY(4);
52       slfrStateY(3) <= slfrStateY(4);
53       slfrStateY(2) <= slfrStateY(3);
54       slfrStateY(1) <= slfrStateY(2);
55       slfrStateY(0) <= slfrStateY(1);
56     end if;
57   end process lsfr_pro;
58 end aramis;
```

Sorgente 17: Classe RebootCounter della FPGA.

```
1  ----------------------------------------------------------------------
2  -- Title      : RebootCounter
3  -- Project    : Aramis
4  ----------------------------------------------------------------------
5  -- File       : RebootCounter.1.0.1.vhd
6  -- Author     : Tomas A. Alarcon R.
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-09-05
9  -- Last update: 2009-10-26
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------
13 -- Description:  This element is in charge of counting the number of time that
14 -- was a CPU reboot
15 ----------------------------------------------------------------------
16 -- Copyright (c) 2009
17 ----------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version     Author          Description
20 -- 2009-09-05  1.0.1       Tomas Alarcon   Created
21 ----------------------------------------------------------------------
22
23 library ieee;
24 use ieee.std_logic_arith.all;
25 use ieee.std_logic_1164.all;
26 use ieee.std_logic_unsigned.all;
27 --use ieee.numeric_std.all;
28
29 library work;
30 use work.Op_Code.all;
31
32 entity RebootCounter is
33
34   port (
35     incrementRbCounter : in  std_logic;   -- Increment order
36     resetRebootCounter : in  std_logic;   -- Make to zero the reboot counter
37     clk                : in  std_logic;
38     Grst               : in  std_logic;   -- General reset of all system
39     resetCPU           : out std_logic;   -- Signal that requests the CPU reset
40     reloadFlash        : out std_logic);  -- Signal that requests the code reload
           on the memory flash
41
42 end RebootCounter;
43
44 architecture Aramis of RebootCounter is
45   attribute syn_preserve : boolean;
46 --  attribute syn_preserve of aramis: architecture is true;
47   signal activeTimer : std_logic;
48   attribute syn_preserve of activeTimer : signal is true;
49   signal timerFlag : std_logic;
50   signal resetCPUflag : std_logic;
51   signal rebootNumber : std_logic_vector (COUNTER-1 downto 0);  -- Signal that must
         have to registration of the times numbers that was did a CPU reset or a
        Flash reload
52   signal rebootNumberCopy : std_logic_vector (COUNTER-1 downto 0);
53   signal interruptTimer2 : std_logic;
54   signal timerRebootCounter : integer range 0 to TIMERLIMIT;
55   signal timer2RebootCounter : integer range 0 to TIMER2RCLIMIT + 1;
56 begin  -- Aramis
57
```

```
58  -- purpose: Counter from zero the times number that was did a CPU reboot or
59  -- reset and chooses between make a CPU reset or a Flash reload or Makes zero
60   -- the  reboot counter or Announce that reload the flash memory it's did or
61  -- Announce that Cpu reset it's did
62  -- type   : combinational
63  -- inputs : IncrementRbCounter, ResetRebootCounter
64  -- outputs: ResetCPU, ReloadFlash
65  Counter: process (incrementRbCounter, Grst, resetRebootCounter, interruptTimer2)
66  begin  -- process Counter
67    if Grst = '0' or resetRebootCounter = '0' or interruptTimer2 = '0' then
                          -- asynchronous reset (active low)
68      rebootNumber <= (others => '0');
69
70    elsif IncrementRbCounter'event and IncrementRbCounter = '0' then
71      RebootNumber <= RebootNumber + 1;
72    end if;
73  end process Counter;
74
75  -- purpose: Counter 10 ms
76  -- type   : sequential
77  -- inputs : clk, Grst
78  -- outputs:
79  process (clk, Grst, interruptTimer2, resetRebootCounter)
80
81  begin  -- process
82    if Grst = '0' or interruptTimer2 = '0' or resetRebootCounter = '0' then
                          -- asynchronous reset (active low)
83      timerRebootCounter <= 0;
84      timerFlag <= '0';
85      reloadFlash <= '1';
86      resetCPUflag <= '1';
87      resetCPU <= '1';
88      activeTimer <= '0';
89      rebootNumberCopy <= (others => '0');
90    elsif clk'event and clk = '1'   then  -- rising clock edge
91        if activeTimer = '1' then
92          if timerRebootCounter < TIMERLIMIT then
93            timerFlag <= '0';
94            timerRebootCounter <= timerRebootCounter + 1;
95          elsif timerRebootCounter = TIMERLIMIT  then
96            timerFlag <= '1';
97            timerRebootCounter <= 0;
98            activeTimer <= '0';
99          end if;
100       elsif rebootNumber > rebootNumberCopy then  -- this condition increase only
              once for the comparison of rebbotcounter
101         rebootNumberCopy <= rebootNumber;
102         if rebootNumber > x"2" then
103           reloadFlash <= '0';
104         elsif rebootNumber <= x"2" then
105           resetCPUflag <= '0';
106           resetCPU <= '0';
107           activeTimer <= '1';
108           timerFlag <= '0';
109           reloadFlash <= '1';
110         end if;
111       elsif resetCPUflag = '0' and timerFlag = '1' then
112         resetCPUflag <= '1';
113         resetCPU <= '1';
114         activeTimer <= '0';
115         reloadFlash <= '1';
116       end if;
```

```
117       end if;
118     end process;
119
120     -- purpose: Counter Tclk*TIMER2RCLIMIT seconds to erase the reboot counter
121     -- type    : sequential
122     -- inputs : clk, Grst
123     -- outputs: interruptTimer2
124     Timer2RC: process (clk, Grst)
125
126     begin  -- process Timer2RC
127       if Grst = '0' then                    -- asynchronous reset (active low)
128         timer2RebootCounter<= 0;
129         interruptTimer2 <= '1';
130       elsif clk'event and clk = '1' then  -- rising clock edge
131         if timer2RebootCounter <= TIMER2RCLIMIT  then
132           timer2RebootCounter <= timer2RebootCounter + 1;
133           interruptTimer2 <= '1';
134         else
135           timer2RebootCounter <= 0;
136           interruptTimer2 <= '0';
137         end if;
138       end if;
139     end process Timer2RC;
140
141   end Aramis;
```

Sorgente 18: Classe ProgramControl della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title       : ProgramControl
3  -- Project     : Aramis
4  ----------------------------------------------------------------------------
5  -- File        : ProgramControl.vhd
6  -- Author      : Tomas Alarcon
7  -- Company     : Politecnico di TOrino
8  -- Created     : 2009-05-18
9  -- Last update: 2009-11-02
10 -- Platform    : Actel
11 -- Standard    : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Read the default memory content
14 ----------------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author  Description
19 -- 2009-05-18  1.0      TAlarcon     Created
20 ----------------------------------------------------------------------------
21 library ieee;
22 use ieee.std_logic_arith.all;
23 use ieee.std_logic_1164.all;
24 use ieee.std_logic_signed.all;
25 library work;
26 use work.Op_Code.all;
27
28 entity ProgramControl is
29
30   port (
31     reloadFlash                 : in  std_logic;  -- Requests reload the memory
       flash wiht the correct information
32     Grst                        : in  std_logic;
33     clk                         : in  std_logic;
34     passwordInfo                : in  std_logic_vector (TWOBITS - 1 downto 0);
35     writeMemoryDone             : in  std_logic;
36     sendPassword                : out std_logic;  -- Requests the send to and check
         the password with the CPU
37     readIndex                   : out std_logic;
38     writeAllMemory              : out std_logic;
39     writePartMemory             : out std_logic;
40     boutonAddressMemory         : out std_logic_vector (WORDCPU - 1 downto 0);
41     writeFlash                  : out std_logic;
42     resetRebootCounter          : out std_logic;
43     resetIndexCounter           : out std_logic;
44     enableWDT                   : out std_logic;
45     index                       : in  integer range 0 to NUMBLOCKS);  -- Index
         Number
46
47 end ProgramControl;
48
49 architecture Aramis of ProgramControl is
50
51   signal requestIndexFlag : std_logic;  -- To see if there was a requestIndex
52   signal sendPasswordC : std_logic;     -- To know when reading the password
53
54 begin  -- Aramis
55
56   -- purpose: Responds to the reload memory request
57   -- type    : combinational
```

```
58  -- inputs : clk, Grst, ReloadFlash
59  -- outputs: SendPassword, RequestIndex
60  ReloadRequests: process (Grst, reloadFlash, writeMemoryDone)
61  begin  -- process ReloadRequests
62    if Grst = '0' or  writeMemoryDone = '0' then                    -- asynchronous
          reset (active low)
63      sendPassword <= '1';
64      sendPasswordC <= '1';
65    elsif reloadFlash'event and reloadFlash = '0' then  -- Downfall ReloadFlash
          edge
66      sendPassword <= '0';
67      sendPasswordC <= '0';
68    end if;
69  end process ReloadRequests;
70
71  -- purpose: takes desitions about the password info
72  -- type   : sequential
73  -- inputs : passwordInfo
74  -- outputs: writeAllMemory, writePartMemory, writeFlash, requestIndexFlag
75  Password_Info: process (clk, Grst, writeMemoryDone)
76  begin  -- process Password_Info
77    if Grst = '0' or writeMemoryDone = '0'  then
78      resetRebootCounter <= '1';
79      resetIndexCounter <= '1';
80      writeFlash <= '1';
81      writePartMemory <= '1';
82      writeAllMemory <= '1';
83      readIndex <= '1';
84      requestIndexFlag <= '1';          -- Same value of readIndex
85      enableWDT <= '1';
86    elsif rising_edge(clk) then
87      if passwordInfo = "01" and sendPasswordC = '0' then  -- Password is good
88        readIndex <= '0';
89        requestIndexFlag <= '0';
90        writePartMemory <= '0';
91        writeFlash <= '0';
92        resetRebootCounter <= '0';
93        resetIndexCounter <= '1';
94        writeAllMemory <= '1';
95        enableWDT <= '0';
96      elsif passwordInfo = "10" and sendPasswordC = '0'  then  -- Password is
            wtrong
97        writeAllMemory <= '0';
98        writeFlash <= '0';
99        resetRebootCounter <= '0';
100       resetIndexCounter <= '0';
101       readIndex <= '1';
102       requestIndexFlag <= '1';          -- Same value of readIndex
103       writePartMemory <= '1';
104       enableWDT <= '0';
105     elsif (passwordInfo = "11" or passwordInfo = "00") and sendPasswordC = '0'
          then
106       requestIndexFlag <= '1';          -- Same value of readIndex
107       resetRebootCounter <= '1';
108       resetIndexCounter <= '1';
109       writeFlash <= '1';
110       writePartMemory <= '1';
111       writeAllMemory <= '1';
112       readIndex <= '1';
113       enableWDT <= '1';
114     end if;
115   end if;
```

```
116   end process Password_Info;
117
118   -- purpose: Calculates the starting address to recharge a piece of memory
119   -- type    : sequential
120   -- inputs : index
121   -- outputs: boutonAddressMemory, writePartMemory, WriteFlash
122   PartMemory_Calcolus: process (clk, Grst)
123     variable addressNum : integer range 0 to TOP64KB-1;
124     variable indexNum : integer range 0 to NUMBLOCKS;
125   begin  -- process PartMemory_Calcolus
126     if Grst = '0' then
127       boutonAddressMemory <= (others => '0');
128       addressNum := 0;
129       indexNum := 0;
130     elsif rising_edge(clk) then
131       if requestIndexFlag = '0'  then
132         indexNum := conv_integer(index);
133         addressNum:= indexNum*NBYTEPERBLOCK + FIRSTADDMEM;
134         boutonAddressMemory <= conv_std_logic_vector(integer(addressNum),WORDCPU);
135       end if;
136     end if;
137   end process PartMemory_Calcolus;
138
139
140 end Aramis;
```

Sorgente 19: Classe ProgramInterface della FPGA.

```
1  ----------------------------------------------------------------------
2  -- Title      : ProgramInterface.1.0.2
3  -- Project    : Aramis
4  ----------------------------------------------------------------------
5  -- File       : Programinterface.1.0.2.vhd
6  -- Author     : Tomas Antonio Alarcon Rivera
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-08-22
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------
13 -- Description: Class was responsible for reprogramming the memory of the CPU
14 ----------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author          Description
19 -- 2009-08-22  1.0.2    Tomas Alarcon   Created
20 ----------------------------------------------------------------------
21
22 library ieee;
23 use ieee.std_logic_1164.all;
24 use ieee.std_logic_arith.all;
25 use ieee.std_logic_signed.all;
26
27 library ieee;
28 library proasic3;
29 use proasic3.all;
30
31 library work;
32 use work.Op_Code.all;
33
34 entity ProgramInterface is
35    port (
36     Grst                : in  std_logic;
37     clk                 : in  std_logic;
38     uartClk             : in  std_logic;
39     writeFlash          : in  std_logic;
40     sendPassword        : in  std_logic;
41     resetCPU            : in  std_logic;  -- Active low
42     rx                  : in  std_logic;
43     writeAllMemory      : in  std_logic;
44     writePartMemory     : in  std_logic;
45     boutonAddressProm   : in  std_logic_vector (WORDCPU -1 downto 0);
46     stateMemoryProm     : in  std_logic_vector (BYTE -1 downto 0);
47     dataReady           : in  std_logic;
48
49     writeMemoryDone     : out std_logic;
50     address             : out integer range 0 to MEMSIZE;
51     getState            : out std_logic;
52     PasswordInfo        : out std_logic_vector (TWOBITS - 1 downto 0);
53     RstCpu              : out std_logic;
54     TckCpu              : out std_logic;
55     tx                  : out std_logic;
56     sendStateWatchdog   : out std_logic;
57     stateWatchdog       : out std_logic_vector(WORDCPU -1 downto 0));
58
59 end ProgramInterface;
60
```

```
61
62 architecture Aramis of ProgramInterface is
63   attribute syn_preserve : boolean;
64
65   component uart
66
67     generic (
68       BAUDRATEDIV : integer range 0 to 65535 := 1250);  -- Baud rate divisor
69                                                          -- f(uartClk=48MHz)
                                                               /781/4=~9600 Br
70     port (
71       clk         : in  std_logic;         -- Clock of general system
72       Grst        : in  std_logic;         -- Global reset
73       txDataL     : in  std_logic;         -- Async load transmit buffer
74       rxDataR     : in  std_logic;         -- Async read receive buffer
75       uartClk     : in  std_logic;         -- Clock used for Transmit/Receive
76       rx          : in  std_logic;         -- Rx
77       dataTxIn    : in  std_logic_vector (BYTE - 1 downto 0);  -- Data to transmit
78       interruptTx : out std_logic;         -- Transmit interrupt: indicate waiting
             for byte
79       interruptRx : out std_logic;         -- Recive interrupt: indicate Byte
             received
80       tx          : out std_logic;         -- TX
81       rxDataOut   : out std_logic_vector (BYTE -1 downto 0));  -- Byte received
82
83   end component;
84   type Mem_Type is array (LENGTHBLOCKDATA -1 downto 0) of std_logic_vector (BYTE -
         1 downto 0);
85
86   signal  bslSyncCount          : integer range 0 to 7;  -- BSL protocol control
         activer
87   attribute syn_preserve of bslSyncCount : signal is true;
88   signal  bslSyncCount2         : integer range 0 to 4;
89   attribute syn_preserve of bslSyncCount2 : signal is true;
90   signal  sendDataCount         : integer range 0 to LENGTHBLOCKDATA;  -- blocData
         index
91   attribute syn_preserve of sendDataCount : signal is true;
92   signal  toErase               : std_logic;
93   attribute syn_preserve of toErase : signal is true;
94   signal  toSend                : std_logic;           -- Flag to indicate than
         the process in that moment is sending data or not
95   attribute syn_preserve of toSend : signal is true;
96   signal  loadTxDataS           : std_logic;
97   signal  txDataS               : std_logic_vector (HALFWORD -1 downto 0);
98   signal  uLoadRxDataS          : std_logic;
99   signal  setupFlag             : std_logic;
100  attribute syn_preserve of setupFlag : signal is true;
101  signal  takeControlBsl        : std_logic;
102  signal  dataSent              : std_logic;
103  attribute syn_preserve of dataSent : signal is true;
104  signal  interruptTxS          : std_logic;
105  signal  syncDone              : std_logic;
106  attribute syn_preserve of syncDone : signal is true;
107  signal  interruptRxS          : std_logic;
108  attribute syn_preserve of interruptRxS : signal is true;
109  signal  interruptRxS2         : std_logic;
110  attribute syn_preserve of interruptRxS2 : signal is true;
111  signal  rxDataS               : std_logic_vector (HALFWORD -1 downto 0);
112  signal  Acknowledge           : std_logic_vector (HALFWORD -1 downto 0);
113  attribute syn_preserve of Acknowledge : signal is true;
114  signal  readAllMemoryCounter  : integer range NUMBLOCKS + 1 downto 0;
115  signal  readBlockMemory       : integer range NBYTEPERBLOCK + 2 downto 0;
```

```
116   signal  dataBegin              : integer range 0 to 10;
117   signal  boutonAddressPromN     : integer range 0 to TOP64KB;
118   signal  enableTimer1           : std_logic;
119   signal  intTimer1              : std_logic;
120   signal  writeMemoryDoneS       : std_logic;
121   signal  reloadSingelBlockCo    : std_logic_vector (BYTE - 1 downto 0) := (others
        => '0');
122   signal  reloadAllBlocksCo      : std_logic_vector (BYTE - 1 downto 0) := (others
        => '0');
123   signal  send2Password          : std_logic;
124   attribute syn_preserve of send2Password : signal is true;
125   signal  limitSendDataCount     : integer range 0 to NUMBLOCKS + 1;  -- this signal
          will be the limit for the send data
126   signal  setupBegin             : std_logic_vector (WORDCPU -1 downto 0);
127   signal  blockData              : Mem_Type;
128   attribute syn_preserve of blockData :signal is true;
129   signal  writeMemoryDoneErase   : integer range 0 to 2;
130   signal  convAddress            : std_logic;
131   signal  evenodd                : std_logic;
132   signal  ckl                    : std_logic_vector (BYTE -1 downto 0);
133   signal  ckh                    : std_logic_vector (BYTE -1 downto 0);
134
135
136
137
138 begin  -- Aramis
139
140   UA1:uart
141     port map (
142       clk          => clk,
143       Grst         => Grst,
144       txDataL      => loadTxDataS,
145       rxDataR      => uLoadRxDataS,
146       uartClk      => uartClk,
147       rx           => rx,
148       dataTxIn     => txDataS,
149       interruptTx => interruptTxS,
150       interruptRx => interruptRxS,
151       tx           => tx,
152       rxDataOut    => rxDataS);
153
154   -- purpose: Send the security password of th CPU BSL protocol
155   -- type    : sequential
156   -- inputs : clk, Grst, SendPassword, Acknomledge
157   -- outputs: TX, RstCpu,TckCpu
158   Programming: process (clk, Grst)
159
160   begin  -- process Programming
161     if Grst = '0' then                  -- asynchronous reset (active low)
162       PasswordInfo <= "11";
163       RstCpu <= '1';
164       bslSyncCount <= 0;
165       txDataS <= x"00";
166       loadTxDataS <= '0';
167       uLoadRxDataS <= '0';
168       Acknowledge <= x"00";
169       sendDataCount <= 0;
170       setupFlag <= '0';
171       syncDone<= '0';
172       toSend <= '0';
173       dataSent <= '0';
174       setupBegin <= (others => '0');
```

```
        boutonAddressPromN <= 0;
        writeMemoryDone <= '1';
        writeMemoryDoneS <= '1';
        toErase <= '0';
        enableTimer1 <= '0';
        writeMemoryDoneErase <= 0;
        send2Password <= '0';
        sendStateWatchdog <= '1';
        stateWatchdog<= (others => '0');
        reloadAllBlocksCo <= (others => '0');
        reloadSingelBlockCo <= (others => '0');
        blockData <= (x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"
            00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00",x"00");
        readAllMemoryCounter <= 0;
        readBlockMemory <= 0;
        bslSyncCount2 <= 0;
        limitSendDataCount <= 0;
        dataBegin <= 0;
        TckCpu <= '1';
        takeControlBsl <= '0';
        getState <= '1';
        address <= 0;
        convAddress <= '0';
        evenodd <= '0';
        ckl <= (others => '0');
        ckh <= (others => '0');

      elsif clk'event and clk = '1' then  -- rising clock edge

--------------------------------------------------------------------------------
-- Reset CPU Process
--------------------------------------------------------------------------------
        if resetCPU = '0' then
          RstCpu <= '0';
        elsif resetCPU = '1' then
          RstCpu <= '1';
        end if;
--------------------------------------------------------------------------------
-- Resets the signal after a memory write well, with a delay of 2 clock periods
--------------------------------------------------------------------------------

        if writeMemoryDoneS = '0' and writeMemoryDoneErase = 2 then
```

```
216           writeMemoryDone <= '1';
217           writeMemoryDoneS <= '1';
218           sendStateWatchdog <= '1';
219           writeMemoryDoneErase <= 0;
220         elsif writeMemoryDoneS = '0' and writeMemoryDoneErase <2 then
221           writeMemoryDoneErase <= writeMemoryDoneErase + 1;
222         end if;
223 ------------------------------------------------------------------------------
224 --Setup the limit of send data counter and the bata buffer to send to the
225 --BSL unit
226 ------------------------------------------------------------------------------
227
228         if sendPassword = '0' and setupFlag = '0' then  --1080
                ffffffffffffffffffffffffffffffffffffffffffffffff0000000024241080 <--LB
229           blockData(0) <= x"80";
230           blockData(1) <= x"10";
231           blockData(2) <= x"24";
232           blockData(3) <= x"24";
233           blockData(4) <= x"00";
234           blockData(5) <= x"00";
235           blockData(6) <= x"00";
236           blockData(7) <= x"00";
237           blockData(8) <= x"ff";
238           blockData(9) <= x"ff";
239           blockData(10) <= x"ff";
240           blockData(11) <= x"ff";
241           blockData(12) <= x"ff";
242           blockData(13) <= x"ff";
243           blockData(14) <= x"ff";
244           blockData(15) <= x"ff";
245           blockData(16) <= x"ff";
246           blockData(17) <= x"ff";
247           blockData(18) <= x"ff";
248           blockData(19) <= x"ff";
249           blockData(20) <= x"ff";
250           blockData(21) <= x"ff";
251           blockData(22) <= x"ff";
252           blockData(23) <= x"ff";
253           blockData(24) <= x"ff";
254           blockData(25) <= x"ff";
255           blockData(26) <= x"ff";
256           blockData(27) <= x"ff";
257           blockData(28) <= x"ff";
258           limitSendDataCount <= SENDPASSWORDCOUNT;  --31
259           dataBegin <= 0;
260           setupFlag <= '1';
261           toSend <= '1';
262         end if;
263 ------------------------------------------------------------------------------
264 ------------------------------------------------------------------------------
265
266         if writeAllMemory = '0' and setupFlag = '0' then
267           if  toErase = '0'  then --1080a506000004041880
268             blockData(0) <= x"80";
269             blockData(1) <= x"18";
270             blockData(2) <= x"04";
271             blockData(3) <= x"04";
272             blockData(4) <= x"00";
273             blockData(5) <= x"00";
274             blockData(6) <= x"06";
275             blockData(7) <= x"a5";
276             limitSendDataCount <= ERASEMASSMEMCOUNT;  --11
```

```
277           dataBegin <= LENGTHERASEMASSMEM;
278           setupFlag <= '1';
279           toSend <= '1';
280         else
281           if send2Password = '0' then  --1080
                ffffffffffffffffffffffffffffffffffffffffffffffff0000000024241080 <--LB
282             blockData(0) <= x"80";
283             blockData(1) <= x"10";
284             blockData(2) <= x"24";
285             blockData(3) <= x"24";
286             blockData(4) <= x"00";
287             blockData(5) <= x"00";
288             blockData(6) <= x"00";
289             blockData(7) <= x"00";
290             blockData(8) <= x"ff";
291             blockData(9) <= x"ff";
292             blockData(10) <= x"ff";
293             blockData(11) <= x"ff";
294             blockData(12) <= x"ff";
295             blockData(13) <= x"ff";
296             blockData(14) <= x"ff";
297             blockData(15) <= x"ff";
298             blockData(16) <= x"ff";
299             blockData(17) <= x"ff";
300             blockData(18) <= x"ff";
301             blockData(19) <= x"ff";
302             blockData(20) <= x"ff";
303             blockData(21) <= x"ff";
304             blockData(22) <= x"ff";
305             blockData(23) <= x"ff";
306             blockData(24) <= x"ff";
307             blockData(25) <= x"ff";
308             blockData(26) <= x"ff";
309             blockData(27) <= x"ff";
310             blockData(28) <= x"ff";
311             limitSendDataCount <= SENDPASSWORDCOUNT;  --31
312             dataBegin <= 0;
313             send2Password <= '1';
314             setupFlag <= '1';
315           else
316             blockData(0) <= x"80"; --"00fa0000fefe1080
317             blockData(1) <= x"10";
318             blockData(2) <= x"fe";
319             blockData(3) <= x"fe";
320             blockData(4) <= x"00";
321             blockData(5) <= x"00";
322             blockData(6) <= x"fa";
323             blockData(7) <= x"00";
324             limitSendDataCount <= SENDBLOCKDATACOUNT;  --261
325             dataBegin <= LENGTHRXBLKDAT_HDR;
326             setupFlag <='1';
327           end if;
328         end if;
329       end if;
330 ------------------------------------------------------------------------------
331 ------------------------------------------------------------------------------
332
333       if writePartMemory = '0' and setupFlag = '0' then
334         blockData(0) <= x"80";
335         blockData(1) <= x"10";
336         blockData(2) <= x"fe";
337         blockData(3) <= x"fe";
```

```
338           blockData(4) <= x"00";
339           blockData(5) <= x"00";
340           blockData(6) <= x"fa";
341           blockData(7) <= x"00";
342           limitSendDataCount <= SENDBLOCKDATACOUNT;  --261
343           dataBegin <= LENGTHRXBLKDAT_HDR;
344           setupFlag <='1';
345         end if;
346 ------------------------------------------------------------------------------
347 -- Access protocol for the BSL
348 ------------------------------------------------------------------------------
349         if (sendPassword = '0' or writeFlash = '0') and takeControlBsl = '0' then
350 
351           case bslSyncCount is
352             when 0 => RstCpu <= '0';
353                       TckCpu <= '1';
354                       if intTimer1 = '0' then
355                         enableTimer1 <= '1';
356                       elsif intTimer1 = '1' then
357                         bslSyncCount <= bslSyncCount + 1;
358                         enableTimer1 <= '0';
359                       end if;
360             when 1 => TckCpu <= '0';
361                       RstCpu <= '0';
362                       if intTimer1 = '0' then
363                         enableTimer1 <= '1';
364                       elsif intTimer1 = '1' then
365                         bslSyncCount <= bslSyncCount + 1;
366                         enableTimer1 <= '0';
367                       end if;
368             when 2 => TckCpu <= '1';
369                       RstCpu <= '0';
370                       if intTimer1 = '0' then
371                         enableTimer1 <= '1';
372                       elsif intTimer1 = '1' then
373                         bslSyncCount <= bslSyncCount + 1;
374                         enableTimer1 <= '0';
375                       end if;
376             when 3 => TckCpu <= '1';
377                       RstCpu <= '0';
378                       if intTimer1 = '0' then
379                         enableTimer1 <= '1';
380                       elsif intTimer1 = '1' then
381                         bslSyncCount <= bslSyncCount + 1;
382                         enableTimer1 <= '0';
383                       end if;
384             when 4 => TckCpu <= '0';
385                       RstCpu <= '0';
386                       if intTimer1 = '0' then
387                         enableTimer1 <= '1';
388                       elsif intTimer1 = '1' then
389                         bslSyncCount <= bslSyncCount + 1;
390                         enableTimer1 <= '0';
391                       end if;
392             when 5 => RstCpu <= '1';
393                       TckCpu <= '0';
394                       if intTimer1 = '0' then
395                         enableTimer1 <= '1';
396                       elsif intTimer1 = '1' then
397                         bslSyncCount <= bslSyncCount + 1;
398                         enableTimer1 <= '0';
399                       end if;
```

```
400           when 6 => TckCpu <= '1';
401                     RstCpu <= '1';
402                     if intTimer1 = '0' then
403                       enableTimer1 <= '1';
404                     elsif intTimer1 = '1' then
405                       bslSyncCount <= bslSyncCount + 1;
406                       takeControlBsl <= '1';
407                       enableTimer1 <= '0';
408                     end if;
409           when 7 => null;
410         end case;
411       end if;
412 ------------------------------------------------------------------------
413 -- Reading process memory EEPROM** or F-RAM**
414 ------------------------------------------------------------------------
415       if writeAllMemory = '0' and toSend = '0' and send2Password = '1' and toErase
              = '1' then
416
417         setupBegin <= conv_std_logic_vector(integer(readAllMemoryCounter*
                NBYTEPERBLOCK + FIRSTADDMEM),LENGTHPROM);
418         blockData(4) <= setupBegin(BYTE - 1 downto 0);
419         blockData(5) <= setupBegin(LENGTHPROM - 1 downto BYTE);
420         if readBlockMemory = NBYTEPERBLOCK then
421           toSend <= '1';
422           readBlockMemory <= 0;
423         else
424           address <= readAllMemoryCounter*NBYTEPERBLOCK+readBlockMemory;
425           getState <= '0';
426           if dataReady = '1' then
427             blockData(dataBegin + readBlockMemory)<= stateMemoryProm;
428             readBlockMemory <= readBlockMemory + 1;
429             getState <= '1';
430           end if;
431         end if;
432       end if;
433 ------------------------------------------------------------------------
434 ------------------------------------------------------------------------
435       if writePartMemory = '0' and toSend = '0'  then
436         blockData(4) <= boutonAddressProm(BYTE - 1 downto 0);
437         blockData(5) <= boutonAddressProm(LENGTHPROM - 1 downto BYTE);
438         if readBlockMemory = NBYTEPERBLOCK then
439           toSend <= '1';
440           readBlockMemory <= 0;
441         else
442           boutonAddressPromN <= conv_integer(std_logic_vector(boutonAddressProm));
443           convAddress <= '1';
444           if convAddress = '1' then
445             address <= boutonAddressPromN-FIRSTADDMEM+readBlockMemory;
446             getState <= '0';
447             if dataReady = '1' then
448               blockData(readBlockMemory + dataBegin)<= stateMemoryProm;
449               readBlockMemory <= readBlockMemory + 1;
450               convAddress <= '0';
451               getState <= '1';
452             end if;
453           end if;
454         end if;
455       end if;
456 ------------------------------------------------------------------------
457 -- Protocol for sending the data through the UART
458 ------------------------------------------------------------------------
459       ------------------------------------------------------------------
```

```
460       -- Synchronization sequence, before each command
461       ------------------------------------------------------------------------
462       if takeControlBsl = '1' and setupFlag = '1' and dataSent = '0' and toSend =
              '1' then
463         case bslSyncCount2 is
464           when 0 =>
465             if interruptTxS = '1' then
466               txDataS <= x"80";
467               loadTxDataS <= '1';
468             else
469               bslSyncCount2 <= 1;
470               loadTxDataS <= '0';
471             end if;
472           when 1 =>
473             if interruptRxS2 = '1' then
474               uLoadRxDataS <= '1';
475               Acknowledge <= rxDataS;
476
477               bslSyncCount2 <= 2;
478             else
479               uLoadRxDataS <= '0';
480             end if;
481           when 2 =>
482             case Acknowledge is
483               when x"90" =>
484                 Acknowledge <= x"00";
485                 bslSyncCount2 <= 3;
486                 syncDone <= '1';
487               when x"A0" =>
488                 Acknowledge <= x"00";
489                 bslSyncCount2 <= 0;
490                 syncDone<= '0';
491               when others => null;
492             end case;
493           when others => null;
494         end case;
495       end if;
496       ------------------------------------------------------------------------
497       -- Sending comands
498       ------------------------------------------------------------------------
499       if syncDone = '1' then
500         if toSend = '1' then
501           if sendDataCount = limitSendDataCount then
502             sendDataCount <= 0;
503             dataSent <= '1';
504             syncDone <= '0';
505           else
506             if interruptTxS = '1' then
507               loadTxDataS <= '1';
508               case evenodd is
509                 when '0' =>
510                   ckl <= not(ckl xor blockData(sendDataCount));
511                   evenodd <= '1';
512                 when '1' =>
513                   ckh <= not(ckh xor blockData(sendDataCount));
514                   evenodd <= '0';
515               end case;
516               if sendDataCount = limitSendDataCount - 2 then
517                 sendDataCount <= sendDataCount + 1;
518                 txDataS <= ckl;
519               elsif sendDataCount = limitSendDataCount - 1 then
520                 sendDataCount <= sendDataCount + 1;
```

```
521                 txDataS <= ckh;
522               else
523                 sendDataCount <= sendDataCount + 1;
524                 txDataS <= blockData(sendDataCount);
525               end if;
526             else
527               loadTxDataS <= '0';
528             end if;
529           end if;
530         end if;
531       end if;
532       ------------------------------------------------------------------------
533       -- Receiving BSL response unit, size of decision for each case
534       ------------------------------------------------------------------------
535       if dataSent = '1' then
536         if toSend = '1' then
537           ------------------------------------------------------------------------
538           ------------------------------------------------------------------------
539           if sendPassword= '0' then
540             if interruptRxS2 = '1' then
541               uLoadRxDataS <= '1';
542               Acknowledge <= rxDataS;
543             else
544               uLoadRxDataS <= '0';
545             end if;
546             case Acknowledge is
547               when x"90" =>
548                 Acknowledge <= x"00";
549                 PasswordInfo <= "01";
550                 sendDataCount <= 0;
551                 setupFlag <= '0';
552                 dataSent <= '0';
553                 toSend <= '0';
554               when x"A0" =>
555                 Acknowledge <= x"00";
556                 PasswordInfo <= "10";
557                 sendDataCount <= 0;
558                 dataSent <= '0';
559                 setupFlag <= '0';
560                 toSend <= '0';
561               when others => null;
562             end case;
563             ------------------------------------------------------------------------
564             ------------------------------------------------------------------------
565           elsif writeFlash = '0'   then
566             if interruptRxS2 = '1' then
567               uLoadRxDataS <= '1';
568               Acknowledge <= rxDataS;
569             else
570               uLoadRxDataS <= '0';
571             end if;
572             case Acknowledge is
573               when x"90" =>
574                 Acknowledge <= x"00";
575                 txDataS <= x"00";
576                 sendDataCount <= 0;
577                 toSend <= '0';
578                 dataSent <= '0';
579                 if writeAllMemory ='0' then
580                   if toErase = '0' then
581                     toErase <= '1';
582                     setupFlag <= '0';
```

```
583                   else
584                     if send2Password = '0' then
585                       send2Password <= '1';
586                       setupFlag <= '0';
587                     else
588                       if readAllMemoryCounter = limitSendDataCount then --
589                         readAllMemoryCounter <= 0;
590                         RstCpu <= '1';
591                         writeMemoryDone <= '0';
592                         writeMemoryDoneS <= '0';
593                         reloadAllBlocksCo <= reloadAllBlocksCo + 1;
594                         sendStateWatchdog <= '0';
595                         stateWatchdog(WORDCPU-1 downto HALFWORD)<=
                                reloadAllBlocksCo;
596                         send2Password <= '0';
597                         setupFlag <= '0';
598                         toErase <= '0';
599                         takeControlBsl <= '0';
600                       else
601                         readAllMemoryCounter <= readAllMemoryCounter + 1;
602                       end if;
603                     end if;
604                   end if;
605 ------------------------------------------------------------------------------
606 ------------------------------------------------------------------------------
607                 elsif writePartMemory ='0' then
608 
609                   writeMemoryDone <= '0';
610                   writeMemoryDoneS <= '0';
611                   reloadSingelBlockCo <= reloadSingelBlockCo + 1;
612                   sendStateWatchdog <= '0';
613                   stateWatchdog(HALFWORD-1 downto 0)<= reloadSingelBlockCo;
614                   takeControlBsl <= '0';
615                   setupFlag <= '0';
616                 end if;
617               when x"A0" =>
618                 Acknowledge <= x"00";
619                 setupFlag <= '0';
620                 if writeAllMemory = '0' then
621                   readAllMemoryCounter <= readAllMemoryCounter - 1;
622                   -- this is the comand to reset the sending of the blockmemory
                          after a sending failed
623                 end if;
624                 toSend <= '0';
625                 dataSent <= '0';
626               when others => null;
627             end case;
628           end if;
629         end if;
630       end if;
631     end if;
632   end process Programming;
633 ------------------------------------------------------------------------------
634 ------------------------------------------------------------------------------
635 
636 -- purpose: generate signal for read data in just time
637 -- type   : sequential
638 -- inputs : interruptRxS, Grst, uLoadRxDataS
639 -- outputs: interruptRxS2
640   Inte: process (interruptRxS, Grst, uLoadRxDataS)
641   begin  -- process Inte
```

```
642     if Grst = '0' or uLoadRxDataS = '1' then                    -- asynchronous
            reset (active low)
643       interruptRxS2 <= '0';
644     elsif interruptRxS'event and interruptRxS = '1' then  -- rising clock edge
645       interruptRxS2<= '1';
646     end if;
647   end process Inte;
648
649 -- purpose: count a time determinate by limitTimer1
650 -- type   : sequential
651 -- inputs : clk, Grst, limitTimer1, enable timer
652 -- outputs: intTimer1
653   timer_1: process (clk, Grst, enableTimer1)
654     variable countTimer1 : integer range 0 to BSLDELAY;  -- Timer limit
655   begin  -- process timer_1
656     if Grst = '0' or enableTimer1 = '0' then                    -- asynchronous
            reset (active low)
657       intTimer1 <= '0';
658       countTimer1:= BSLDELAY;--limitTimer1;
659     elsif clk'event and clk = '1' then    -- rising clock edge
660       if enableTimer1 = '1' then
661         if countTimer1 = 1 then
662           intTimer1 <= '1';
663           countTimer1:= 0;
664         elsif countTimer1 = 0 then
665           null;
666         else
667           countTimer1:= countTimer1 -1;
668         end if;
669       end if;
670     end if;
671   end process timer_1;
672 end Aramis;
```

Sorgente 20: Classe UART della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : uart
3  -- Project    :
4  ----------------------------------------------------------------------------
5  -- File       : uart.1.0.1.vhd
6  -- Author     : Tomas  <Tom@MacBook-Pro-de-Tomas.local>
7  -- Company    :
8  -- Created    : 2009-09-14
9  -- Last update: 2009-11-02
10 -- Platform   :
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: Uart based on Uart project of Philippe Carton (philippe.
       carton2@libertysurf.fr)
14 ----------------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author  Description
19 -- 2009-09-14  1.0      Tom Created
20 ----------------------------------------------------------------------------
21 library ieee;
22 use ieee.std_logic_1164.all;
23
24 library work;
25 use work.Op_Code.all;
26
27 entity uart is
28
29   generic (
30     BAUDRATEDIV : integer range 0 to 65535 := 260);  -- Baud rate divisor
31
32   port (
33     clk         : in  std_logic;        -- Clock of general system
34     Grst        : in  std_logic;        -- Global reset
35     txDataL     : in  std_logic;        -- Async load transmit buffer
36     rxDataR     : in  std_logic;        -- Async read receive buffer
37     uartClk     : in  std_logic;        -- Clock used for Transmit/Receive
38     rx          : in  std_logic;        -- Rx
39     dataTxIn    : in  std_logic_vector (BYTE - 1 downto 0);  -- Data to transmit
40     interruptTx : out std_logic;  -- Transmit interrupt: indicate waiting for byte
41     interruptRx : out std_logic;  -- Recive interrupt: indicate Byte received
42     tx          : out std_logic;        -- TX
43     rxDataOut   : out std_logic_vector (BYTE -1 downto 0));  -- Byte received
44
45 end uart;
46
47
48 architecture Aramis of uart is
49
50   component clockdivider
51     generic (
52       COUNT : integer range 0 to 65535);
53     port (
54       clk         : in  std_logic;        -- Clock
55       Grst        : in  std_logic;        -- Global reset
56       countEnable : in  std_logic;        -- Counter Enable
57       overFlow    : out std_logic);       -- Output
58   end component;
59
```

```
60   component txunit
61     port (
62       uartClk     : in  std_logic;          -- Clock signal
63       Grst        : in  std_logic;          -- Global reset
64       enableTx    : in  std_logic;          -- Enable unit signal
65       loadData    : in  std_logic;          -- Asynchronous load
66       dataIn      : in  std_logic_vector (BYTE -1 downto 0);  -- Byte to transmit
67       tx          : out std_logic;          -- Data output
68       txUnitBusy  : out std_logic);         -- Tx busy
69   end component;
70
71   component rxunit
72     port (
73       uartClk       : in  std_logic;        -- System clock signal
74       Grst          : in  std_logic;        -- Global reset
75       rxEnable      : in  std_logic;        -- Enable input
76       readData      : in  std_logic;        -- Async read received byte
77       rx            : in  std_logic;        -- Data input
78       dataAvailable : out std_logic;        -- Byte available
79       dataOut       : out std_logic_vector (BYTE -1 downto 0));  -- Byte received
80   end component;
81
82   signal rxData : std_logic_vector (BYTE -1 downto 0);  -- Last byte received
83   signal txUnitBusy : std_logic;          -- Transmiter busy
84   signal rxEnable : std_logic;            -- Enable rx unit
85   signal dataAvailable : std_logic;       -- Data recived and available
86   signal txEnable : std_logic;            -- Enable tx unit
87   signal readData : std_logic;            -- Async read reciver buffer
88   signal gnd : std_logic;                 -- gnd signal
89   signal vcc : std_logic;                 -- vcc signal
90
91
92 begin  -- Aramis
93   gnd <= '0';
94   vcc <= '1';
95   rxrate:clockdivider                     -- baud rate adjust
96     generic map (
97       COUNT => BAUDRATEDIV)
98     port map (
99       clk         => uartClk,
100      Grst        => Grst,
101      countEnable => vcc,
102      overFlow    => rxEnable);
103  txrate:clockdivider                     -- 4 divider for tx
104    generic map (
105      COUNT => 4)
106    port map (
107      clk         => uartClk,
108      Grst        => Grst,
109      countEnable => rxEnable,
110      overFlow    => txEnable);
111  Utxunit:txunit
112    port map (
113      uartClk    => uartClk,
114      Grst       => Grst,
115      enableTx   => txEnable,
116      loadData   => txDataL,
117      dataIn     => dataTxIn,
118      tx         => tx,
119      txUnitBusy => txUnitBusy);
120  Urxunit:rxunit
121    port map (
```

```
122       uartClk       => uartClk,
123       Grst          => Grst,
124       rxEnable      => rxEnable,
125       readData      => readData,
126       rx            => rx,
127       dataAvailable => dataAvailable,
128       dataOut       => rxData);
129
130   interruptTx <= not txUnitBusy;
131   interruptRx <= dataAvailable;
132   rxDataOut <= rxData;
133
134   -- purpose: implement the logical for to made the uart
135   -- type   : sequential
136   -- inputs : clk, Grst, txDataL
137   -- outputs: dataOut, tx
138   uart_proc: process (clk, Grst)
139   begin  -- process uart_proc
140     if Grst = '0' then                  -- asynchronous reset (active low)
141       readData <= '0';
142     elsif clk'event and clk = '1' then  -- rising clock edge
143       if rxDataR = '1' then
144         readData <= '1';
145       else
146         readData <= '0';
147       end if;
148     end if;
149   end process uart_proc;
150
151 end Aramis;
```

Sorgente 21: Classe txUnit della FPGA.

```
1  ----------------------------------------------------------------------
2  -- Title       : txUnit
3  -- Project     : Aramis
4  ----------------------------------------------------------------------
5  -- File        : txUnit.vhd
6  -- Author      : Tomas Alarcon
7  -- Company     : Politecnico di Torino
8  -- Created     : 2009-09-15
9  -- Last update: 2009-11-02
10 -- Platform    : Actel
11 -- Standard    : VHDL'87
12 ----------------------------------------------------------------------
13 -- Description: txUnit is a parallel to serial unit trasmitter. Taken from the
14 -- project done by Philippe and customized for the project Aramis
15 ----------------------------------------------------------------------
16 -- Copyright (c) 2009 This core adheres to the GNU public license
17 ----------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-09-15  1.0      Tomas Alarcon   Customized
21 ----------------------------------------------------------------------
22
23
24 library ieee;
25 use ieee.std_logic_1164.all;
26
27 library work;
28 use work.Op_Code.all;
29
30
31 entity txunit is
32
33   port (
34     uartClk     : in  std_logic;          -- Clock signal
35     Grst        : in  std_logic;          -- Global reset
36     enableTx    : in  std_logic;          -- Enable unit signal
37     loadData    : in  std_logic;          -- Asynchronous load
38     dataIn      : in  std_logic_vector (BYTE -1 downto 0);  -- Byte to transmit
39     tx          : out std_logic;          -- Data output
40     txUnitBusy  : out std_logic);         -- Tx busy
41
42 end txunit;
43
44 architecture Aramis of txunit is
45
46   component synchroniser
47     port (
48       Grst      : in  std_logic;
49       asynchIn  : in  std_logic;            -- Asynchronous signal
50       clk       : in  std_logic;            -- Clock
51       synchOut  : out std_logic);
52   end component;
53
54   signal transBuff       : std_logic_vector (BYTE -1 downto 0);  -- Transmit buffer
55   signal transReg        : std_logic_vector (BYTE -1 downto 0);  -- Transmit
       register
56   signal buffLoad        : std_logic;    -- Buffer loaded
57   signal synchroLoad     : std_logic;    -- Syncrhonised load signal
58   signal bitPos : integer range 0 to 11; -- Bit position in the frame
59
```

```
60 begin  -- Aramis
61
62   sync:synchroniser
63     port map (
64       Grst     => Grst,
65       asynchIn => loadData,
66       clk      => uartClk,
67       synchOut => synchroLoad);
68
69   txUnitBusy <= synchroLoad or buffLoad;
70   -- purpose: transforms a parallel data to serial and to transmit
71   -- type   : sequential
72   -- inputs : uartClk, Grst, enableTx, dataIn, transBuff, transReg, buffLoad
73   -- outputs: tx, txUnitBusy
74   tx_process: process (uartClk, Grst)
75
76   begin  -- process tx_process
77     if Grst = '0' then                  -- asynchronous reset (active low)
78       transReg <= (others => '0');
79       transBuff <= (others => '0');
80       buffLoad <= '0';
81       bitPos <= 0;
82       tx <= '1';
83     elsif uartClk'event and uartClk = '1' then  -- rising clock edge
84       if (enableTx = '0' or enableTx = '1') and synchroLoad = '1' then
85         transBuff <= dataIn;
86         buffLoad <= '1';
87       elsif enableTx = '1' and synchroLoad = '0' then
88         case bitPos is
89           when 0 =>                     -- idle or stop bit
90             tx <= '1';
91             if buffLoad = '1' then
92               transReg <= transBuff;
93               buffLoad <= '0';
94               bitPos <= 1;
95             end if;
96           when 1 =>                     -- start bit
97             tx <= '0';
98             bitPos <= 2;
99           when 10 =>                    -- even bit parity
100            tx <= transReg(0) xor transReg(1) xor transReg(2) xor transReg(3) xor
                 transReg(4) xor transReg(5) xor transReg(6) xor transReg(7);
101            bitPos <= 11;
102          when 11 =>                    --Next is stop bit
103            bitPos <= 0;
104          when others =>                -- Serialisation of transReg
105            tx <= transReg(bitPos-2);
106            bitPos <= bitPos + 1;
107        end case;
108      end if;
109    end if;
110  end process tx_process;
111
112 end Aramis;
```

Sorgente 22: Classe rxUnit della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : rxUnit
3  -- Project    : Aramis
4  ----------------------------------------------------------------------------
5  -- File       : rxUnit.vhd
6  -- Author     : Tomas Alarcon
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-09-15
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: rxUnit is a serial to parallel unit receiver. Taken from the
14 -- project done by Philippe and customized for the project Aramis
15 ----------------------------------------------------------------------------
16 -- Copyright (c) 2009 This core adheres to the GNU public license
17 ----------------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-09-15  1.0      Tomas Alarcon   Created
21 ----------------------------------------------------------------------------
22
23 library ieee;
24 use ieee.std_logic_1164.all;
25
26 library work;
27 use work.Op_Code.all;
28
29
30 entity rxunit is
31
32   port (
33     uartClk       : in  std_logic;      -- System clock signal
34     Grst          : in  std_logic;      -- Global reset
35     rxEnable      : in  std_logic;      -- Enable input
36     readData      : in  std_logic;      -- Async read received byte
37     rx            : in  std_logic;      -- Data input
38     dataAvailable : out std_logic;      -- Byte available
39     dataOut       : out std_logic_vector (BYTE -1 downto 0));  -- Byte received
40
41 end rxunit;
42
43 architecture Aramis of rxunit is
44
45   signal receiveReg  : std_logic_vector ( BYTE -1 downto 0); -- Receiver register
46   signal receiveLoad : std_logic;       -- Byte received
47   signal errorData   : std_logic;         -- indicate an error in the comunication,
         not use in this project
48
49 begin  -- Aramis
50
51   -- purpose: To indicate than a byte is available for to be read
52   -- type   : sequential
53   -- inputs : receiveLoad, Grst, readData
54   -- outputs: dataAvailable
55   data_Available: process (receiveLoad, Grst, readData)
56   begin  -- process data_Available
57     if Grst = '0' or readData = '1' then-- asynchronous reset (active low)
58       dataAvailable <= '0';             -- Negate dataAvailable when receiveReg
             read
```

```
59      elsif rising_edge(receiveLoad) then  -- rising clock edge
60        dataAvailable <= '1';
61      end if;
62    end process data_Available;
63
64    -- purpose: Receiver Process
65    -- type    : sequential
66    -- inputs : uartClk, Grst, rxEnable, rx, receiveReg
67    -- outputs: dataOut
68    rx_proc: process (uartClk, Grst)
69      variable bitPos : integer range 0 to 11;  -- Position of bit in the frame
70      variable sampleCount : integer range 0 to 3;  -- Count from 0 to 3 in each bit
71      variable parity : std_logic;          -- Parity variable
72    begin  -- process rx_proc
73      if Grst = '0' then                    -- asynchronous reset (active low)
74        receiveReg <= (others => '0');
75        sampleCount := 0;
76        dataOut <= (others => '0');
77        receiveLoad <= '0';
78        bitPos := 0;
79        errorData <= '0';
80      elsif uartClk'event and uartClk = '1' then  -- rising clock edge
81        if rxEnable = '1' then
82          case bitPos is
83            when 0 =>                       -- idle
84              receiveLoad <= '0';
85              errorData <= '0';
86              if rx = '0' then
87                sampleCount := 0;
88                bitPos := 1;
89              end if;
90            when 11 =>                      -- Stop bit
91              bitPos:= 0;                   -- next s idle
92              receiveLoad <= '1';           -- Indicate byte received
93              dataOut <= receiveReg;        -- Store received byte
94            when others =>                  -- Deserialisation
95              if sampleCount = 1 and bitPos >= 2 and bitPos < 10 then  -- Sample rx
                    on 1
96                receiveReg(bitPos-2) <= rx;
97              elsif sampleCount = 1 and bitPos = 10  then        -- Parity bit
98                parity := receiveReg(0) xor receiveReg(1) xor receiveReg(2) xor
                      receiveReg(3) xor receiveReg(4) xor receiveReg(5) xor receiveReg
                      (6) xor receiveReg(7) xor rx;
99                if parity = '1' then
100                 errorData<= '1';
101               else
102                 errorData<= '0';
103               end if;
104             end if;
105             if sampleCount = 3 then
106               bitPos := bitPos + 1;
107             end if;
108         end case;
109         if sampleCount = 3 then
110           sampleCount := 0;
111         else
112           sampleCount := sampleCount + 1;
113         end if;
114       end if;
115     end if;
116   end process rx_proc;
117
```

```
118| end Aramis;
```

Sorgente 23: Classi Synchroniser e ClockDivider della FPGA.

```
1  ----------------------------------------------------------------------------
2  -- Title      : Utilitis_Uart
3  -- Project    : Aramis
4  ----------------------------------------------------------------------------
5  -- File       : utilitis_uart.vhd
6  -- Author     : Tomas Alarcon
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-09-15
9  -- Last update: 2009-10-27
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------------
13 -- Description: VHDL utility file. Taken from the
14 -- project done by Philippe and customized for the project Aramis
15 ----------------------------------------------------------------------------
16 -- Copyright (c) 2009 This core adheres to the GNU public license
17 ----------------------------------------------------------------------------
18 -- Revisions  :
19 -- Date        Version  Author          Description
20 -- 2009-09-15  1.0      Tomas Alarcon   Created
21 ----------------------------------------------------------------------------
22
23
24 ----------------------------------------------------------------------------
25 -- Synchroniser:
26 -- Sychronize an input signal (asynchIn) whit an input clock. The result is the
27 -- synchIn signal which is synchronous of clock, and persist for one clock period.
28 ----------------------------------------------------------------------------
29
30 library ieee;
31 use ieee.std_logic_1164.all;
32
33 library work;
34 use work.Op_Code.all;
35
36 entity synchroniser is
37
38   port (
39     Grst     : in  std_logic;
40     asynchIn : in  std_logic;          -- Asynchronous signal
41     clk      : in  std_logic;          -- Clock
42     synchOut : out std_logic);         -- Synchronised signal
43
44 end synchroniser;
45
46 architecture Aramis of synchroniser is
47
48   signal asynchInA : std_logic;
49   signal asynchInS : std_logic;
50   signal synchFlag : std_logic;
51
52 begin  -- Aramis
53
54   Rise_asynchIn: process (asynchIn, synchFlag, Grst)
55   begin  -- process Rise_asynchIn
56     if Grst = '0' or synchFlag = '1' then
57       asynchInA <= '0';
58     elsif rising_edge(asynchIn) then
59       asynchInA <= '1';
60     end if;
```

```
61
62    end process Rise_asynchIn;
63
64    Sync_Process: process (clk,synchFlag,Grst)
65    begin  -- process Sync_Process
66      if Grst = '0' then
67        asynchInS <= '0';
68        synchFlag <= '0';
69      elsif rising_edge(clk) then
70        if asynchInA = '1' then
71          asynchInS <= '1';
72        else
73          asynchInS <= '0';
74        end if;
75        if asynchInS = '1' then
76          synchFlag <= '1';
77        else
78          synchFlag <= '0';
79        end if;
80        if synchFlag = '1' then
81          asynchInS <= '0';
82        end if;
83      end if;
84    end process Sync_Process;
85
86    synchOut <= asynchInS;
87
88  end Aramis;
89
90  ------------------------------------------------------------------------------
91  -- ClockDivider
92  -- This is a parametrizable clock divider.
93  -- The count value is the generic parameter Count.
94  -- It is countEnable enabled. (it will count only if countEnable is high).
95  -- When it overflow, it will emit a pulse on overFlow signal.
96  ------------------------------------------------------------------------------
97  library ieee;
98  use ieee.std_logic_1164.all;
99
100 library work;
101 use work.Op_Code.all;
102
103 entity clockdivider is
104
105   generic (
106     COUNT : integer range 0 to 65535);                    -- Count revolution
107
108   port (
109     clk          : in  std_logic;          -- Clock
110     Grst         : in  std_logic;          -- Global reset
111     countEnable  : in  std_logic;          -- Counter Enable
112     overFlow     : out std_logic);         -- Output
113
114 end clockdivider;
115
116 architecture Aramis of clockdivider is
117
118 begin  -- Aramis
119
120   clock_divider: process (clk, Grst)
121     variable counter : integer range 0 to COUNT-1;
122   begin  -- process clock_divider
```

```
123     if Grst = '0' then                    -- asynchronous reset (active low)
124       counter := COUNT-1;
125       overFlow<= '0';
126     elsif clk'event and clk = '1' then  -- rising clock edge
127       if countEnable = '1' then
128         if counter = 0  then
129           overFlow <= '1';
130           counter := COUNT -1;
131         else
132           overFlow <= '0';
133           counter := counter -1;
134         end if;
135       else
136         overFlow <= '0';
137       end if;
138     end if;
139   end process clock_divider;
140
141 end Aramis;
```

Appendice B

Codice Utilizzato nelle Prove

Sorgente 24: Classe memoryrom della FPGA.

```
1  ----------------------------------------------------------------------
2  -- Title      : MemoryRom
3  -- Project    : Aramis
4  ----------------------------------------------------------------------
5  -- File       : MemoryRom.vhd
6  -- Author     : Tomas Antonio Alarcon Rivera
7  -- Company    : Politecnico di Torino
8  -- Created    : 2009-10-20
9  -- Last update: 2009-11-02
10 -- Platform   : Actel
11 -- Standard   : VHDL'87
12 ----------------------------------------------------------------------
13 -- Description: simulate the Memory rom
14 ----------------------------------------------------------------------
15 -- Copyright (c) 2009
16 ----------------------------------------------------------------------
17 -- Revisions  :
18 -- Date        Version  Author  Description
19 -- 2009-10-20  1.0      Tomas   Created
20 ----------------------------------------------------------------------
21 library ieee;
22 use ieee.std_logic_1164.all;
23 use ieee.std_logic_arith.all;
24 use ieee.std_logic_signed.all;
25
26 library ieee;
27 library proasic3;
28 use proasic3.all;
29
30 library work;
31 use work.Op_Code.all;
32
33 entity memoryrom is
34
35   port (
36     Grst             : in  std_logic;
37     clk              : in  std_logic;
38     getState         : in  std_logic;
39     address          : in  integer range 0 to MEMSIZE - 1;
40     stateMemoryProm  : out std_logic_vector(BYTE -1 downto 0);
```

```
41      dataReady        : out std_logic);
42
43  end memoryrom;
44
45  architecture aramis of memoryrom is
46   type Mem_Type is array (MEMSIZE -1 downto 0) of std_logic_vector (BYTE - 1 downto
         0);
47   constant stateMem : Mem_Type :=(x"ab",x"ab",x"A1",x"D8",x"E7",x"6D",x"21",x"84",x"
         5B",x"33",x"EC",x"FE",x"F8",x"A6",x"B9",x"B2",x"43",x"63",
48   x"05",x"89",x"18",x"02",x"04",x"02",x"00",x"14",x"da",x"13",x"31",x"40",x"00",x"0a
         ",x"3c",x"40",x"00",x"02",x"3d",x"40",x"1c",x"00",x"b0",x"12",
49   x"84",x"13",x"3c",x"40",x"1a",x"11",x"3d",x"40",x"1c",x"11",x"b0",x"12",x"52",x"12
         ",x"b0",x"12",x"86",x"12",x"b0",x"12",x"0c",x"14",x"0a",x"12",
50   x"0b",x"12",x"08",x"12",x"31",x"80",x"14",x"00",x"0a",x"4c",x"38",x"40",x"56",x"00
         ",x"0c",x"41",x"3e",x"40",x"00",x"11",x"3d",x"40",x"0a",x"00",
51   x"b0",x"12",x"ca",x"13",x"ca",x"93",x"06",x"00",x"25",x"20",x"0c",x"4a",x"b0",x"12
         ",x"12",x"12",x"0f",x"4c",x"1b",x"4f",x"02",x"00",x"1e",x"4f",
52   x"02",x"00",x"3e",x"50",x"56",x"00",x"8a",x"4e",x"04",x"00",x"da",x"43",x"06",x"00
         ",x"da",x"43",x"07",x"00",x"8a",x"43",x"0a",x"00",x"12",x"3c",
53   x"1b",x"9a",x"04",x"00",x"0f",x"2c",x"aa",x"eb",x"0a",x"00",x"2b",x"53",x"1b",x"9a
         ",x"04",x"00",x"09",x"20",x"2e",x"4f",x"0e",x"5e",x"0e",x"51",
54   x"aa",x"ee",x"0a",x"00",x"ca",x"43",x"06",x"00",x"ca",x"43",x"07",x"00",x"ca",x"93
         ",x"07",x"00",x"eb",x"23",x"1c",x"4a",x"0a",x"00",x"31",x"50",
55   x"14",x"00",x"30",x"40",x"b2",x"13",x"f2",x"d0",x"30",x"00",x"53",x"00",x"f2",x"d0
         ",x"30",x"00",x"2e",x"00",x"f2",x"d0",x"c0",x"00",x"04",x"00",
56   x"5e",x"42",x"70",x"00",x"c2",x"43",x"70",x"00",x"f2",x"d0",x"10",x"00",x"70",x"00
         ",x"f2",x"d0",x"80",x"00",x"70",x"00",x"f2",x"d0",x"40",x"00",
57   x"70",x"00",x"f2",x"d0",x"20",x"00",x"70",x"00",x"f2",x"d0",x"20",x"00",x"71",x"00
         ",x"f2",x"40",x"6d",x"00",x"74",x"00",x"c2",x"43",x"75",x"00",
58   x"e2",x"42",x"73",x"00",x"d2",x"c3",x"70",x"00",x"f2",x"d0",x"40",x"00",x"00",x"00
         ",x"30",x"41",x"0a",x"12",x"0f",x"43",x"3e",x"40",x"09",x"00",
59   x"3d",x"40",x"00",x"11",x"3a",x"40",x"56",x"00",x"bc",x"90",x"0a",x"00",x"00",x"00
         ",x"0b",x"2c",x"a2",x"4c",x"00",x"02",x"92",x"4c",x"02",x"00",
60   x"02",x"02",x"9c",x"53",x"00",x"00",x"bc",x"50",x"56",x"00",x"02",x"00",x"05",x"3c
         ",x"8c",x"43",x"00",x"00",x"bc",x"40",x"00",x"11",x"02",x"00",
61   x"3c",x"40",x"00",x"02",x"3a",x"41",x"30",x"41",x"0a",x"12",x"0b",x"12",x"08",x"12
         ",x"09",x"12",x"0a",x"4c",x"0b",x"4d",x"68",x"43",x"09",x"4a",
62   x"04",x"3c",x"29",x"53",x"68",x"93",x"08",x"24",x"29",x"53",x"09",x"9b",x"07",x"24
         ",x"89",x"93",x"00",x"00",x"f7",x"27",x"68",x"93",x"f8",x"27",
63   x"a9",x"12",x"f6",x"3f",x"78",x"53",x"ef",x"23",x"30",x"40",x"b0",x"13",x"0f",x"43
         ",x"b2",x"40",x"80",x"5a",x"20",x"01",x"f2",x"d0",x"30",x"00",
64   x"53",x"00",x"e2",x"d3",x"32",x"00",x"d2",x"d3",x"1e",x"00",x"f2",x"40",x"23",x"00
         ",x"40",x"00",x"f2",x"d0",x"80",x"00",x"01",x"00",x"32",x"d2",
65   x"1f",x"53",x"1f",x"53",x"3f",x"90",x"e8",x"fd",x"fb",x"23",x"e2",x"43",x"31",x"00
         ",x"f8",x"3f",x"01",x"3c",x"8f",x"12",x"1e",x"42",x"1a",x"02",
66   x"0e",x"93",x"0b",x"24",x"2f",x"4e",x"a2",x"4f",x"1a",x"02",x"1f",x"4e",x"04",x"00
         ",x"1c",x"4e",x"02",x"00",x"0c",x"93",x"f2",x"23",x"8f",x"12",
67   x"f1",x"3f",x"30",x"41",x"0a",x"12",x"2f",x"4c",x"8f",x"93",x"00",x"00",x"03",x"20
         ",x"1c",x"92",x"1a",x"02",x"02",x"20",x"30",x"40",x"ba",x"13",
68   x"9f",x"42",x"1a",x"02",x"00",x"00",x"82",x"4c",x"1a",x"02",x"3a",x"41",x"30",x"41
         ",x"0d",x"12",x"0c",x"12",x"0f",x"12",x"0e",x"12",x"3c",x"40",
69   x"04",x"02",x"b0",x"12",x"20",x"13",x"6c",x"d3",x"c2",x"ec",x"1d",x"00",x"3e",x"41
         ",x"3f",x"41",x"3c",x"41",x"3d",x"41",x"00",x"13",x"0a",x"12",
70   x"0a",x"4c",x"0c",x"4a",x"2c",x"53",x"b0",x"12",x"40",x"11",x"8a",x"4c",x"0e",x"00
         ",x"1d",x"4a",x"0e",x"00",x"0c",x"4a",x"b0",x"12",x"06",x"14",
71   x"5c",x"43",x"3a",x"41",x"30",x"41",x"0a",x"12",x"0a",x"4c",x"0c",x"4a",x"b0",x"12
         ",x"96",x"13",x"ca",x"43",x"06",x"00",x"ca",x"43",x"07",x"00",
72   x"0c",x"4a",x"3a",x"41",x"30",x"41",x"0a",x"12",x"0a",x"4c",x"0c",x"4a",x"b0",x"12
         ",x"c0",x"11",x"0c",x"4a",x"2c",x"53",x"b0",x"12",x"40",x"13",
73   x"0c",x"4a",x"3a",x"41",x"30",x"41",x"0a",x"12",x"21",x"83",x"0a",x"4c",x"81",x"4a
         ",x"00",x"00",x"0d",x"41",x"5c",x"43",x"b0",x"12",x"10",x"14",
```

```
74  x"f9",x"3f",x"0f",x"4c",x"0f",x"5d",x"03",x"3c",x"cc",x"43",x"00",x"00",x"1c",x"53
        ",x"0c",x"9f",x"fb",x"23",x"30",x"41",x"0f",x"43",x"3e",x"40",
75  x"00",x"11",x"8c",x"43",x"00",x"00",x"bc",x"40",x"00",x"11",x"02",x"00",x"30",x"41
        ",x"35",x"41",x"34",x"41",x"37",x"41",x"36",x"41",x"39",x"41",
76  x"38",x"41",x"3b",x"41",x"3a",x"41",x"30",x"41",x"0d",x"43",x"7c",x"40",x"10",x"00
        ",x"b0",x"12",x"10",x"14",x"1c",x"43",x"30",x"40",x"70",x"13",
77  x"0c",x"12",x"bc",x"4e",x"00",x"00",x"2c",x"53",x"1d",x"83",x"fb",x"23",x"3c",x"41
        ",x"30",x"41",x"3c",x"40",x"04",x"02",x"b0",x"12",x"58",x"13",
78  x"3c",x"40",x"14",x"11",x"30",x"40",x"de",x"12",x"0c",x"12",x"b0",x"12",x"ba",x"12
        ",x"3c",x"41",x"30",x"40",x"70",x"13",x"cc",x"43",x"06",x"00",
79  x"cc",x"43",x"07",x"00",x"30",x"41",x"2c",x"53",x"30",x"40",x"f6",x"13",x"c2",x"4d
        ",x"77",x"00",x"30",x"41",x"30",x"40",x"ea",x"13",x"30",x"41",
80  x"30",x"41",x"ff",x"ff",x"ff",x"ff");
81  begin  -- aramis
82
83    -- purpose: memory
84    -- type    : sequential
85    -- inputs : clk, Grst, getState, address
86    -- outputs: stateMemoryProm
87    process (clk, Grst)
88    begin  -- process
89      if Grst = '0' then                    -- asynchronous reset (active low)
90        stateMemoryProm <= (others => '0');
91        dataReady <= '0';
92
93      elsif clk'event and clk = '1' then  -- rising clock edge
94        if getState = '0' then
95          stateMemoryProm<= stateMem(address);
96          dataReady <= '1';
97        else
98          stateMemoryProm<= (others => '0');
99          dataReady <= '0';
100       end if;
101     end if;
102   end process;
103
104 end aramis;
```

Sorgente 25: Main utilizzato per le simulazioni e prove de SW_WatchDog file:`./Codice/main.cpp`

```
1  //#include "Supervisor.h"
2
3  #include "SW_WatchDog.h"
4
5
6
7
8  //void SmartWatchDog_CPU::Supervisor::main() {
9      //
           ********************************************************************************
10     //  MSP-FET430P430 Demo - Basic Timer, Toggle P5.1 Inside ISR, 32kHz ACLK
11     //
12     //  Description: Toggles P5.1 by xor'ing P5.1 inside of a basic timer ISR.
13     //  ACLK provides the basic timer clock source. LED toggles every 125ms.
14     //  ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = default DCO = 32 x ACLK = 1048576Hz
15     //  //* An external watch crystal between XIN & XOUT is required for ACLK *//
16     //
17     //                  MSP430FG439
18     //               -----------------
19     //           /|\|                 XIN|-
20     //            | |                    | 32kHz
21     //            --|RST            XOUT|-
22     //              |                    |
23     //              |                P5.1|-->LED
24     //
25     //  M. Buccini
26     //  Texas Instruments Inc.
27     //  Feb 2005
28     //  Built with CCE Version: 3.2.0 and IAR Embedded Workbench Version: 3.21A
29     //******************************************************************************
30             SmartWatchDog_CPU::SW_WatchDog swd;
31
32     void main(void)
33     {
34           ushort cou = 0;
35          // unsigned int cou = 0;
36       WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
37       FLL_CTL0 |= XCAP18PF;                     // Set load cap for 32k xtal
38       P5DIR |= 0x02;                            // Set P5.1 as output
39           P4DIR |= 0x01;
40       BTCTL = BTDIV + BT_fCLK2_DIV16;           // ACLK/(256*16)
41       IE2 |= BTIE;                              // Enable BT interrupt
42
43
44       //_BIS_SR(LPM3_bits + GIE);                 // Enter LPM3, enable interrupts
45
46           _BIS_SR(GIE);
47       while(true){
48         cou++;
49             cou = cou +1;
50         if (cou == 65000){
51           P5OUT ^= 0x02;
52         }
53       }
54     }
55
56     // Basic Timer Interrupt Service Routine
57     #pragma vector=BASICTIMER_VECTOR
```

```
58      __interrupt void basic_timer_ISR(void)
59      {
60
61        P4OUT ^= 0x02 | swd.check();
62      }
63  //}
```

Appendice C

Report UML della Smart Watch Dog

Use Case Scheduling

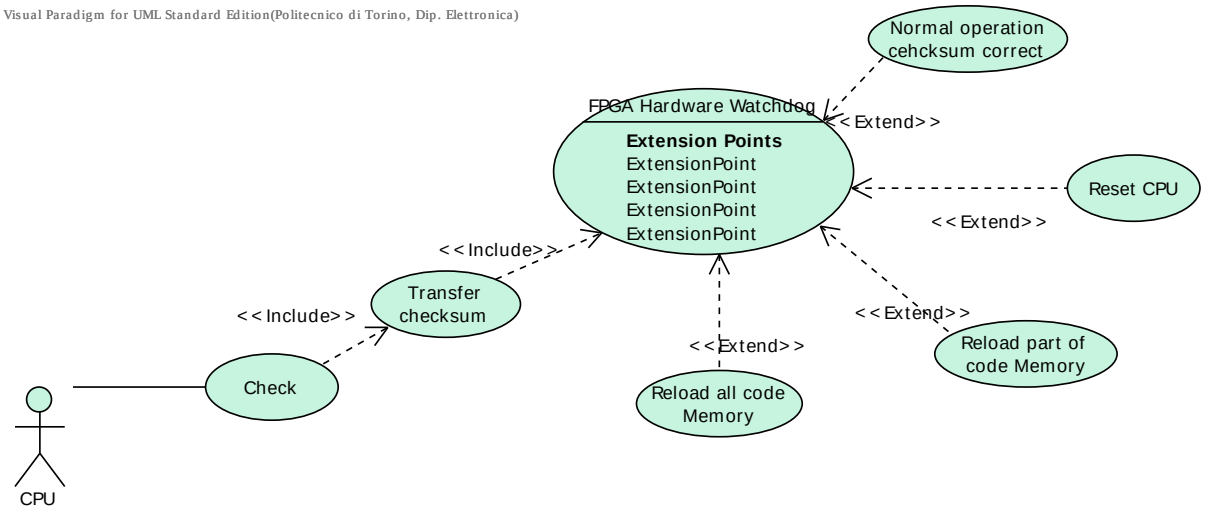
Rank	Use Case Name	Justification
Unspecified	Check	
Unspecified	FPGA Hardware Watchdog	
Unspecified	Transfer checksum	
Unspecified	WatchDogTimer (timeout) Interrupt	
Unspecified	Normal operation cehcksum correct	
Unspecified	Reset CPU	
Unspecified	Reload part of code Memory	
Unspecified	Reload all code Memory	

Stereotypes

HW	
Extend	
Include	
Interface	
access	
user	
SW	
Documentation	Software implementation
HW	
SW	
uController	
uController	
decisionInput	
localPostcondition	
Constant	
common	

Use Case Diagram

Smart WD



Summary

Name	Documentation
CPU	
Check	
Transfer checksum	
FPGA Hardware Watchdog	
Normal operation cehcksum correct	
Reset CPU	
Reload part of code Memory	
Reload all code Memory	

Details

CPU

Check

Transfer checksum

FPGA Hardware Watchdog

Extension Points

ExtensionPoint

ExtensionPoint

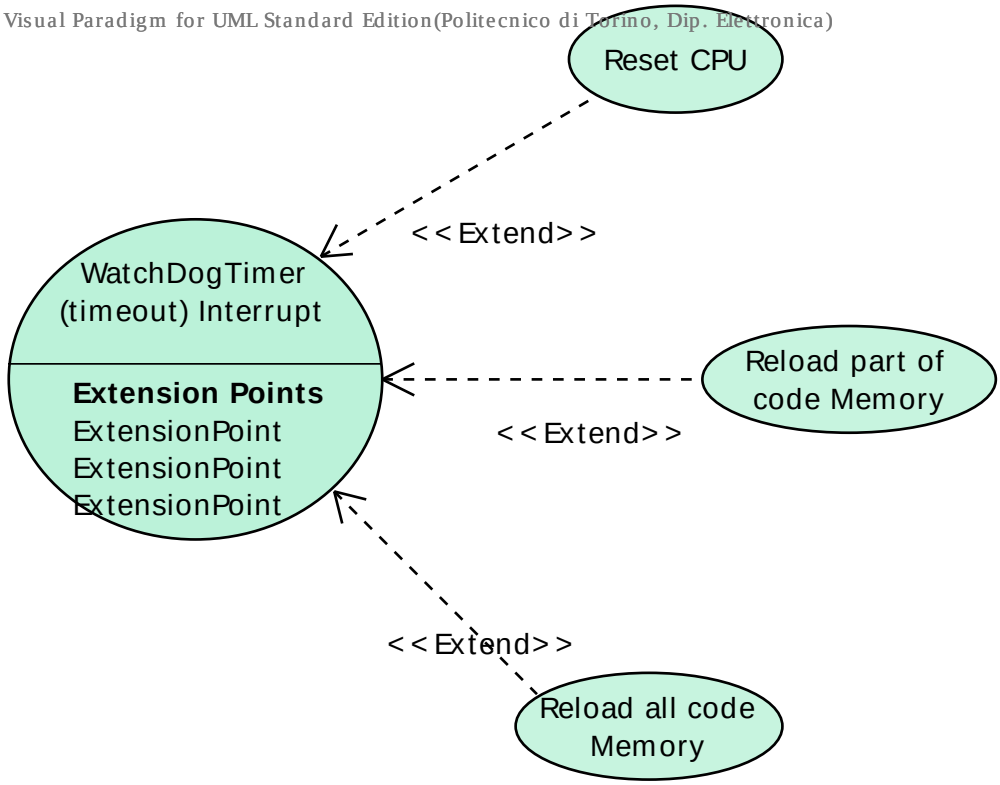
ExtensionPoint

ExtensionPoint

- Normal operation cehcksum correct
- Reset CPU
- Reload part of code Memory
- Reload all code Memory

Use Case Diagram

SWD Interrupt



Summary

Name	Documentation
WatchDogTimer (timeout) Interrupt	
Reset CPU	
Reload part of code Memory	
Reload all code Memory	

Details

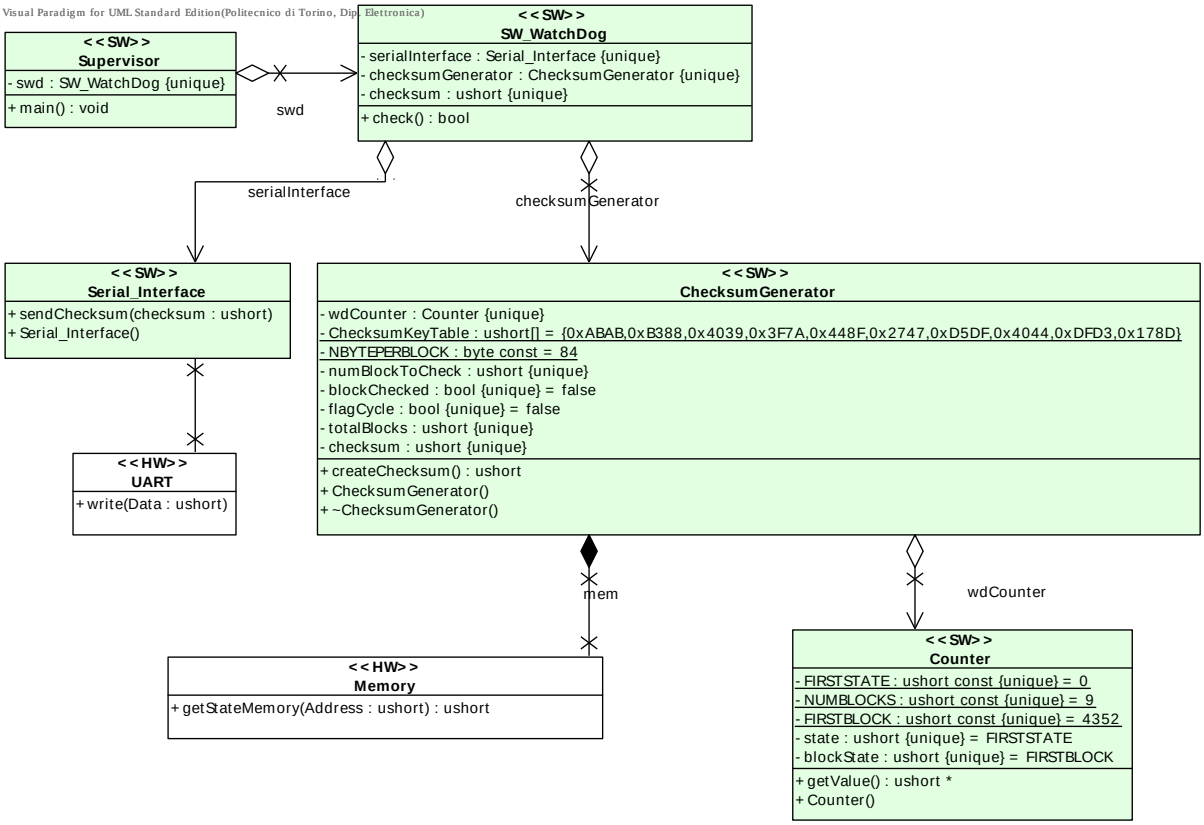
WatchDogTimer (timeout) Interrupt

Extension Points

ExtensionPoint
ExtensionPoint
ExtensionPoint

Class Diagram

Software WatchDog (ucontroller)



Summary

Name	Documentation
SW_WatchDog	
Supervisor	
Serial_Interface	
ChecksumGenerator	
UART	this Class is a class default of the uController the operations described in this class are generic and standard, but they may not coincide exactly with the names of the operations of the UART uController
Counter	
Memory	

Details

SW_WatchDog

Attributes

private serialInterface : Serial_Interface			
Type	Serial_Interface		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private checksumGenerator : ChecksumGenerator			
Type	ChecksumGenerator		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private checksum : ushort			
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public check () : bool	
Static	false
Leaf	false
Transit To	→ N/A (<unnamed> -> <unnamed>)
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

Supervisor

Attributes

private swd : SW_WatchDog			
Type	SW_WatchDog		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public main () : void	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

Serial_Interface

Operations

public sendChecksum (checksum : ushort)		
Parameters	checksum	
	Multiplicity	Unspecified
	Type	ushort
	Direction	inout
Static	false	
Leaf	false	
Documentation	It sends checksum to the FPGA.	
Transit To	→ N/A (<unnamed> -> <unnamed>)	
Ordered	false	
Unique	true	
Dotnet Code Detail	N/A	
Query	false	

public Serial_Interface ()	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

ChecksumGenerator

Attributes

private wdCounter : Counter			
Type	Counter		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private ChecksumKeyTable : ushort			
Documentation	This is a constant "look up table"		
Initial Value	{0xABAB,0xB388,0x4039,0x3F7A,0x448F,0x2747,0xD5DF,0x4044,0xDFD3,0x178D}		
Type Modifier	[]		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity Detail	N/A		
Multiplicity	Unspecified		
Dotnet Code Detail	N/A		
Aggregation	None		

private NBYTEPERBLOCK : byte			
Initial Value	84		
Type	byte		
Getter	false	Setter	false
Derived	false		
Multiplicity Detail	N/A		
Multiplicity	Unspecified		
Dotnet Code Detail	N/A		
Aggregation	None		

private numBlockToCheck : ushort			
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private blockChecked : bool			
Initial Value	false		
Type	bool		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private flagCycle : bool			
Initial Value	false		
Type	bool		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private totalBlocks : ushort			
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private checksum : ushort			
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public createChecksum () : ushort	
Static	false
Leaf	false
Documentation	Generates checksum of sequence specified by **sequence.**
Transit To	→ N/A (<unnamed> -> <unnamed>)
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

public ChecksumGenerator ()	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

public ChecksumGenerator ()	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

UART

Operations

public write (Data : ushort)		
Parameters	Data	
	Multiplicity	Unspecified
	Type	ushort
	Direction	inout
Static	false	
Leaf	false	
Transit To	→ N/A (<unnamed> -> <unnamed>)	
Ordered	false	
Unique	true	
Query	false	

Counter

Attributes

private FIRSTSTATE : ushort			
Initial Value	0		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Dotnet Code Detail	N/A		
Aggregation	None		

private NUMBLOCKS : ushort			
Initial Value	9		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Dotnet Code Detail	N/A		
Aggregation	None		

private FIRSTBLOCK : ushort			
Initial Value	4352		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Dotnet Code Detail	N/A		
Aggregation	None		

private state : ushort			
Initial Value	FIRSTSTATE		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private blockState : ushort			
Initial Value	FIRSTBLOCK		
Type	ushort		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public getValue () : ushort	
Static	false
Leaf	false
Documentation	Returns counter value and Increment counter
Transit To	→ N/A (<unnamed> -> <unnamed>)
Type Modifier	*
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

public Counter ()	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

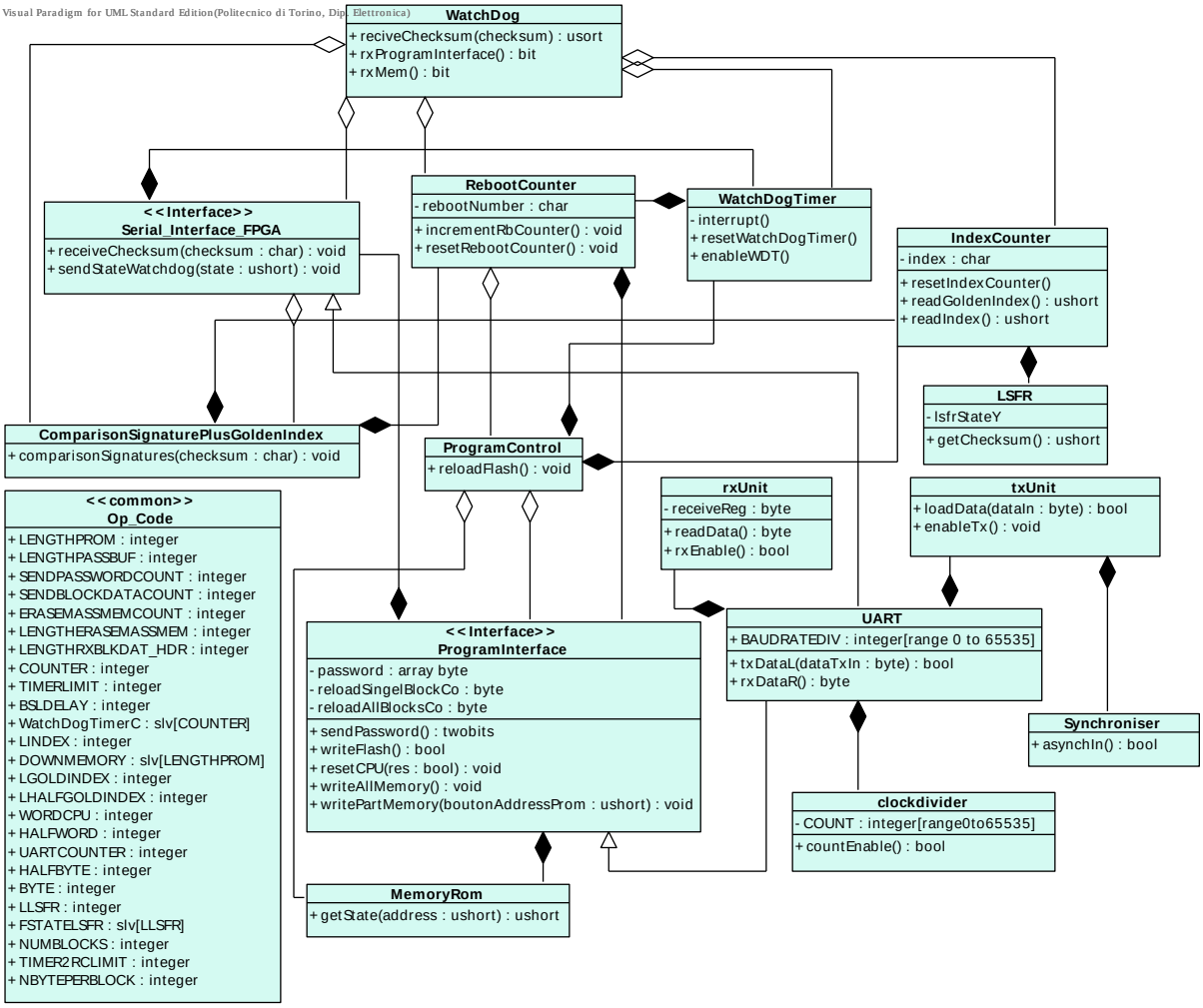
Memory

Operations

public getStateMemory (Address : ushort) : ushort		
Parameters	Address	
	Multiplicity	Unspecified
	Type	ushort
	Direction	inout
Static	false	
Leaf	false	
Transit To	Value	
	→ N/A	
	→ N/A	
	→ N/A	
Ordered	false	
Unique	true	
Query	false	

Class Diagram

FPGA Watch Dog



Summary

Name	Documentation
WatchDog	
RebootCounter	
WatchDogTimer	
Serial_Interface_FPGA	
IndexCounter	
LSFR	
ComparisonSignaturePlusGoldenIndex	
ProgramControl	
rxUnit	
txUnit	

Op_Code	
UART	
ProgramInterface	
Synchroniser	
clockdivider	
MemoryRom	

Details

WatchDog

Operations

public reciveChecksum (checksum) : usort		
Parameters	checksum	
	Multiplicity	Unspecified
	Direction	in
Static	false	
Leaf	false	
Ordered	false	
Unique	true	
Query	false	

public rxProgramInterface () : bit	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

public rxMem () : bit	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

RebootCounter

Attributes

private rebootNumber : char			
Type	char		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public incrementRbCounter () : void	
Static	false
Leaf	false
Transit To	Value
	→ N/A
	→ N/A
	→ N/A
Ordered	false
Unique	true
Query	false

public resetRebootCounter () : void	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

WatchDogTimer

Operations

private interrupt ()	
Static	false
Leaf	false
Transit To	→ N/A (<unnamed> -> <unnamed>)
Ordered	false
Unique	true
Query	false

public resetWatchDogTimer ()	
Static	false
Leaf	false
Transit To	**Value**
	→ N/A
	→ N/A
Ordered	false
Unique	true
Query	false

public enableWDT ()	
Static	false
Leaf	false
Transit To	**Value**
	→ Unable:
	→ enable:
Ordered	false
Unique	true
Query	false

Serial_Interface_FPGA

Operations

public receiveChecksum (checksum : char) : void		
Parameters	**checksum**	
	Multiplicity	Unspecified
	Type	char
	Direction	inout
Static	false	
Leaf	false	
Transit To	**Value**	
	→ N/A	
	→ N/A	
	→ N/A	
Ordered	false	
Unique	true	
Query	false	

public sendStateWatchdog (state : ushort) : void		
Parameters	state	
	Multiplicity	Unspecified
	Type	ushort
	Direction	in
Static	false	
Leaf	false	
Ordered	false	
Unique	true	
Query	false	

IndexCounter

Attributes

private index : char			
Type	char		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public resetIndexCounter ()	
Static	false
Leaf	false
Ordered	false
Unique	true
Dotnet Code Detail	N/A
Query	false

public readGoldenIndex () : ushort	
Static	false
Leaf	false
Documentation	This method read actual golden index (checksum), generate the next golden index and increment the index counter.
Transit To	→ N/A (<unnamed> -> <unnamed>)
Ordered	false
Unique	true
Query	false

public readIndex () : ushort	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

LSFR

Attributes

private lsfrStateY			
Initial Value	#1000		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public getChecksum () : ushort	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

ComparisonSignaturePlusGoldenIndex

Operations

public comparisonSignatures (checksum : char) : void		
Parameters	checksum	
	Multiplicity	Unspecified
	Type	char
	Direction	inout
Static	false	
Leaf	false	
Transit To	Value	
	→ N/A	
	→ N/A	
Ordered	false	
Unique	true	
Query	false	

ProgramControl

Operations

public reloadFlash () : void	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

rxUnit

Attributes

private receiveReg : byte			
Type	byte		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public readData () : byte	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

public rxEnable () : bool	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

txUnit

Operations

public loadData (dataIn : byte) : bool		
Parameters	dataIn	
	Multiplicity	Unspecified
	Type	byte
	Direction	in
Static	false	
Leaf	false	
Ordered	false	
Unique	true	
Query	false	

public enableTx () : void	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

Op_Code

Attributes

public LENGTHPROM : integer			
Initial Value	1		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LENGTHPASSBUF : integer			
Initial Value	30		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public SENDPASSWORDCOUNT : integer			
Initial Value	31		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public SENDBLOCKDATACOUNT : integer			
Initial Value	261		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public ERASEMASSMEMCOUNT : integer			
Initial Value	11		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LENGTHERASEMASSMEM : integer			
Initial Value	10		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LENGTHRXBLKDAT_HDR : integer			
Initial Value	8		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public COUNTER : integer			
Initial Value	8		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public TIMERLIMIT : integer			
Initial Value	200000		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public BSLDELAY : integer			
Initial Value	24000		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public WatchDogTimerC : slv			
Initial Value	x"EA"		
Type Modifier	[COUNTER]		
Type	slv		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LINDEX : integer			
Initial Value	16		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public DOWNMEMORY : slv			
Initial Value	x"0000"		
Type Modifier	[LENGTHPROM]		
Type	slv		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LGOLDINDEX : integer			
Initial Value	16		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LHALFGOLDINDEX : integer			
Initial Value	8		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public WORDCPU : integer			
Initial Value	16		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public HALFWORD : integer			
Initial Value	8		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public UARTCOUNTER : integer			
Initial Value	4		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public HALFBYTE : integer			
Initial Value	4		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public BYTE : integer			
Initial Value	8		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public LLSFR : integer			
Initial Value	5		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public FSTATELSFR : slv			
Initial Value	"1000"		
Type Modifier	[LLSFR]		
Type	slv		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public NUMBLOCKS : integer			
Initial Value	262		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public TIMER2RCLIMIT : integer			
Initial Value	200000000		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

public NBYTEPERBLOCK : integer			
Initial Value	250		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

UART

Attributes

public BAUDRATEDIV : integer			
Initial Value	260		
Type Modifier	[range 0 to 65535]		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public txDataL (dataTxIn : byte) : bool		
Parameters	dataTxIn	
	Multiplicity	Unspecified
	Type	byte
	Direction	in
Static	false	
Leaf	false	
Ordered	false	
Unique	true	
Query	false	

public rxDataR () : byte	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

ProgramInterface

Attributes

private password : array byte			
Type	array byte		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private reloadSingelBlockCo : byte			
Type	byte		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

private reloadAllBlocksCo : byte			
Type	byte		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public sendPassword () : twobits	
Static	false
Leaf	false
Documentation	PasswordInfo: if password is good the passwordinfo has the value 1h if password is wrong the passwordinfo has the value Ah others has the value Fh
Transit To	→ N/A (<unnamed> -> <unnamed>)
Ordered	false
Unique	true
Query	false

public writeFlash () : boolean	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

public resetCPU (res : bool) : void		
Parameters	res	
	Multiplicity	Unspecified
	Type	bool
	Direction	inout
Static	false	
Leaf	false	
Transit To	Value	
	→ N/A	
	→ N/A	
Ordered	false	
Unique	true	
Query	false	

public writeAllMemory () : void	
Static	false
Leaf	false
Transit To	**Value** → N/A → N/A
Ordered	false
Unique	true
Query	false

public writePartMemory (boutonAddressProm : ushort) : void		
Parameters	**boutonAddressProm**	
	Multiplicity	Unspecified
	Type	ushort
	Direction	in
Static	false	
Leaf	false	
Transit To	→ N/A (<unnamed> -> <unnamed>)	
Ordered	false	
Unique	true	
Query	false	

Synchroniser

Operations

public asynchIn () : bool	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

clockdivider

Attributes

private COUNT : integer			
Type Modifier	[range0to65535]		
Type	integer		
Getter	false	Setter	false
Derived	false		
Multiplicity	Unspecified		
Aggregation	None		

Operations

public countEnable () : bool	
Static	false
Leaf	false
Ordered	false
Unique	true
Query	false

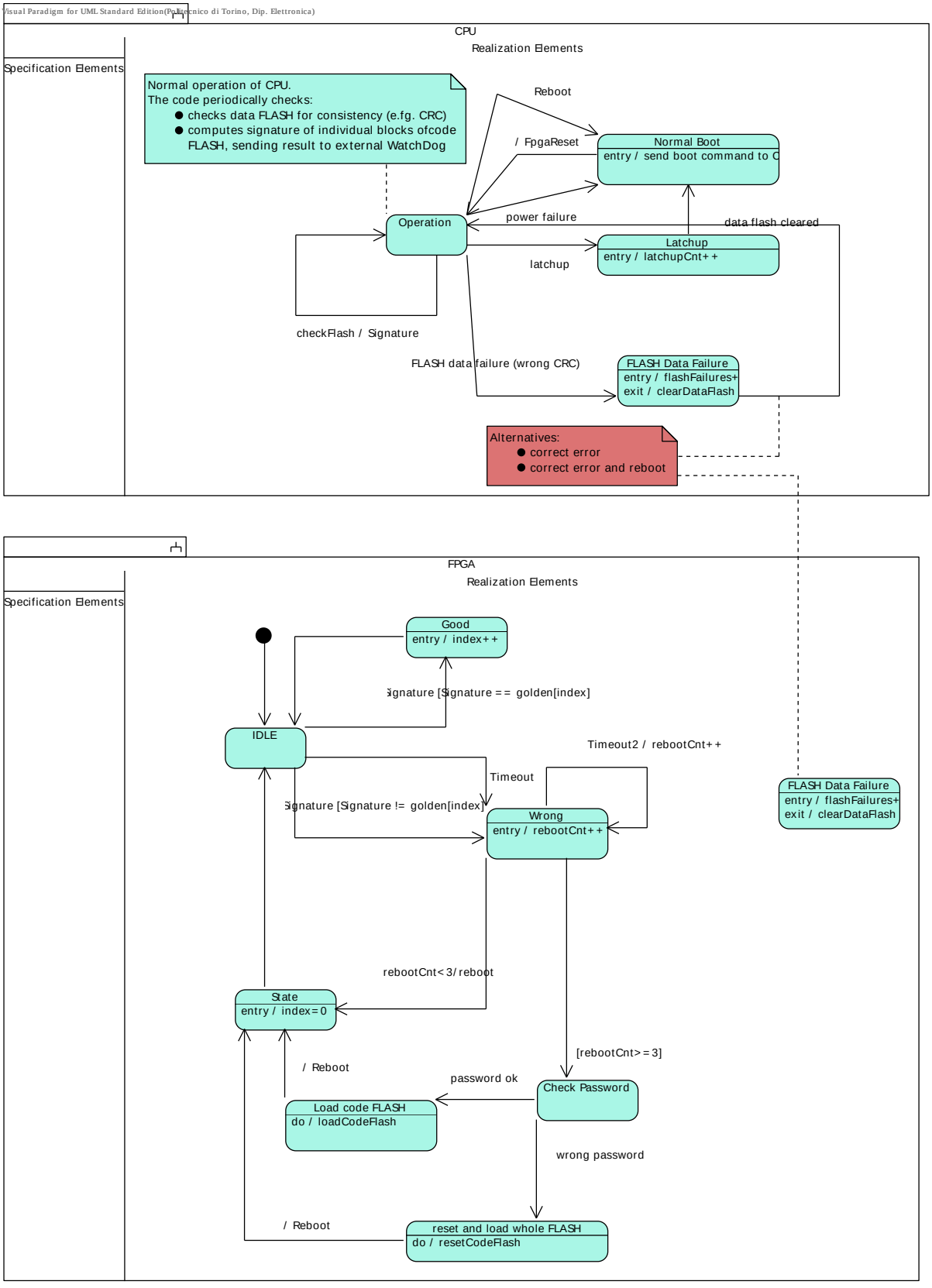
MemoryRom

Operations

public getState (address : ushort) : ushort		
Parameters	address	
	Multiplicity	Unspecified
	Type	ushort
	Direction	inout
Static	false	
Leaf	false	
Ordered	false	
Unique	true	
Query	false	

State Machine Diagram

Radiation Tolerance



Summary

Name	Documentation
Initial	
FPGA	
IDLE	
Good	
Wrong	
Check Password	Activates a deep reset of CPU, including code and data FLASH
Load code FLASH	Activates a deep reset of CPU, including code and data FLASH. Also reads, increments and writes flashCodeFailures in an unchecked FLASH areas
reset and load whole FLASH	Activates a deep reset of CPU, including code and data FLASH. Also writes flashCodeFailures=1 in an unchecked FLASH areas
State	
CPU	
Normal Boot	Activates CPU normal boot mechanism
Operation	
FLASH Data Failure	
Latchup	
N/A	Normal operation of CPU. The code periodically checks: • checks data FLASH for consistency (e.fg. CRC) • computes signature of individual blocks of code FLASH, sending result to external WatchDog
N/A	Alternatives: • correct error • correct error and reboot
FLASH Data Failure	

Details

Initial

FPGA

Children

Name	Documentation
Check Password	Activates a deep reset of CPU, including code and data FLASH
Load code FLASH	Activates a deep reset of CPU, including code and data FLASH. Also reads, increments and writes flashCodeFailures in an unchecked FLASH areas
State	
reset and load whole FLASH	Activates a deep reset of CPU, including code and data FLASH. Also writes flashCodeFailures=1 in an unchecked FLASH areas

IDLE

Good

Wrong

Check Password

Load code FLASH

reset and load whole FLASH

State

CPU

Children

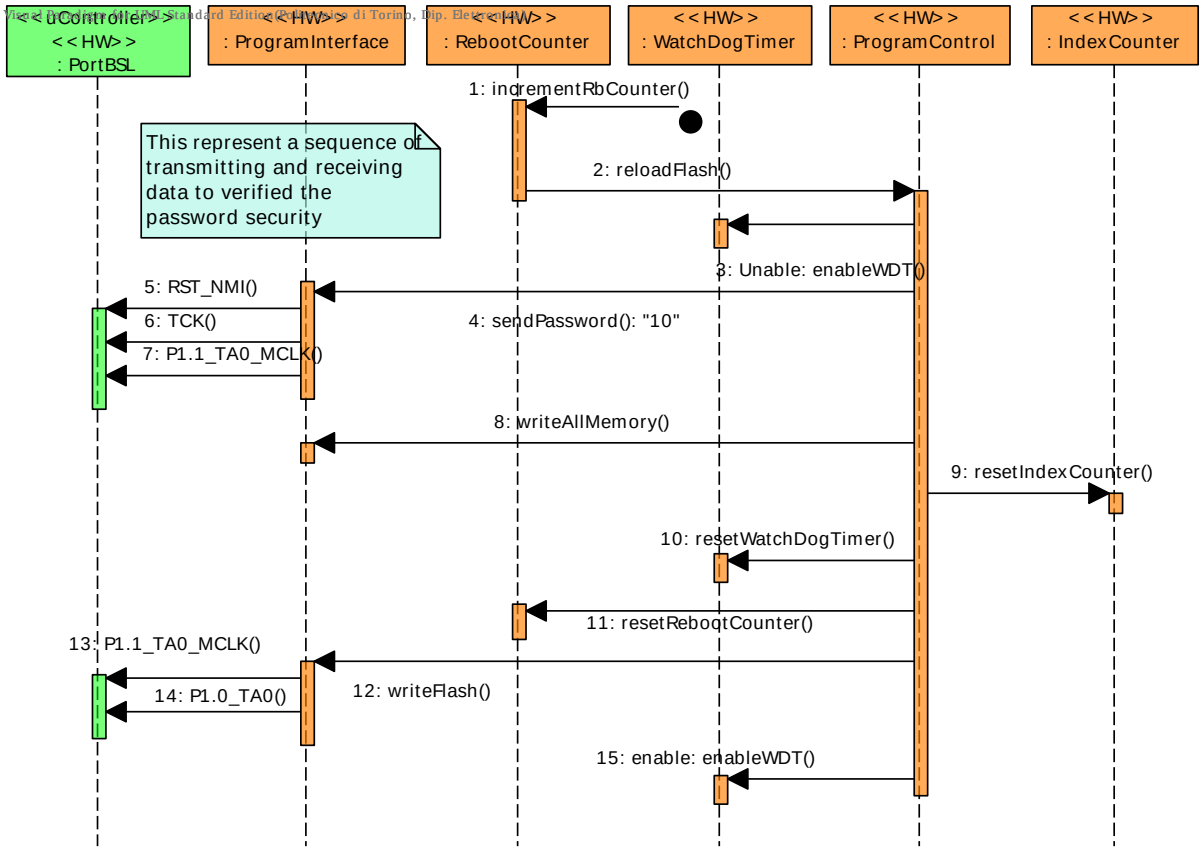
Name	Documentation
N/A	Normal operation of CPU. The code periodically checks: • checks data FLASH for consistency (e.fg. CRC) • computes signature of individual blocks of code FLASH, sending result to external WatchDog

N/A	Alternatives: • correct error • correct error and reboot
FLASH Data Failure	
FLASH Data Failure	
RAM Data Failure	
RAM Data Failure	
Normal Boot	Activates CPU normal boot mechanism
Operation	

Normal Boot
Operation
FLASH Data Failure
Latchup
N/A
N/A
FLASH Data Failure

Sequence Diagram

Reload_All_Memory



Summary

Name	Documentation
: RebootCounter	
: ProgramControl	
: WatchDogTimer	
: ProgramInterface	
: PortBSL	
: IndexCounter	
N/A	This represent a sequence of transmitting and receiving data to verified the password security

Details

: RebootCounter

 **: ProgramControl**

 **: WatchDogTimer**

 **: ProgramInterface**

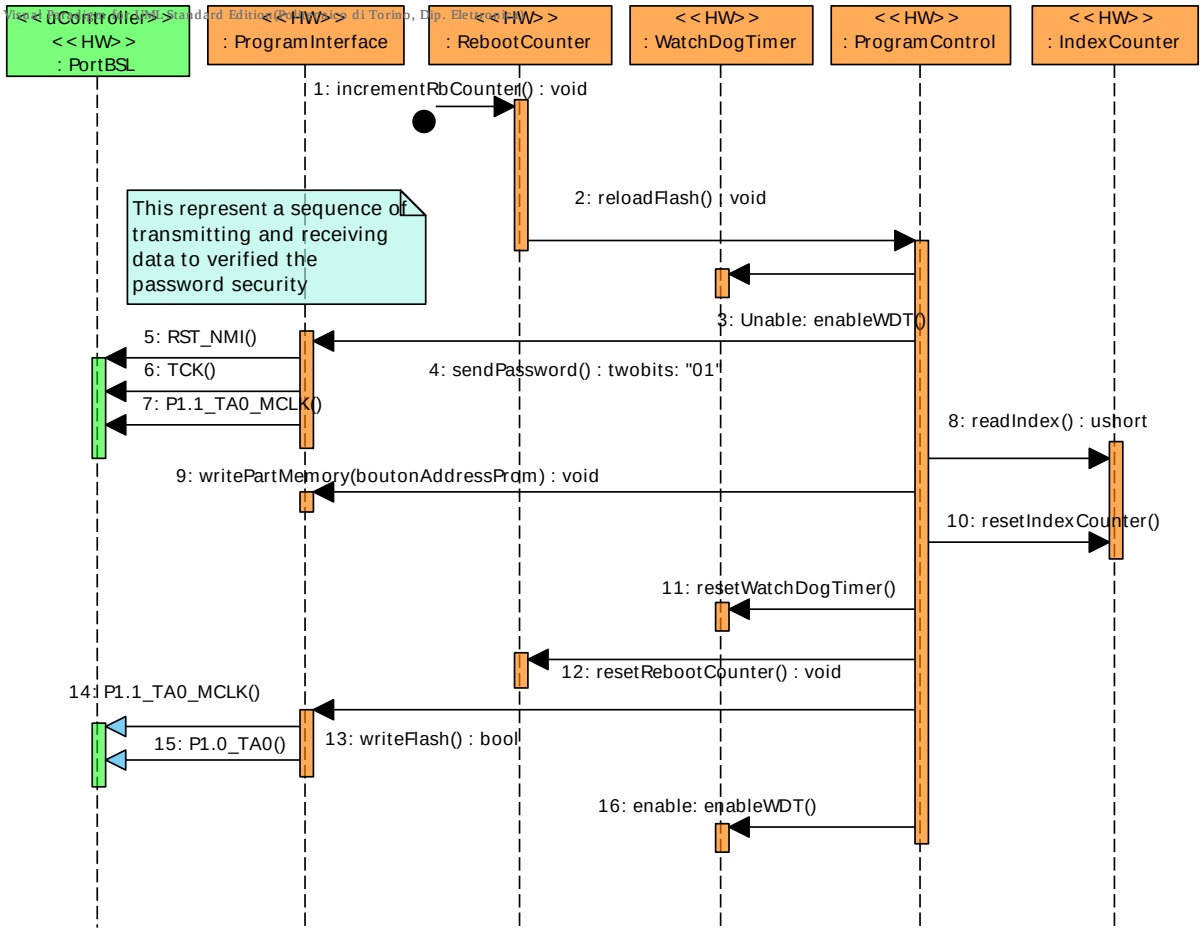
 **: PortBSL**

 **: IndexCounter**

 **N/A**

Sequence Diagram

Reload_Part_Memory



Summary

Name	Documentation
: RebootCounter	
: ProgramControl	
: WatchDogTimer	
: ProgramInterface	
: PortBSL	
: IndexCounter	
N/A	This represent a sequence of transmitting and receiving data to verified the password security

Details

: RebootCounter

: ProgramControl

: WatchDogTimer

: ProgramInterface

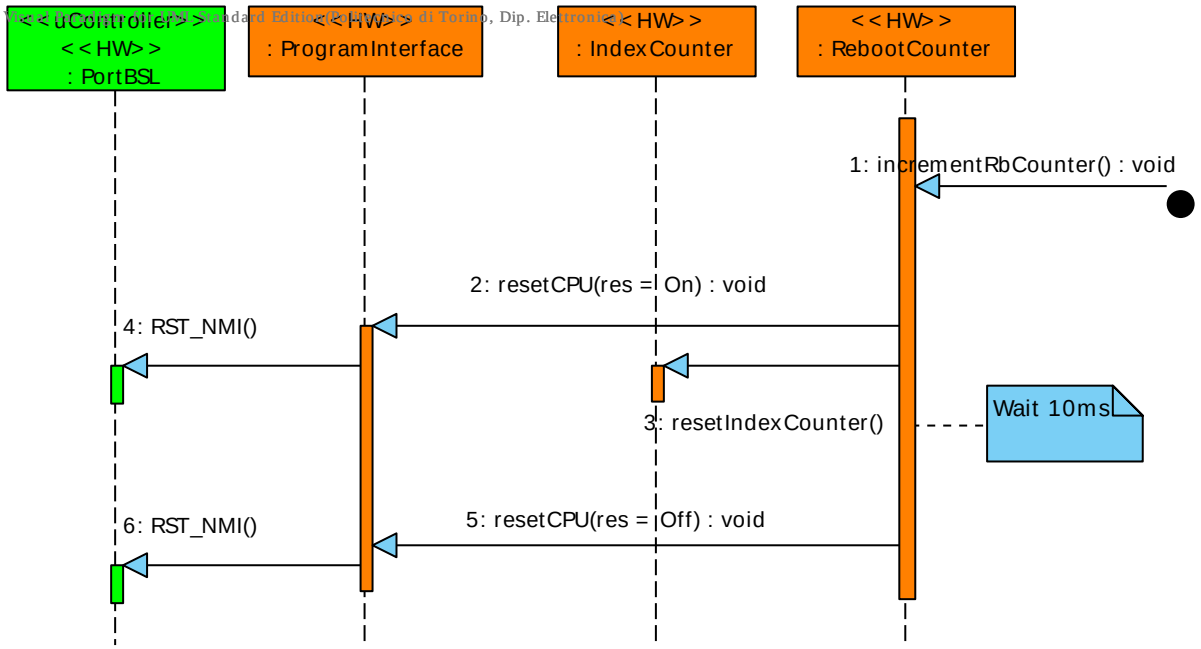
: PortBSL

: IndexCounter

N/A

Sequence Diagram

Reset_Cpu



Summary

Name	Documentation
: RebootCounter	
: ProgramInterface	
: PortBSL	
: IndexCounter	
N/A	Wait 10ms

Details

: RebootCounter

: ProgramInterface

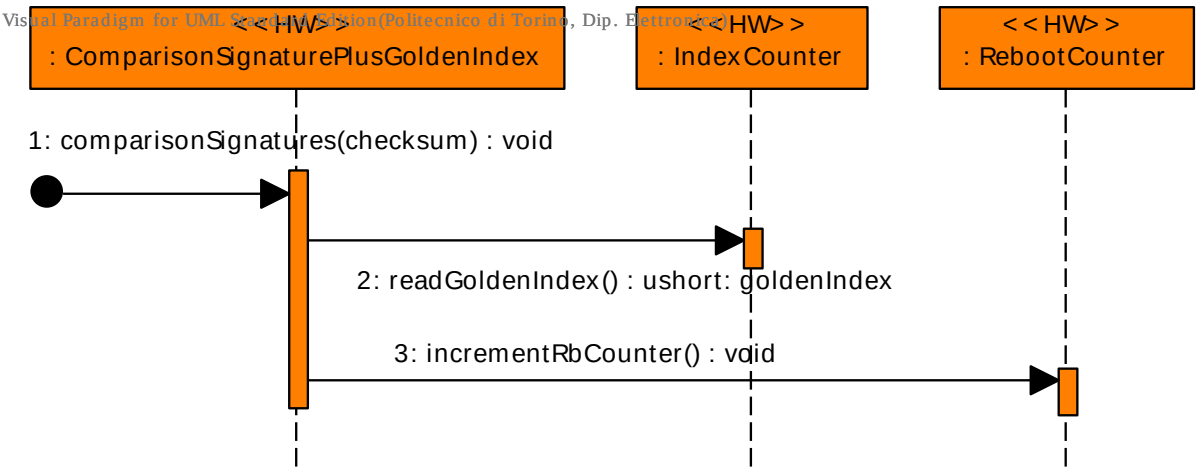
: PortBSL

: IndexCounter

N/A

Sequence Diagram

checksum!=goldenIndex



Summary

Name	Documentation
: ComparisonSignaturePlusGoldenIndex	
: IndexCounter	
: RebootCounter	

Details

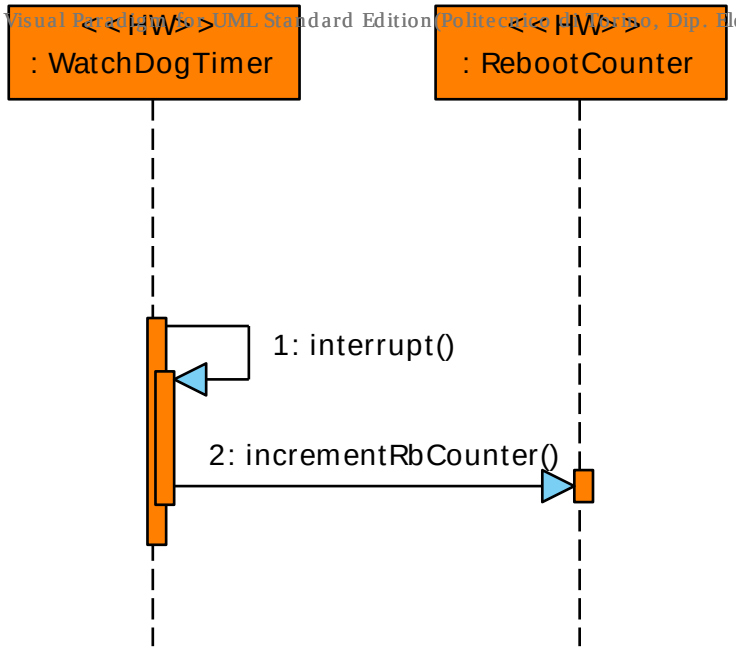
: ComparisonSignaturePlusGoldenIndex

: IndexCounter

: RebootCounter

Sequence Diagram

WatchdogTimer_Interruption



Summary

Name	Documentation
: WatchDogTimer	
: RebootCounter	

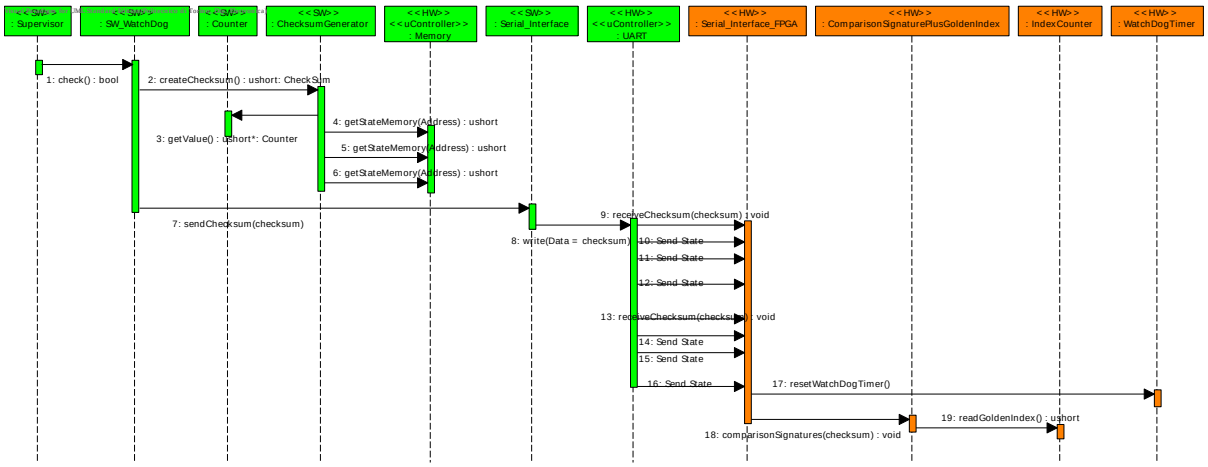
Details

: WatchDogTimer

: RebootCounter

Sequence Diagram

Normal Operation



Summary

Name	Documentation
: Supervisor	
: SW_WatchDog	
: ChecksumGenerator	
: Counter	
: Memory	
: Serial_Interface	
: UART	
: Serial_Interface_FPGA	
: WatchDogTimer	
: ComparisonSignaturePlusGoldenIndex	
: IndexCounter	

Details

: Supervisor

: SW_WatchDog

: ChecksumGenerator

: Counter

 **: Memory**

 **: Serial_Interface**

 **: UART**

 **: Serial_Interface_FPGA**

 **: WatchDogTimer**

 **: ComparisonSignaturePlusGoldenIndex**

 **: IndexCounter**

Appendice D

Software Utilizzati per lo svolgimento di questa tesi

1. Visual Paradigm 7.0.
2. IDE IAR Worckbench.
3. Libero Project 8.6.
4. Aqua Emacs.
5. Symplify Pro
6. Designer
7. Modelsim ES6.5
8. Code::Blocks 8.02
9. LaTeX

Appendice E

Dispositivi

Stereotypes

HW	
Extend	
Include	
Interface	
access	
user	
SW	
Documentation	Software implementation
HW	
SW	
uController	
uController	
decisionInput	
localPostcondition	
Constant	
common	

Figure A.2 shows the bottom view of the 484-Pin FBGA.

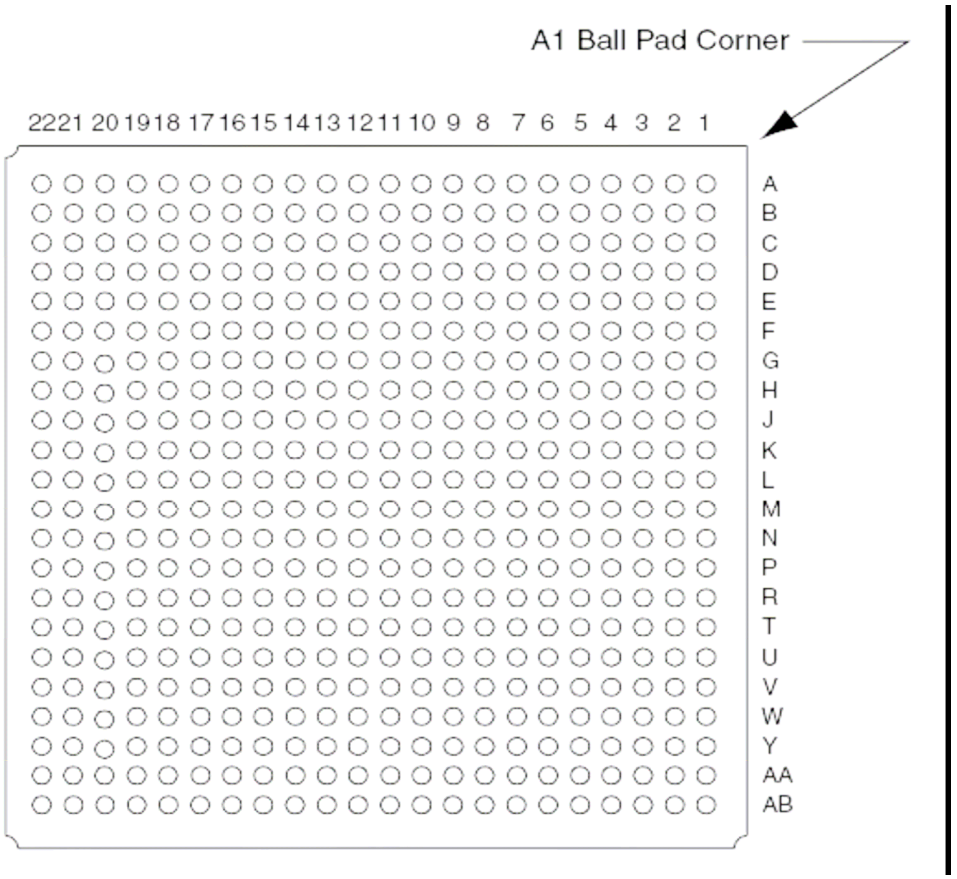


Figure A.2- 484 Pin FBGA Package

Note: For package manufacturing and environmental information, visit the Packaging Solutions page:

http://www.actel.com/products/solutions/package/default.aspx.

Due to the comprehensive and flexible nature of M1AGL FPGA device user I / Os, a naming scheme is used to show the details of the I / O. The name identifies to which I / O bank it belongs, as well as the pairing and pin polarity for differential I / Os.

I / O Nomenclature	=	Gmn / IOuxwBy
Gmn is only used for I / Os that also have CCC access – i.e., global pins.		
G	=	Global
m	=	Global pin location associated with each CCC on the device: A (northwest corner), B (northeast corner), C (east middle), D (southeast corner), E (southwest corner), and F (west middle)
n	=	Global input MUX and pin number of the associated Global location m, either A0, A1, A2, B0, B1, B2, C0, C1, or C2
u	=	I / O pair number in the bank, starting at 00 from the northwest I / O bank in a clockwise direction
x	=	P (Positive) or N (Negative) for differential pairs, or R (Regular – single-ended) for the I / Os that support single-ended and voltage-referenced I / O standards only. U (Positive-LVDS only) or V (Negative-LVDS only) restrict the I / O differential pair from being selected as LVPECL pair.
w	=	D (Differential Pair) or P (Pair) or S (Single-Ended). D (Differential Pair) if both members of the pair are bonded out to adjacent pins or are separated only by one GND or NC pin; P (Pair) if both members of the pair are bonded out but do not meet the adjacency requirement; or S (Single-Ended) if the I / O pair is not bonded out. For Differential (D) pairs, adjacency for ball grid packages means only vertical or horizontal. Diagonal adjacency does not meet the requirements for a true differential pair.
B	=	Bank
y	=	Bank number [0..3]. Bank number starting at 0 from the northwest I / O bank in a clockwise direction

Table A.1 - I/O Nomenclature

The following table shows the FPGA pin connections to the M1AGL Development Board signals and the Sample Design signals. Note that the board also allows use of the FG256 BGA package although the FG484 is supplied.

FG256 Ball	FG484 Ball	Pre-loaded Design Signal	Schematic Signal	I / O	Dev. Kit Function
N6	T9	pbRstN	BUF2_PBRST_N	I	Push Button Reset
R4	V7	poRstN	PORESET_N	I	Power on Reset
N4	T7	flashRstN	FLASH_RST_N	O	Reset to Flash Chips
B1	E4	extClk	OSC_CLK	I	System Clock
D6	G9	rs232Atx	RS232_TX	O	UART Transmit
A6	D9	rs232Arx	RS232_RX	I	UART Receive
N3	T6	memAddr[2]	MEM_ADDR2	O	SRAM / FLASH Address
M4	R7	memAddr[3]	MEM_ADDR3	O	SRAM / FLASH Address
C5	F8	memAddr[4]	MEM_ADDR4	O	SRAM / FLASH Address
C6	F9	memAddr[5]	MEM_ADDR5	O	SRAM / FLASH Address
B4	E7	memAddr[6]	MEM_ADDR6	O	SRAM / FLASH Address
A4	D7	memAddr[7]	MEM_ADDR7	O	SRAM / FLASH Address
C2	F5	memAddr[8]	MEM_ADDR8	O	SRAM / FLASH Address
C1	F4	memAddr[9]	MEM_ADDR9	O	SRAM / FLASH Address
D4	G7	memAddr[10]	MEM_ADDR10	O	SRAM / FLASH Address
D2	G5	memAddr[11]	MEM_ADDR11	O	SRAM / FLASH Address
D1	G4	memAddr[12]	MEM_ADDR12	O	SRAM / FLASH Address
E4	H7	memAddr[13]	MEM_ADDR13	O	SRAM / FLASH Address
E3	H6	memAddr[14]	MEM_ADDR14	O	SRAM / FLASH Address
E2	H5	memAddr[15]	MEM_ADDR15	O	SRAM / FLASH Address
F2	J5	memAddr[16]	MEM_ADDR16	O	SRAM / FLASH Address
K1	N4	memAddr[17]	MEM_ADDR17	O	SRAM / FLASH Address
G4	K7	memAddr[18]	MEM_ADDR18	O	SRAM / FLASH Address
H5	L8	memAddr[19]	MEM_ADDR19	O	SRAM / FLASH Address
J5	M8	memAddr[20]	MEM_ADDR20	O	SRAM / FLASH Address
H3	L6	memAddr[21]	MEM_ADDR21	O	SRAM / FLASH Address
H1	L4	memAddr[22]	MEM_ADDR22	O	SRAM / FLASH Address
J1	M4	memAddr[23]	MEM_ADDR23	O	SRAM / FLASH Address
J2	M5	memAddr[24]	MEM_ADDR24	O	SRAM / FLASH Address
M2	R5	memAddr[25]	MEM_ADDR25	O	SRAM / FLASH Address
T8	W11	memData[0]	MEM_DATA0	I / O	SRAM / FLASH Data
P8	U11	memData[1]	MEM_DATA1	I / O	SRAM / FLASH Data
R8	V11	memData[2]	MEM_DATA2	I / O	SRAM / FLASH Data
R7	V10	memData[3]	MEM_DATA3	I / O	SRAM / FLASH Data
T7	W10	memData[4]	MEM_DATA4	I / O	SRAM / FLASH Data
P7	U10	memData[5]	MEM_DATA5	I / O	SRAM / FLASH Data
N8	T11	memData[6]	MEM_DATA6	I / O	SRAM / FLASH Data
T6	W9	memData[7]	MEM_DATA7	I / O	SRAM / FLASH Data
R6	V9	memData[8]	MEM_DATA8	I / O	SRAM / FLASH Data
P6	U9	memData[9]	MEM_DATA9	I / O	SRAM / FLASH Data
N7	T10	memData[10]	MEM_DATA10	I / O	SRAM / FLASH Data
T5	W8	memData[11]	MEM_DATA11	I / O	SRAM / FLASH Data
R5	V8	memData[12]	MEM_DATA12	I / O	SRAM / FLASH Data
B5	E8	memData[13]	MEM_DATA13	I / O	SRAM / FLASH Data

T4	W7	memData[14]	MEM_DATA14	I / O	SRAM / FLASH Data
B6	E9	memData[15]	MEM_DATA15	I / O	SRAM / FLASH Data
N10	T13	memData[16]	MEM_DATA16	I / O	SRAM / FLASH Data
T11	W14	memData[17]	MEM_DATA17	I / O	SRAM / FLASH Data
R11	V14	memData[18]	MEM_DATA18	I / O	SRAM / FLASH Data
P10	U13	memData[19]	MEM_DATA19	I / O	SRAM / FLASH Data
T10	W13	memData[20]	MEM_DATA20	I / O	SRAM / FLASH Data
M9	R12	memData[21]	MEM_DATA21	I / O	SRAM / FLASH Data
P9	U12	memData[22]	MEM_DATA22	I / O	SRAM / FLASH Data
R10	V13	memData[23]	MEM_DATA23	I / O	SRAM / FLASH Data
N9	T12	memData[24]	MEM_DATA24	I / O	SRAM / FLASH Data
T9	W12	memData[25]	MEM_DATA25	I / O	SRAM / FLASH Data
R9	V12	memData[26]	MEM_DATA26	I / O	SRAM / FLASH Data
M8	R11	memData[27]	MEM_DATA27	I / O	SRAM / FLASH Data
A5	D8	memData[28]	MEM_DATA28	I / O	SRAM / FLASH Data
C4	F7	memData[29]	MEM_DATA29	I / O	SRAM / FLASH Data
L3	P6	memData[30]	MEM_DATA30	I / O	SRAM / FLASH Data
M1	R4	memData[31]	MEM_DATA31	I / O	SRAM / FLASH Data
N2	T5	flashHiCeN	FLASH_HCE_N	O	Flash Chip Enable (high chip)
M3	R6	flashLoCeN	FLASH_LCE_N	O	Flash Chip Enable (low chip)
L4	P7	flashWeN	FLASH_WE_N	O	Flash Write Enable
N1	T4	flashOeN	FLASH_OE_N	O	Flash Output Enable
N11	T14	sramCeN	SRAM_CE_N	O	SRAM Chip Enable
T14	W17	sramBsN[0]	SRBS0_N	O	SRAM Byte Select 0
R13	V16	sramBsN[1]	SRBS1_N	O	SRAM Byte Select 1
T12	W15	sramBsN[2]	SRBS2_N	O	SRAM Byte Select 2
T13	W16	sramBsN[3]	SRBS3_N	O	SRAM Byte Select 3
R12	V15	sramWeN	SRAM_WE_N	O	SRAM Write Enable
P11	U14	sramOeN	SRAM_OE_N	O	SRAM Output Enable
B7	E10	ledOut[0]	LED0	O	Drives LED 0
C7	F10	ledOut[1]	LED1	O	Drives LED 1
P5	U8	ledOut[2]	LED2	O	Drives LED 2
T2	W5	ledOut[3]	LED3	O	Drives LED 3
P4	U7	ledOut[4]	LED4	O	Drives LED 4
R3	V6	ledOut[5]	LED5	O	Drives LED 5
P2	U5	ledOut[6]	LED6	O	Drives LED 6
P1	U4	ledOut[7]	LED7	O	Drives LED 7
R1	V4	ledOut[8]	LED8	O	Drives LED 8
R2	V5	ledOut[9]	LED9	O	Drives LED 9
A15	D18	switchIn[0]	SWITCH0	I	Switch Input 0
A14	D17	switchIn[1]	SWITCH1	I	Switch Input 1
B14	E17	switchIn[2]	SWITCH2	I	Switch Input 2
C13	F16	switchIn[3]	SWITCH3	I	Switch Input 3
A12	D15	switchIn[4]	SWITCH4	I	Switch Input 4
D11	G14	switchIn[5]	SWITCH5	I	Switch Input 5
B11	E14	switchIn[6]	SWITCH6	I	Switch Input 6
C11	F14	switchIn[7]	SWITCH7	I	Switch Input 7
D10	G13	switchIn[8]	SWITCH8	I	Switch Input 8
A11	D14	switchIn[9]	SWITCH9	I	Switch Input 9
Unused	Unused	gpio0[9:0]	Unused	I / O	General Purpose IO

N/A	P3	gpio0[10]	IOS_P5 (unused)	I / O	General Purpose IO
N16	T19	gpio0[11]	DIFFB2PRX	I / O	General Purpose IO
P16	U19	gpio0[12]	DIFFB2NRX	I / O	General Purpose IO
J14	M17	gpio0[13]	DIFFA1P	I / O	General Purpose IO
K15	N18	gpio0[14]	DIFFA1N	I / O	General Purpose IO
N/A	K1	gpio0[15]	IOS_P6 (unused)	I / O	General Purpose IO
L15	P18	gpio0[16]	DIFFA2P	I / O	General Purpose IO
L14	P17	gpio0[17]	DIFFA2N	I / O	General Purpose IO
L16	P19	gpio0[18]	DIFFB1P	I / O	General Purpose IO
M16	R19	gpio0[19]	DIFFB1N	I / O	General Purpose IO
Unused	Unused	gpio0[31:20]	Unused	I / O	General Purpose IO
A2	D5	gpio1[0]	GPIOA_0	I / O	General Purpose IO
A3	D6	gpio1[1]	GPIOA_1	I / O	General Purpose IO
A7	D10	gpio1[2]	GPIOA_2	I / O	General Purpose IO
D7	G10	gpio1[3]	GPIOA_3	I / O	General Purpose IO
D8	G11	gpio1[4]	GPIOA_4	I / O	General Purpose IO
B8	E11	gpio1[5]	GPIOA_5	I / O	General Purpose IO
A8	D11	gpio1[6]	GPIOA_6	I / O	General Purpose IO
C8	F11	gpio1[7]	GPIOA_7	I / O	General Purpose IO
E8	H11	gpio1[8]	GPIOA_8	I / O	General Purpose IO
C9	F12	gpio1[9]	GPIOA_9	I / O	General Purpose IO
B9	E12	gpio1[10]	GPIOA_10	I / O	General Purpose IO
A9	D12	gpio1[11]	GPIOA_11	I / O	General Purpose IO
D9	G12	gpio1[12]	GPIOA_12	I / O	General Purpose IO
E9	H12	gpio1[13]	GPIOA_13	I / O	General Purpose IO
C10	F13	gpio1[14]	GPIOA_14	I / O	General Purpose IO
A10	D13	gpio1[15]	GPIOA_15	I / O	General Purpose IO
B10	E13	gpio1[16]	GPIOA_16	I / O	General Purpose IO
B13	E16	gpio1[17]	GPIOA_17	I / O	General Purpose IO
A13	D16	gpio1[18]	GPIOA_18	I / O	General Purpose IO
C12	F15	gpio1[19]	GPIOA_19	I / O	General Purpose IO
B12	E15	gpio1[20]	GPIOA_20	I / O	General Purpose IO
K4	N7	gpio1[21]	GPIOA_21	I / O	General Purpose IO
K2	N5	gpio1[22]	GPIOA_22	I / O	General Purpose IO
J4	M7	gpio1[23]	GPIOA_23	I / O	General Purpose IO
G1	K4	gpio1[24]	GPIOA_24	I / O	General Purpose IO
G2	K5	gpio1[25]	GPIOA_25	I / O	General Purpose IO
G3	K6	gpio1[26]	GPIOA_26	I / O	General Purpose IO
F4	J7	gpio1[27]	GPIOA_27	I / O	General Purpose IO
F3	J6	gpio1[28]	GPIOA_28	I / O	General Purpose IO
F1	J4	gpio1[29]	GPIOA_29	I / O	General Purpose IO
E1	H4	gpio1[30]	GPIOA_30	I / O	General Purpose IO
B2	E5	gpio1[31]	GPIOA_31	I / O	General Purpose IO
E13	H16	gpio2[0]	GPIOB_0	I / O	2.5V IO
C16	F19	gpio2[1]	GPIOB_1	I / O	LVDS IO Transmit Pair with termination
D16	G19	gpio2[2]	GPIOB_2	I / O	
G15	K18	gpio2[3]	GPIOB_3	I / O	LVDS IO Transmit Pair with termination
G16	K19	gpio2[4]	GPIOB_4	I / O	
E16	J18	gpio2[5]	GPIOBRX_5	I / O	LVDS IO Receive Pair with termination
F16	K17	gpio2[6]	GPIOBRX_6	I / O	

B15	E18	gpio2[7]	GPIOB_7	I / O	LVDS IO Transmit Pair with termination
B16	E19	gpio2[8]	GPIOB_8	I / O	
D14	G17	gpio2[9]	GPIOB_9	I / O	LVDS IO Transmit Pair with termination
C15	F18	gpio2[10]	GPIOB_10	I / O	
E14	H17	gpio2[11]	GPIOBRX_11	I / O	LVDS IO Receive Pair with termination
H15	L18	gpio2[12]	GPIOBRX_12	I / O	
E15	H18	gpio2[13]	GPIOB_13	I / O	LVDS IO Transmit Pair with termination
F14	J17	gpio2[14]	GPIOB_14	I / O	
F15	J18	gpio2[15]	GPIOB_15	I / O	LVDS IO Transmit Pair with termination
G14	K17	gpio2[16]	GPIOB_16	I / O	
J15	M18	gpio2[17]	GPIOBRX_17	I / O	LVDS IO Receive Pair with termination
K14	N17	gpio2[18]	GPIOBRX_18	I / O	
G13	K16	gpio2[19]	GPIOB_19	I / O	LVDS IO Transmit Pair with termination
H12	L15	gpio2[20]	GPIOB_20	I / O	
H13	L16	gpio2[21]	GPIOB_21	I / O	LVDS IO Transmit Pair with termination
H16	L19	gpio2[22]	GPIOB_22	I / O	
J13	M16	gpio2[23]	GPIOBRX_23	I / O	LVDS IO Receive Pair with termination
H14	L17	gpio2[24]	GPIOBRX_24	I / O	
J16	M19	gpio2[25]	GPIOB_25	I / O	LVDS IO Transmit Pair with termination
K16	N19	gpio2[26]	GPIOB_26	I / O	
M14	R17	gpio2[27]	GPIOB_27	I / O	LVDS IO Transmit Pair with termination
L13	P16	gpio2[28]	GPIOB_28	I / O	
M15	R18	gpio2[29]	GPIOBRX_29	I / O	LVDS IO Receive Pair with termination
N15	T18	gpio2[30]	GPIOBRX_30	I / O	
F13	J16	gpio2[31]	GPIOB_31	I / O	2.5V IO
N/A	A6	P5_ODD_35_5[0]	GPIOC_1	I / O	General Purpose IO
N/A	A7	P5_ODD_35_5[1]	GPIOC_3	I / O	General Purpose IO
N/A	A10	P5_ODD_35_5[2]	GPIOC_5	I / O	General Purpose IO
N/A	A11	P5_ODD_35_5[3]	GPIOC_7	I / O	General Purpose IO
N/A	B6	P5_ODD_35_5[4]	GPIOC_9	I / O	General Purpose IO
N/A	B7	P5_ODD_35_5[5]	GPIOC_11	I / O	General Purpose IO
N/A	B10	P5_ODD_35_5[6]	GPIOC_13	I / O	General Purpose IO
N/A	AB17	P5_ODD_35_5[7]	GPIOC_15	I / O	General Purpose IO
N/A	AB16	P5_ODD_35_5[8]	GPIOC_17	I / O	General Purpose IO
N/A	AB13	P5_ODD_35_5[9]	GPIOC_19	I / O	General Purpose IO
N/A	AB12	P5_ODD_35_5[10]	GPIOC_21	I / O	General Purpose IO
N/A	AA16	P5_ODD_35_5[11]	GPIOC_23	I / O	General Purpose IO
N/A	Y2	P5_ODD_35_5[12]	GPIOC_25	I / O	General Purpose IO
N/A	U1	P5_ODD_35_5[13]	GPIOC_27	I / O	General Purpose IO
N13	T16	P5_ODD_35_5[14]	GPIOC_29	I / O	General Purpose IO
N/A	M3	P5_ODD_35_5[15]	GPIOC_31	I / O	General Purpose IO
N/A	A12	P5_EVEN_34_4[0]	GPIOC_0	I / O	General Purpose IO
N/A	A13	P5_EVEN_34_4[1]	GPIOC_2	I / O	General Purpose IO
N/A	A16	P5_EVEN_34_4[2]	GPIOC_4	I / O	General Purpose IO
N/A	A17	P5_EVEN_34_4[3]	GPIOC_6	I / O	General Purpose IO
N/A	B13	P5_EVEN_34_4[4]	GPIOC_8	I / O	General Purpose IO

N/A	B16	P5_EVEN_34_4[5]	GPIOC_10	I / O	General Purpose IO
N/A	B17	P5_EVEN_34_4[6]	GPIOC_12	I / O	General Purpose IO
N/A	AB9	P5_EVEN_34_4[7]	GPIOC_14	I / O	General Purpose IO
N/A	AB8	P5_EVEN_34_4[8]	GPIOC_16	I / O	General Purpose IO
N/A	AB7	P5_EVEN_34_4[9]	GPIOC_18	I / O	General Purpose IO
N/A	AB6	P5_EVEN_34_4[10]	GPIOC_20	I / O	General Purpose IO
N/A	AA7	P5_EVEN_34_4[11]	GPIOC_22	I / O	General Purpose IO
N/A	AA6	P5_EVEN_34_4[12]	GPIOC_24	I / O	General Purpose IO
N/A	W2	P5_EVEN_34_4[13]	GPIOC_26	I / O	General Purpose IO
N/A	T1	P5_EVEN_34_4[13]	GPIOC_28	I / O	General Purpose IO
M13	R16	P5_EVEN_34_4[15]	GPIOC_30	I / O	General Purpose IO
N/A	J21	P5_38_37[0]	DIFFC1P	I / O	General Purpose IO
N/A	J22	P5_38_37[1]	DIFFC1N	I / O	General Purpose IO
N/A	K22	P5_40_39[0]	DIFFC2P	I / O	General Purpose IO
N/A	K21	P5_40_39[1]	DIFFC2N	I / O	General Purpose IO

Table A.2- M1AGL FPGA Signals

Board Schematics

For detailed Schematic, please refer to the **"M1AGL Schematics.pdf"** in the **"Dev Kit Documentation"** folder.

There are 7 pages to the Schematics, titled as follows:

1. POWER SUPPLIES
2. PWR, HDRS, LVDS
3. SRAM & FLASH
4. CLK, RESET, USB, ETC.
5. FPGA
6. FPGA
7. FLASHPRO3

The schematic number is SOC-AGL-S-002. Please reference this number and the schematic sheet when contacting support@socsolutions.com for M1AGL Development Board and schematic support.

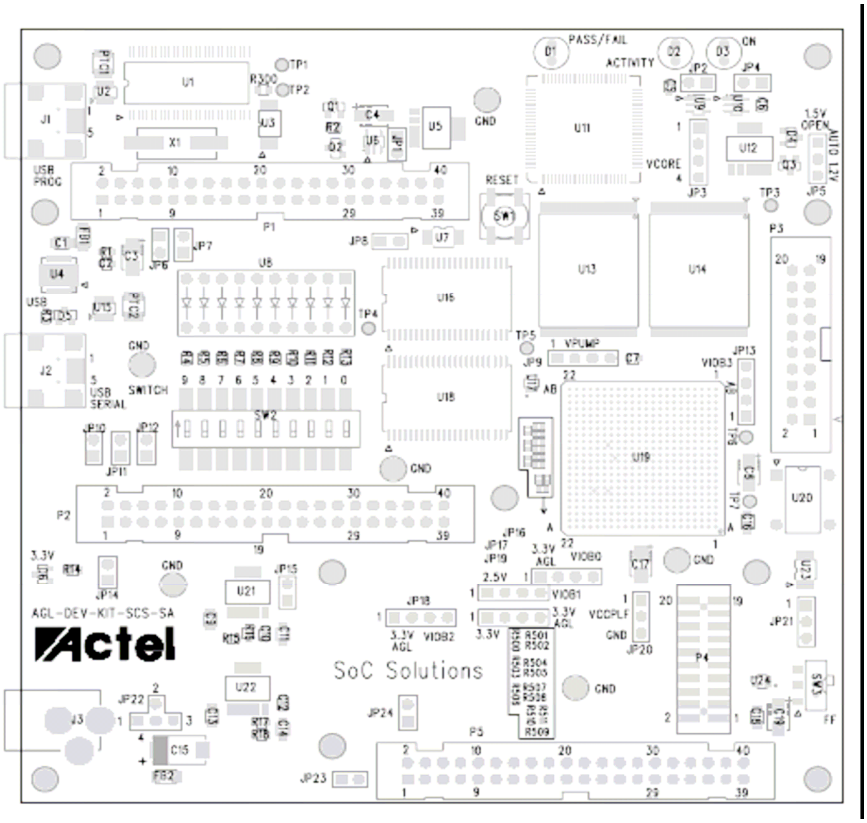
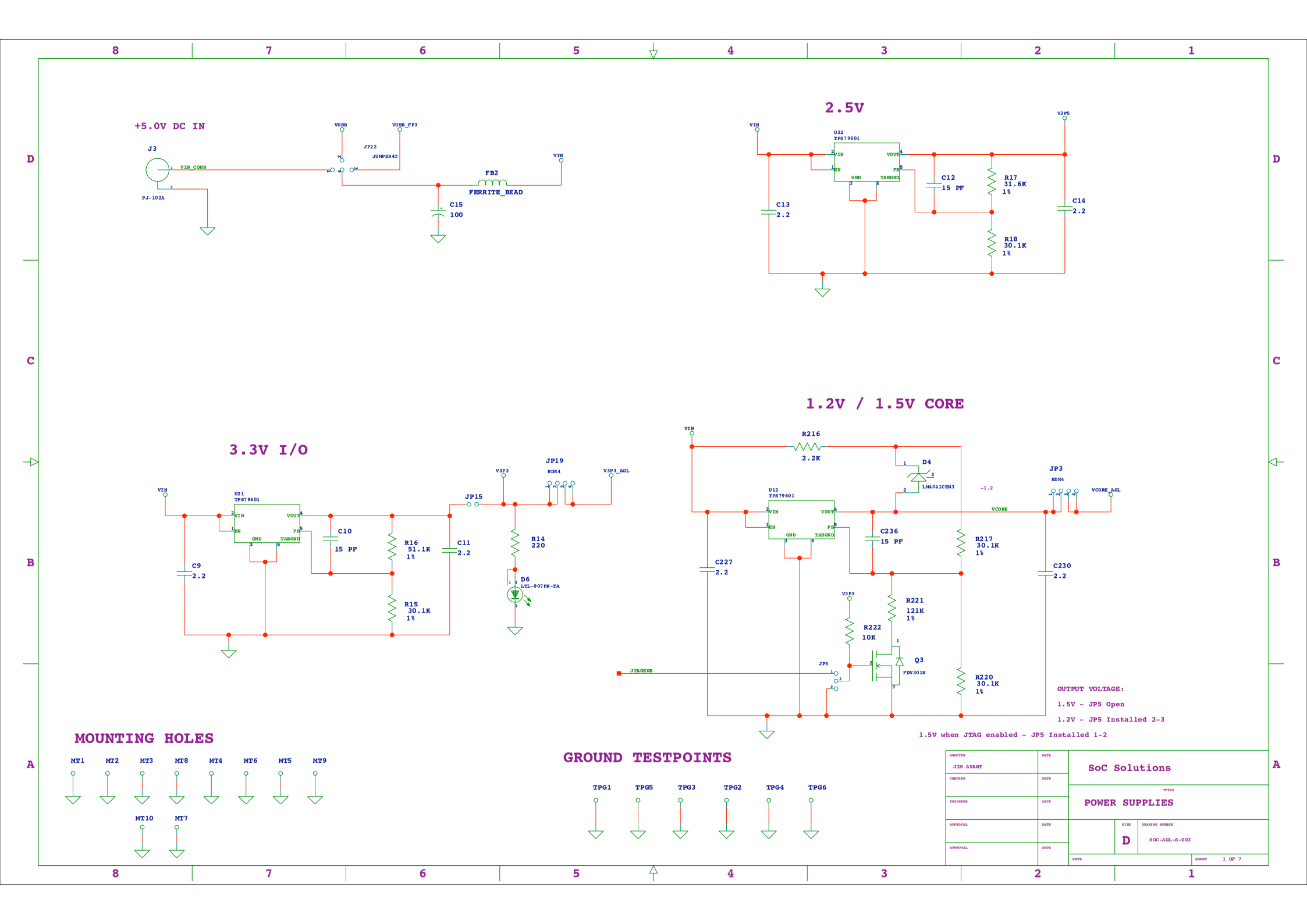
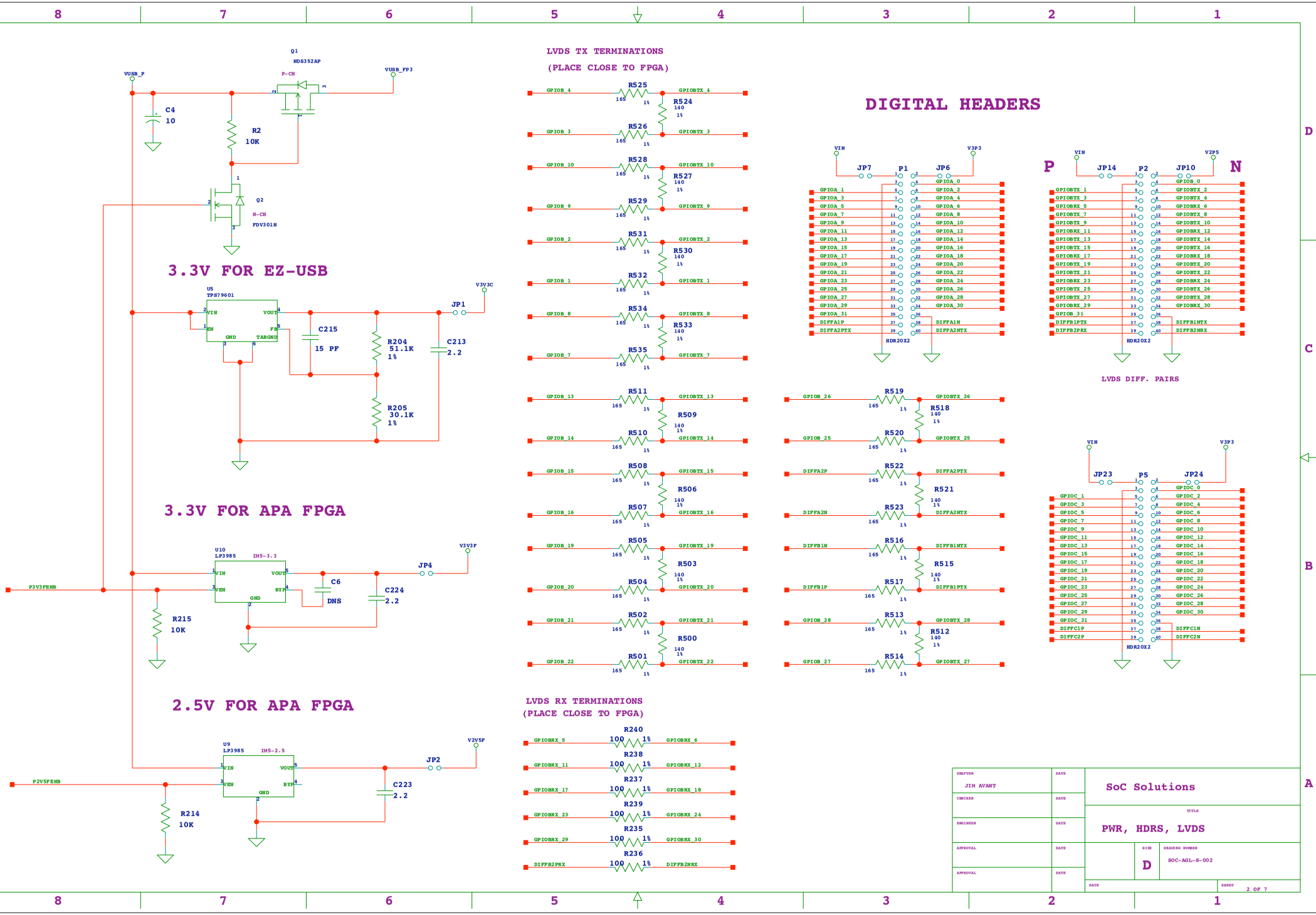
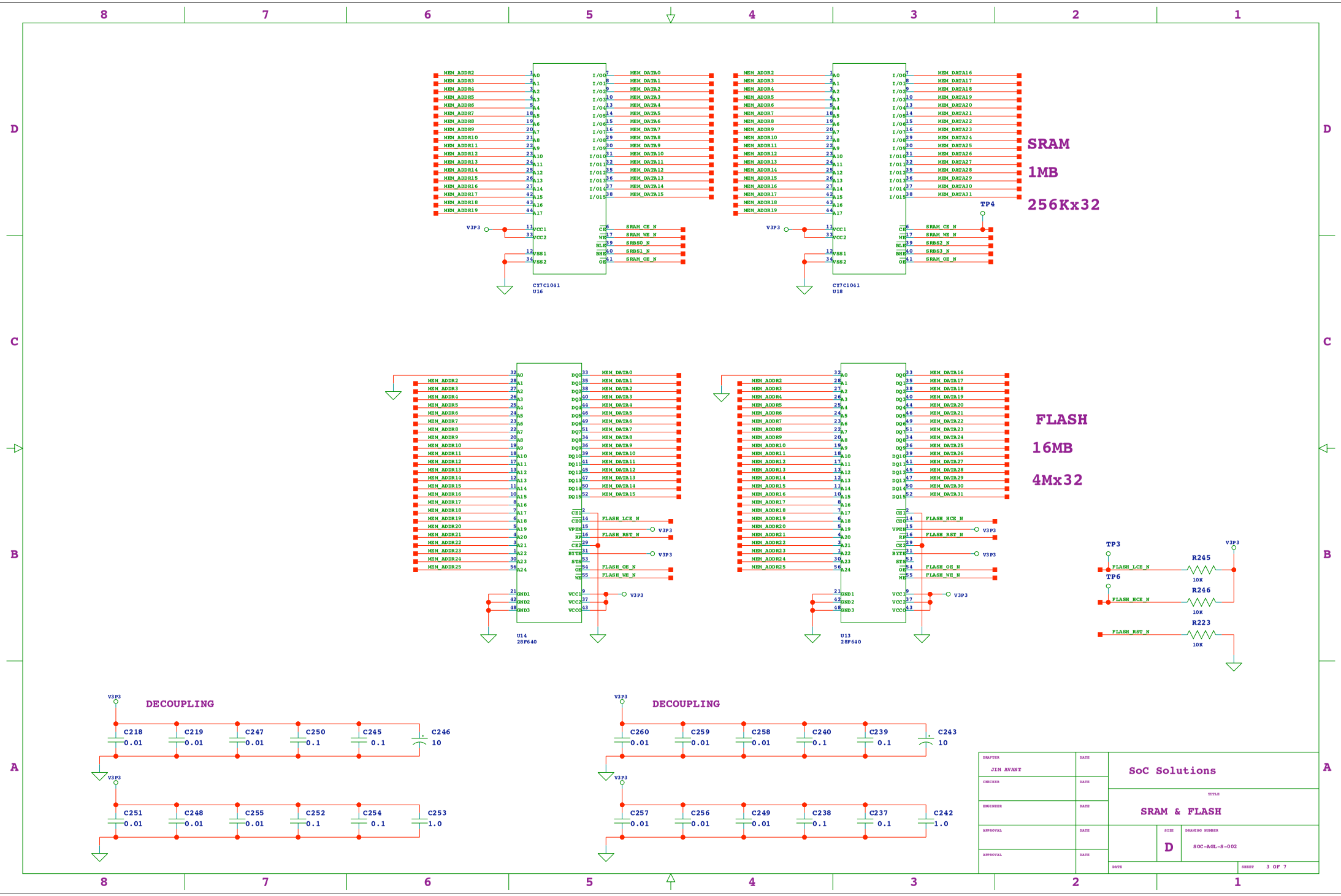
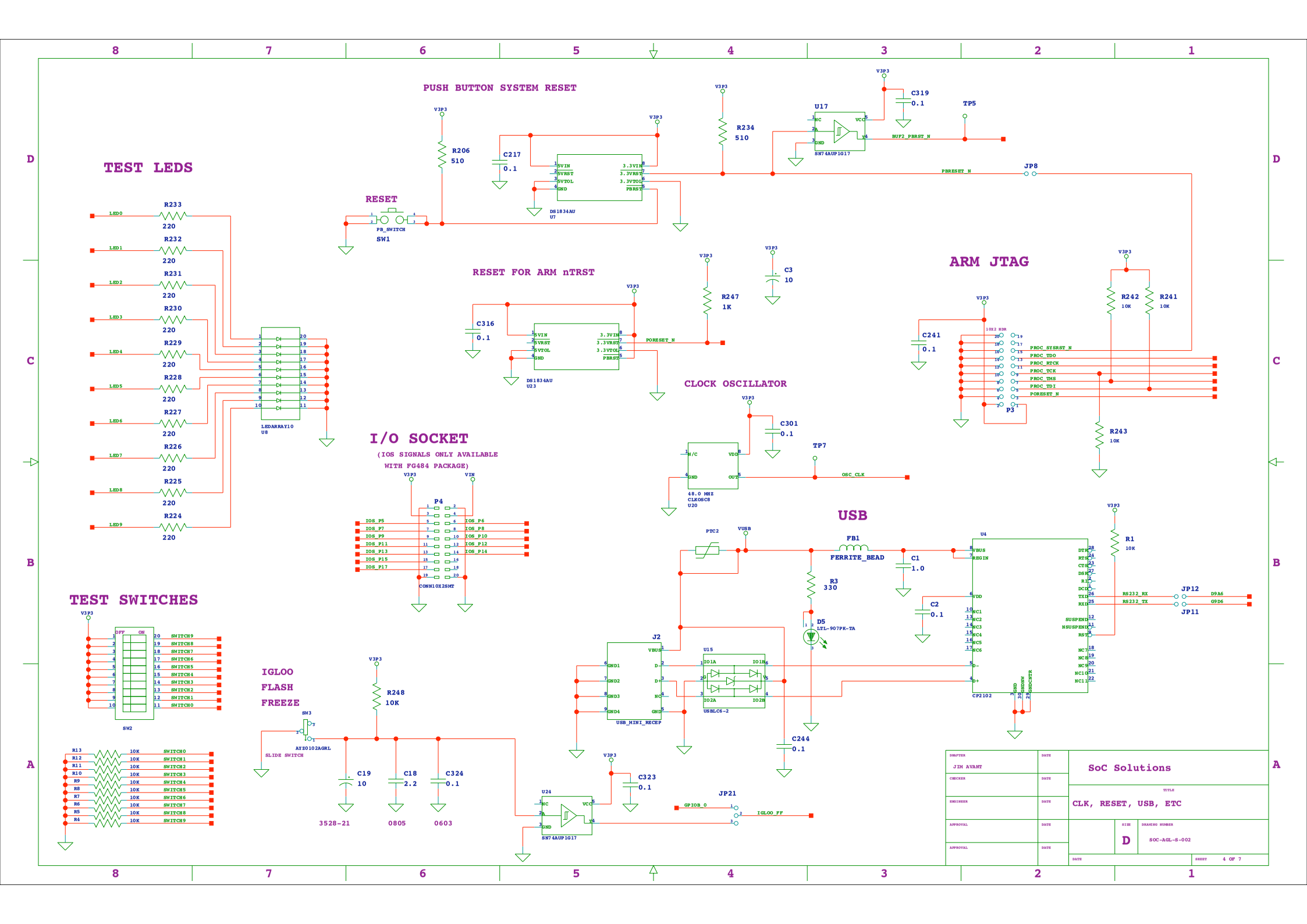


Figure B.1 - M1AGL Development Board Top Assembly Drawing

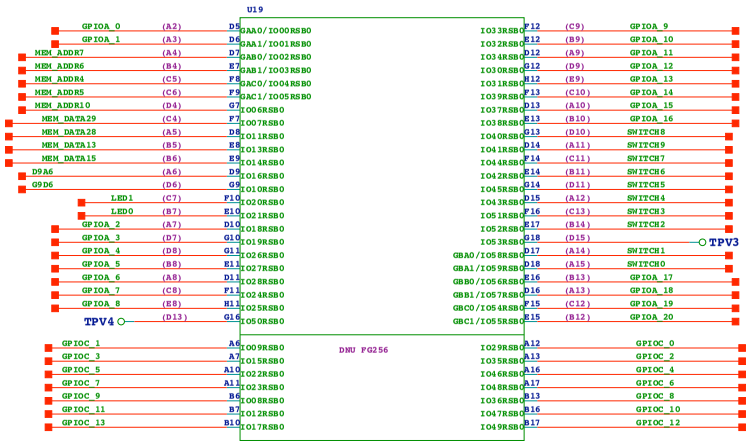

+5.0V DC IN
J3
VIN_CONN
PJ-102A
VUSB
VUSB_FP3
JP22
JUMPER4T
FB2
FERRITE_BEAD
VIN
C15
100
2.5V
VIN
U22
TPS79601
VIN
VOUT
EN
FB
GND
TABGND
V2P5
C12
15 PF
R17
31.6K
1%
C13
2.2
C14
2.2
R18
30.1K
1%
3.3V I/O
VIN
U21
TPS79601
JP15
V3P3
JP19
HDR4
V3P3_AGL
C10
15 PF
R16
51.1K
1%
C11
2.2
R14
220
C9
2.2
R15
30.1K
1%
D6
LTL-907PK-TA
1.2V / 1.5V CORE
VIN
R216
2.2K
D4
LM4041CEM3
-1.2
JP3
HDR4
VCORE_AGL
U12
TPS79601
VCORE
C236
15 PF
R217
30.1K
1%
C227
2.2
C230
2.2
V3P3
R221
121K
1%
R222
10K
Q3
FDV301N
JP5
JTAGENB
R220
30.1K
1%
OUTPUT VOLTAGE:
1.5V - JP5 Open
1.2V - JP5 Installed 2-3
1.5V when JTAG enabled - JP5 Installed 1-2
MOUNTING HOLES
MT1
MT2
MT3
MT8
MT4
MT6
MT5
MT9
MT10
MT7
GROUND TESTPOINTS
TPG1
TPG5
TPG3
TPG2
TPG4
TPG6
DRAFTER
JIM AVANT
DATE
CHECKER
ENGINEER
APPROVAL
SoC Solutions
TITLE
POWER SUPPLIES
SIZE
D
DRAWING NUMBER
SOC-AGL-S-002
SHEET
1 OF 7


3.3V FOR EZ-USB
3.3V FOR APA FPGA
2.5V FOR APA FPGA
LVDS TX TERMINATIONS
(PLACE CLOSE TO FPGA)
LVDS RX TERMINATIONS
(PLACE CLOSE TO FPGA)
DIGITAL HEADERS
LVDS DIFF. PAIRS
Q1
NDS352AP
P-CH
Q2
N-CH
FDV301N
VUSB_P
VUSB_FP3
C4
10
R2
10K
U5
TPS79601
JP1
V3V3C
C215
15 PF
R204
51.1K
1%
R205
30.1K
1%
C213
2.2
U10
LP3985
IM5-3.3
JP4
V3V3F
P3V3FENB
R215
10K
C6
DNS
C224
2.2
U9
LP3985
IM5-2.5
JP2
V2V5F
P2V5FENB
R214
10K
C223
2.2
R525
R524
R526
R528
R527
R529
R531
R530
R532
R534
R533
R535
R511
R509
R510
R508
R506
R507
R505
R503
R504
R502
R500
R501
R240
R238
R237
R239
R235
R236
JP7
P1
JP6
HDR20X2
JP14
P2
JP10
P
N
JP23
P5
JP24
R519
R518
R520
R522
R521
R523
R516
R515
R517
R513
R512
R514
SoC Solutions
PWR, HDRS, LVDS
JIM AVANT
SOC-AGL-S-002
D
2 OF 7

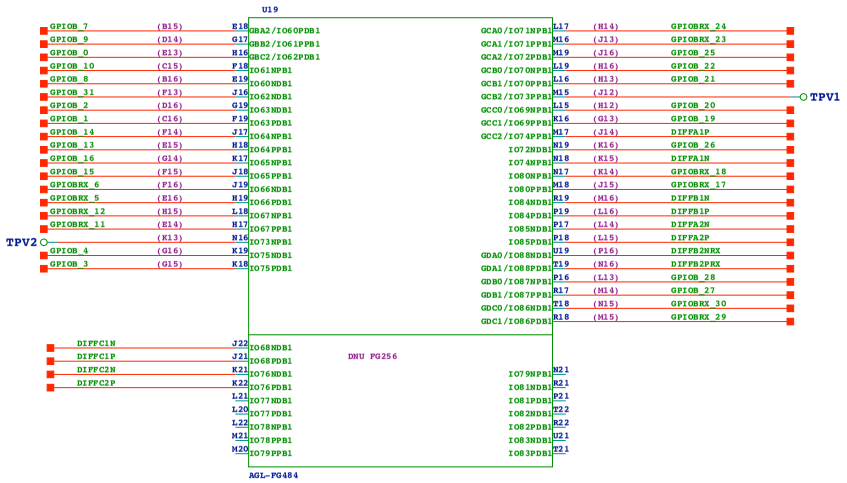

SRAM
1MB
256Kx32
CY7C1041
U16
CY7C1041
U18
TP4
FLASH
16MB
4Mx32
U14
28F640
U13
28F640
TP3
TP6
R245
R246
R223
10K
DECOUPLING
C218 0.01
C219 0.01
C247 0.01
C250 0.1
C245 0.1
C246 10
C251 0.01
C248 0.01
C255 0.01
C252 0.1
C254 0.1
C253 1.0
DECOUPLING
C260 0.01
C259 0.01
C258 0.01
C240 0.1
C239 0.1
C243 10
C257 0.01
C256 0.01
C249 0.01
C238 0.1
C237 0.1
C242 1.0
SoC Solutions
SRAM & FLASH
JIM AVANT
SOC-AGL-S-002
D
3 OF 7


PUSH BUTTON SYSTEM RESET
RESET
PB_SWITCH
SW1
R206
510
C217
0.1
DS1834AU
U7
R234
510
U17
SN74AUP1G17
C319
0.1
TP5
BUF2_PBRST_N
PBRESET_N
JP8
TEST LEDS
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
R233
R232
R231
R230
R229
R228
R227
R226
R225
R224
220
LEDARRAY10
U8
RESET FOR ARM nTRST
C316
0.1
DS1834AU
U23
R247
1K
C3
10
PORESET_N
ARM JTAG
C241
0.1
10X2 HDR
P3
PROC_SYSRST_N
PROC_TDO
PROC_RTCK
PROC_TCK
PROC_TMS
PROC_TDI
PORESET_N
R242
10K
R241
10K
R243
10K
CLOCK OSCILLATOR
C301
0.1
TP7
OSC_CLK
48.0 MHZ
CLKOSC8
U20
I/O SOCKET
(IOS SIGNALS ONLY AVAILABLE
WITH FG484 PACKAGE)
P4
IOS_P5
IOS_P7
IOS_P9
IOS_P11
IOS_P13
IOS_P15
IOS_P17
IOS_P6
IOS_P8
IOS_P10
IOS_P12
IOS_P14
CONN10X2SMT
USB
PTC2
VUSB
FB1
FERRITE_BEAD
C1
1.0
C2
0.1
R3
330
D5
LTL-907PK-TA
U4
CP2102
R1
10K
JP12
JP11
RS232_RX
RS232_TX
D9A6
G9D6
J2
USB_MINI_RECEP
U15
USBLC6-2
C244
0.1
TEST SWITCHES
SW2
SWITCH0
SWITCH1
SWITCH2
SWITCH3
SWITCH4
SWITCH5
SWITCH6
SWITCH7
SWITCH8
SWITCH9
R13
R12
R11
R10
R9
R8
R7
R6
R5
R4
10K
IGLOO
FLASH
FREEZE
SW3
AYZ0102AGRL
SLIDE SWITCH
R248
10K
C19
10
3528-21
C18
2.2
0805
C324
0.1
0603
U24
SN74AUP1G17
C323
0.1
JP21
GPIOB_0
IGLOO_FF
SoC Solutions
CLK, RESET, USB, ETC
JIM AVANT
SOC-AGL-S-002
D
4 OF 7

BANK 0



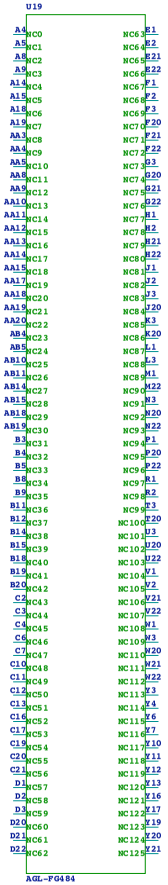
BANK 1



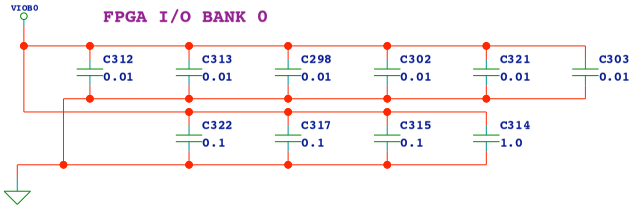
NOTE: BGA BALL NUMBERS SHOWN FOR FG484 PACKAGE WITH FG256 BALL NUMBERS IN PARENTHESES.

NOTE: BOTTOM PORTION OF FPGA SYMBOLS MARKED "DNU FG256" SHOWS BALLS THAT ARE NOT AVAILABLE WHEN THE FG256 PACKAGE IS USED.

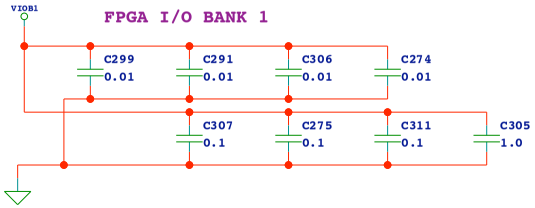
NC

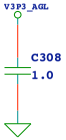


FPGA I/O BANK 0

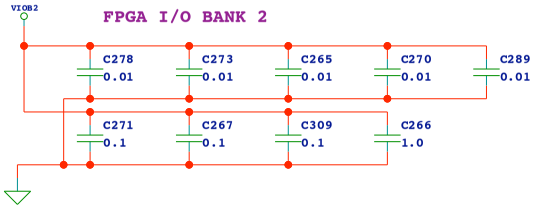


FPGA I/O BANK 1

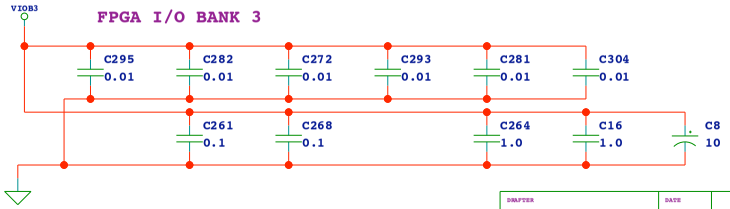




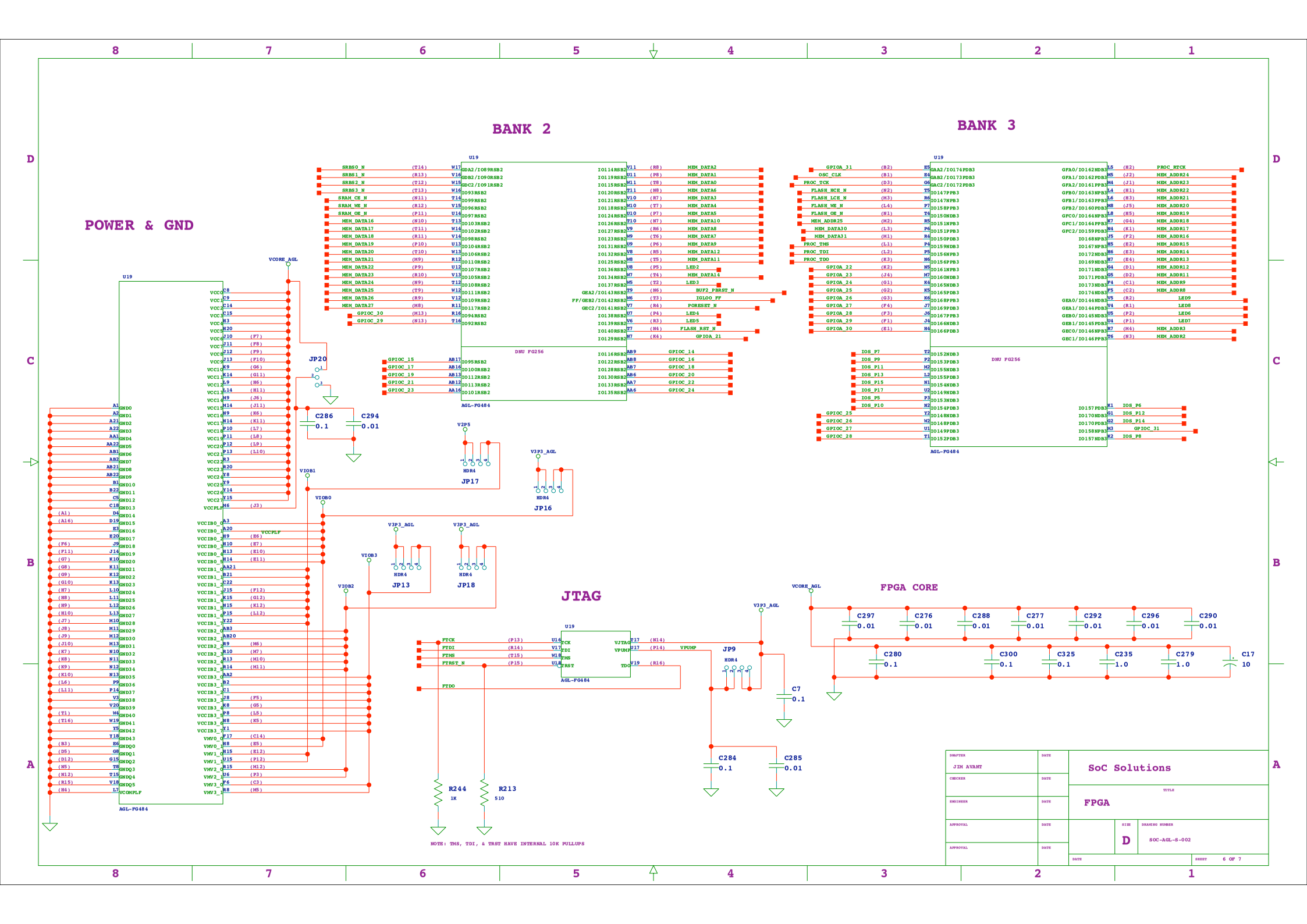
FPGA I/O BANK 2



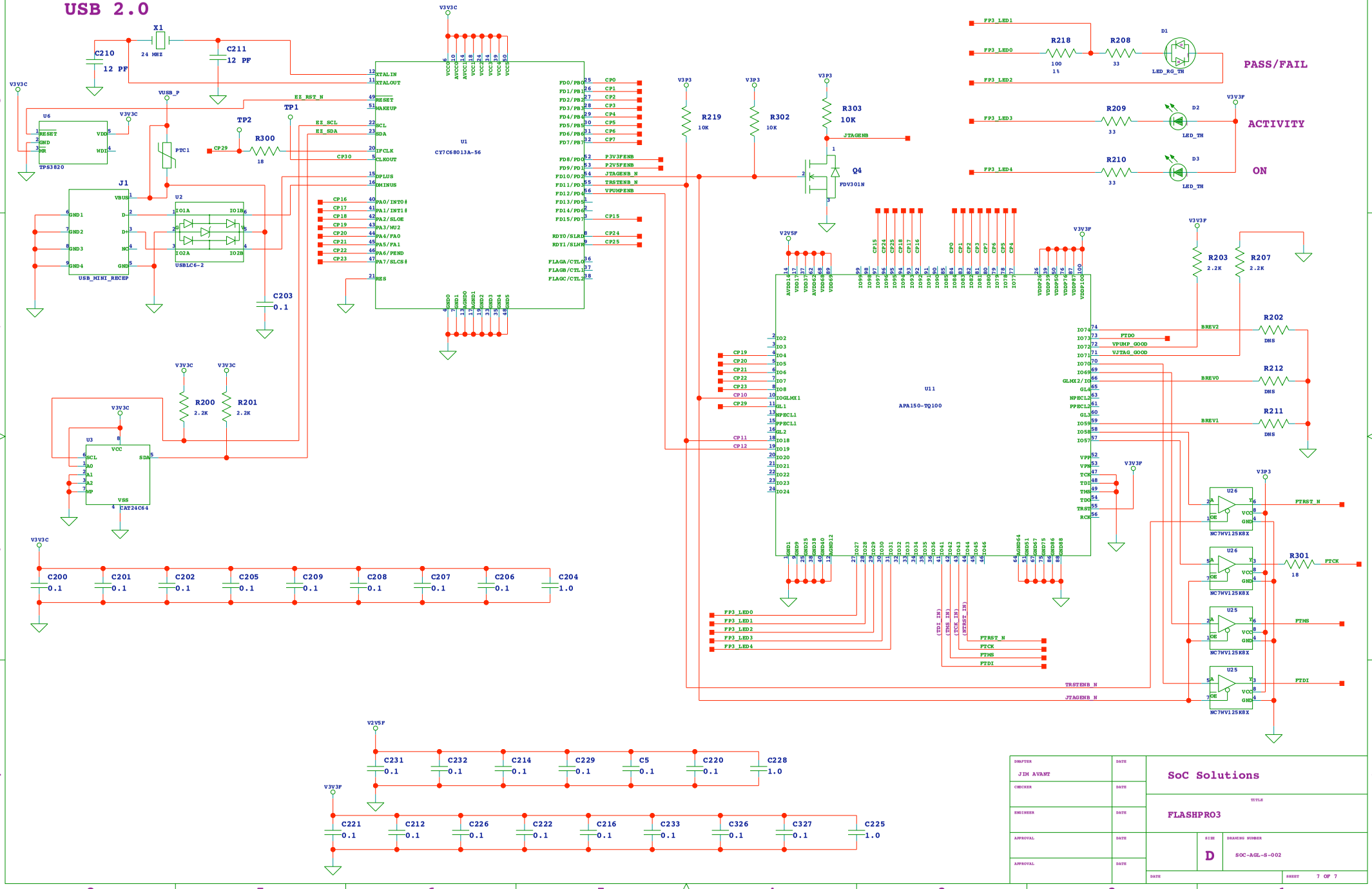
FPGA I/O BANK 3



DRAFTER: JIM AVANT	DATE	SoC Solutions	
CHECKER	DATE	TITLE: FPGA	
ENGINEER	DATE		
APPROVAL	DATE	SIZE: D	DRAWING NUMBER: SOC-AGL-S-002
APPROVAL	DATE	DATE	SHEET: 5 OF 7


BANK 2
BANK 3
POWER & GND
JTAG
FPGA CORE
U19
AGL-FG484
DNU FG256
JP20
JP17
JP16
JP13
JP18
JP9
HDR4
C286
0.1
C294
0.01
C7
0.1
C284
0.1
C285
0.01
R244
1K
R213
510
C297
C276
C288
C277
C292
C296
C290
C280
C300
C325
C235
C279
C17
10
VCORE_AGL
V3P3_AGL
VPUMP
VCCPLF
VIOB1
VIOB0
VIOB3
VIOB2
NOTE: TMS, TDI, & TRST HAVE INTERNAL 10K PULLUPS
SoC Solutions
FPGA
JIM AVANT
SOC-AGL-S-002
SIZE
D
SHEET 6 OF 7

USB 2.0


X1
24 MHZ
C210
12 PF
C211
12 PF
U6
TPS3820
J1
USB_MINI_RECEP
U2
USBLC6-2
PTC1
TP1
TP2
R300
18
C203
0.1
U1
CY7C68013A-56
R219
10K
R302
10K
R303
10K
JTAGENB
Q4
FDV301N
U11
APA150-TQ100
R200
2.2K
R201
2.2K
U3
CAT24C64
R203
2.2K
R207
2.2K
R202
DNS
R212
DNS
R211
DNS
R301
18
U26
U25
NC7WV125K8X
PASS/FAIL
ACTIVITY
ON
R218
100
1%
R208
33
R209
33
R210
33
D1
D2
D3
LED_RG_TH
LED_TH
C200
C201
C202
C205
C209
C208
C207
C206
C204
1.0
C231
C232
C214
C229
C5
C220
C228
C221
C212
C226
C222
C216
C233
C326
C327
C225
SoC Solutions
FLASHPRO3
JIM AVANT
SOC-AGL-S-002
D
7 OF 7

Appendice F

Report di prestazioni della Smart Wacth Dog fase *HW*

Sorgente 26: Report di Synthesis

```
--------------------------------------------------------------------------------
Report for cell watchdog.aramis
                                                  Cell usage:
                                           cell   count      area    count*area
                                         CLKBUF       3       0.0           0.0
            ComparisonSignaturePlusGonIdenIndex         1      30.0          30.0
                                            GND       1       0.0           0.0
                                          INBUF       2       0.0           0.0
                                   IndexCounter       1      43.0          43.0
                                          NOR2B       1       1.0           1.0
                                         OUTBUF       4       0.0           0.0
                                 ProgramControl       1      44.0          44.0
                               ProgramInterface       1    2889.0        2889.0
                                  RebootCounter       1     412.0         412.0
                                            VCC       1       0.0           0.0
                                  WatchdogTimer       1     110.0         110.0
                                      memoryrom       1    1359.0        1359.0
                                serialinterface       1     323.0         323.0

                                          TOTAL      20                  5211.0
--------------------------------------------------------------------------------
Report for cell memoryrom.netlist
     Instance path: MR
                                                  Cell usage:
                                           cell   count      area    count*area
                                           AND2      14       1.0          14.0
                                            AO1      15       1.0          15.0
                                           AO1A       4       1.0           4.0
                                           AO1B       7       1.0           7.0
                                           AO1C      13       1.0          13.0
                                           AO1D      10       1.0          10.0
                                           AOI1      14       1.0          14.0
                                          AOI1B      40       1.0          40.0
                                         DFN1C0      41       1.0          41.0
                                            GND       1       0.0           0.0
                                            MX2       2       1.0           2.0
                                          NAND2      10       1.0          10.0
```

NOR2	123	1.0	123.0
NOR2A	72	1.0	72.0
NOR2B	58	1.0	58.0
NOR3	3	1.0	3.0
NOR3A	5	1.0	5.0
NOR3B	15	1.0	15.0
NOR3C	203	1.0	203.0
OA1	28	1.0	28.0
OA1A	53	1.0	53.0
OA1B	1	1.0	1.0
OA1C	3	1.0	3.0
OAI1	8	1.0	8.0
OR2	128	1.0	128.0
OR2A	155	1.0	155.0
OR2B	89	1.0	89.0
OR3	76	1.0	76.0
OR3A	77	1.0	77.0
OR3B	63	1.0	63.0
OR3C	16	1.0	16.0
VCC	1	0.0	0.0
XNOR2	5	1.0	5.0
XO1A	1	1.0	1.0
XOR2	7	1.0	7.0
TOTAL	1361		1359.0

Report **for** cell ProgramInterface.netlist
Instance path: PI

Cell usage:

cell	count	area	count*area
AND2	9	1.0	9.0
AND3	2	1.0	2.0
AO1	7	1.0	7.0
AO18	2	1.0	2.0
AO1A	7	1.0	7.0
AO1B	33	1.0	33.0
AO1C	10	1.0	10.0
AO1D	6	1.0	6.0
AOI1	10	1.0	10.0
AOI1B	11	1.0	11.0
AX1	5	1.0	5.0
AX1A	1	1.0	1.0
AX1B	5	1.0	5.0
AX1C	11	1.0	11.0
AX1D	3	1.0	3.0
AX1E	3	1.0	3.0
AXO1	1	1.0	1.0
AXO2	1	1.0	1.0
AXOI5	3	1.0	3.0
CLKINT	3	0.0	0.0
DFN1C0	35	1.0	35.0
DFN1C1	8	1.0	8.0
DFN1E0C0	551	1.0	551.0
DFN1E0C1	1	1.0	1.0
DFN1E0P0	3	1.0	3.0
DFN1E1C0	351	1.0	351.0
DFN1E1P0	2	1.0	2.0
DFN1P0	2	1.0	2.0
DFN1P1	8	1.0	8.0
GND	1	0.0	0.0
INV	3	1.0	3.0
MAJ3	3	1.0	3.0

MIN3	2	1.0	2.0
MX2	812	1.0	812.0
MX2B	9	1.0	9.0
MX2C	1	1.0	1.0
NAND2	6	1.0	6.0
NOR2	36	1.0	36.0
NOR2A	41	1.0	41.0
NOR2B	53	1.0	53.0
NOR3	11	1.0	11.0
NOR3A	41	1.0	41.0
NOR3B	20	1.0	20.0
NOR3C	19	1.0	19.0
OA1	6	1.0	6.0
OA1A	8	1.0	8.0
OA1C	14	1.0	14.0
OAI1	7	1.0	7.0
OR2	27	1.0	27.0
OR2A	38	1.0	38.0
OR2B	36	1.0	36.0
OR3	75	1.0	75.0
OR3A	88	1.0	88.0
OR3B	97	1.0	97.0
OR3C	15	1.0	15.0
VCC	1	0.0	0.0
XA1	4	1.0	4.0
XA1A	2	1.0	2.0
XA1C	1	1.0	1.0
XAI1	2	1.0	2.0
XAI1A	1	1.0	1.0
XNOR2	32	1.0	32.0
XNOR3	3	1.0	3.0
XO1	1	1.0	1.0
XOR2	55	1.0	55.0
XOR3	4	1.0	4.0
ZOR3	2	1.0	2.0
uartZ1	1	223.0	223.0
TOTAL	2672		2889.0

Report **for** cell uartZ1.netlist
Instance path: PI.UA1

Cell usage:

cell	count	area	count*area
DFN1C0	1	1.0	1.0
GND	1	0.0	0.0
VCC	1	0.0	0.0
clockdividerZ1_txrate_1_1	1	8.0	8.0
clockdividerZ2	1	84.0	84.0
rxunit_Urxunit_1_1	1	62.0	62.0
txunit_Utxunit_1_1	1	68.0	68.0
TOTAL	7		223.0

Report **for** cell rxunit_Urxunit_1_1.netlist
Original Cell name rxunit_Urxunit_1
Instance path: PI.UA1.Urxunit

Cell usage:

cell	count	area	count*area
AND2	1	1.0	1.0
AX1E	1	1.0	1.0
DFN1C0	4	1.0	4.0
DFN1C1	1	1.0	1.0

```
                     DFN1E0C0        1        1.0        1.0
                     DFN1E1C0       18        1.0       18.0
                          GND        1        0.0        0.0
                         MX2A        1        1.0        1.0
                         NOR2        1        1.0        1.0
                        NOR2A        2        1.0        2.0
                        NOR2B       10        1.0       10.0
                         NOR3        2        1.0        2.0
                        NOR3A        1        1.0        1.0
                        NOR3B        6        1.0        6.0
                        NOR3C        3        1.0        3.0
                         OAI1        1        1.0        1.0
                          OR2        1        1.0        1.0
                         OR2A        2        1.0        2.0
                         OR2B        1        1.0        1.0
                         OR3B        1        1.0        1.0
                          VCC        1        0.0        0.0
                         XOR2        4        1.0        4.0

                 TOTAL              64                  62.0
---------------------------------------------------------------------
Report for cell txunit_Utxunit_1_1.netlist
  Original Cell name txunit_Utxunit_1
     Instance path: PI.UA1.Utxunit
                                    Cell usage:
                         cell      count     area     count*area
                          AO1        2        1.0        2.0
                         AO1C        2        1.0        2.0
                       DFN1C0        4        1.0        4.0
                     DFN1E0P0        1        1.0        1.0
                     DFN1E1C0       17        1.0       17.0
                          GND        1        0.0        0.0
                          MX2        4        1.0        4.0
                         MX2B        1        1.0        1.0
                         MX2C        4        1.0        4.0
                         NOR2        3        1.0        3.0
                        NOR2A        3        1.0        3.0
                        NOR2B        2        1.0        2.0
                         NOR3        1        1.0        1.0
                        NOR3A        3        1.0        3.0
                        NOR3B        1        1.0        1.0
                          OA1        1        1.0        1.0
                         OA1B        1        1.0        1.0
                          OR2        2        1.0        2.0
                         OR2A        1        1.0        1.0
                         OR3B        2        1.0        2.0
                          VCC        1        0.0        0.0
                         XA1C        1        1.0        1.0
                         XOR2        4        1.0        4.0
                         XOR3        2        1.0        2.0
               synchroniser_1        1        6.0        6.0

                 TOTAL              65                  68.0
---------------------------------------------------------------------
Report for cell synchroniser_1.netlist
  Original Cell name synchroniser
     Instance path: PI.UA1.Utxunit.sync
                                    Cell usage:
                         cell      count     area     count*area
                       DFN1C0        2        1.0        2.0
                       DFN1P1        1        1.0        1.0
                          GND        1        0.0        0.0
```

```
                    NOR2        1      1.0       1.0
                    OR2A        2      1.0       2.0
                    VCC         1      0.0       0.0

              TOTAL             8                6.0
------------------------------------------------------------
Report for cell clockdividerZ1_txrate_1_1.netlist
  Original Cell name clockdividerZ1_txrate_1
    Instance path:  PI.UA1.txrate
                              Cell usage:
                    cell       count     area    count*area
                    AND2        1      1.0       1.0
                    DFN1C0      1      1.0       1.0
                    DFN1P0      2      1.0       2.0
                    GND         1      0.0       0.0
                    NOR3A       1      1.0       1.0
                    VCC         1      0.0       0.0
                    XOR2        3      1.0       3.0

              TOTAL            10                8.0
------------------------------------------------------------
Report for cell clockdividerZ2.netlist
    Instance path:  PI.UA1.rxrate
                              Cell usage:
                    cell       count     area    count*area
                    AND2        5      1.0       5.0
                    AO1        14      1.0      14.0
                    DFN1C0      7      1.0       7.0
                    DFN1P0      5      1.0       5.0
                    GND         1      0.0       0.0
                    NOR2        3      1.0       3.0
                    NOR2A      10      1.0      10.0
                    NOR2B       2      1.0       2.0
                    NOR3        1      1.0       1.0
                    NOR3A       1      1.0       1.0
                    NOR3C       1      1.0       1.0
                    OR2         5      1.0       5.0
                    VCC         1      0.0       0.0
                    XNOR2      20      1.0      20.0
                    XOR2       10      1.0      10.0

              TOTAL            86               84.0
------------------------------------------------------------
Report for cell ProgramControl.netlist
    Instance path:  PA
                              Cell usage:
                    cell       count     area    count*area
                    AND3        1      1.0       1.0
                    AO1         1      1.0       1.0
                    AX1C        1      1.0       1.0
                    AX1D        3      1.0       3.0
                    DFN0P0      1      1.0       1.0
                    DFN1C0      1      1.0       1.0
                    DFN1E0C0    9      1.0       9.0
                    DFN1E0P0    5      1.0       5.0
                    GND         1      0.0       0.0
                    INV         1      1.0       1.0
                    MIN3        1      1.0       1.0
                    NOR2        1      1.0       1.0
                    NOR2B       6      1.0       6.0
                    NOR3C       1      1.0       1.0
                    OA1         1      1.0       1.0
```

OR2A	3	1.0	3.0
VCC	1	0.0	0.0
XNOR2	2	1.0	2.0
XOR2	3	1.0	3.0
XOR3	3	1.0	3.0
TOTAL	46		44.0

Report **for** cell RebootCounter.netlist
Instance path: RC

Cell usage:

cell	count	area	count*area
AND2	27	1.0	27.0
AND2A	2	1.0	2.0
AND3	48	1.0	48.0
AND3A	1	1.0	1.0
AO1	6	1.0	6.0
AO1A	2	1.0	2.0
AO1B	2	1.0	2.0
AO1C	11	1.0	11.0
AOI1A	3	1.0	3.0
AOI1B	3	1.0	3.0
AX1C	10	1.0	10.0
DFN0C1	21	1.0	21.0
DFN1C0	28	1.0	28.0
DFN1C1	19	1.0	19.0
DFN1E0C1	1	1.0	1.0
DFN1E0P1	1	1.0	1.0
DFN1E1C1	21	1.0	21.0
DFN1E1P1	1	1.0	1.0
DFN1P0	1	1.0	1.0
GND	1	0.0	0.0
INV	2	1.0	2.0
NOR2	10	1.0	10.0
NOR2A	15	1.0	15.0
NOR2B	37	1.0	37.0
NOR3	4	1.0	4.0
NOR3A	9	1.0	9.0
NOR3C	17	1.0	17.0
OA1	4	1.0	4.0
OA1A	6	1.0	6.0
OAI1	1	1.0	1.0
OR2	2	1.0	2.0
OR2A	16	1.0	16.0
OR2B	1	1.0	1.0
OR3C	3	1.0	3.0
VCC	1	0.0	0.0
XNOR2	22	1.0	22.0
XOR2	55	1.0	55.0
TOTAL	414		412.0

Report **for** cell IndexCounter.netlist
Instance path: IC

Cell usage:

cell	count	area	count*area
AND3	1	1.0	1.0
DFN0C1	5	1.0	5.0
DFN1C1	5	1.0	5.0
DFN1E0C1	4	1.0	4.0
GND	1	0.0	0.0
INV	1	1.0	1.0

```
                              NOR2A          1        1.0          1.0
                              NOR2B          3        1.0          3.0
                               OAI1          1        1.0          1.0
                                OR2          1        1.0          1.0
                               OR2B          1        1.0          1.0
                                OR3          1        1.0          1.0
                               OR3C          1        1.0          1.0
                                VCC          1        0.0          0.0
                               XA1A          1        1.0          1.0
                              XNOR2          6        1.0          6.0
                               XOR2          3        1.0          3.0
                               lsfr          1        8.0          8.0

                    TOTAL                   38                    43.0
------------------------------------------------------------------------
Report for cell lsfr.netlist
     Instance path:  IC.lsfr_checksum
                                                Cell usage:
                               cell        count       area     count*area
                             DFN0C0          4        1.0          4.0
                             DFN0P0          1        1.0          1.0
                                GND          1        0.0          0.0
                              NOR2B          1        1.0          1.0
                                VCC          1        0.0          0.0
                               XOR2          1        1.0          1.0
                               XOR3          1        1.0          1.0

                    TOTAL                   10                     8.0
------------------------------------------------------------------------
Report for cell ComparisonSignaturePlusGonIdenIndex.netlist
     Instance path:  CS_G
                                                Cell usage:
                               cell        count       area     count*area
                             DFN1C0          1        1.0          1.0
                             DFN1P0          3        1.0          3.0
                                GND          1        0.0          0.0
                                INV          1        1.0          1.0
                              NOR2A          1        1.0          1.0
                              NOR2B          1        1.0          1.0
                              NOR3C          5        1.0          5.0
                                OA1          1        1.0          1.0
                                OR2          1        1.0          1.0
                               OR2A          1        1.0          1.0
                                VCC          1        0.0          0.0
                               XA1A          5        1.0          5.0
                              XNOR2         10        1.0         10.0

                    TOTAL                   32                    30.0
------------------------------------------------------------------------
Report for cell WatchdogTimer.netlist
     Instance path:  WDT
                                                Cell usage:
                               cell        count       area     count*area
                               AO1B          3        1.0          3.0
                               AO1C          3        1.0          3.0
                               AOI1          4        1.0          4.0
                              AOI1B          1        1.0          1.0
                               AX1C          1        1.0          1.0
                               AX1E          5        1.0          5.0
                             DFN1C0         21        1.0         21.0
                           DFN1E1P0          1        1.0          1.0
                                GND          1        0.0          0.0
```

```
                                    MX2A          1        1.0         1.0
                                   NOR2A         10        1.0        10.0
                                   NOR2B          9        1.0         9.0
                                   NOR3A          3        1.0         3.0
                                   NOR3C         13        1.0        13.0
                                     OA1          2        1.0         2.0
                                    OAI1          1        1.0         1.0
                                     OR2          4        1.0         4.0
                                    OR2A          7        1.0         7.0
                                    OR2B          1        1.0         1.0
                                     OR3          2        1.0         2.0
                                    OR3A          3        1.0         3.0
                                    OR3B          5        1.0         5.0
                                    OR3C          7        1.0         7.0
                                     VCC          1        0.0         0.0
                                     XA1          1        1.0         1.0
                                   XNOR2          2        1.0         2.0

                              TOTAL             112                  110.0
-----------------------------------------------------------------------------
Report for cell serialinterface.netlist
     Instance path: SI
                                              Cell usage:
                                    cell      count       area    count*area
                                    AO1B          1        1.0         1.0
                                  DFN1C0          2        1.0         2.0
                                  DFN1C1          1        1.0         1.0
                                DFN1E0C0         20        1.0        20.0
                                DFN1E1C0         24        1.0        24.0
                                DFN1E1P0          1        1.0         1.0
                                     GND          1        0.0         0.0
                                     MX2          8        1.0         8.0
                                    MX2C         16        1.0        16.0
                                   NOR2A         19        1.0        19.0
                                   NOR2B          1        1.0         1.0
                                    NOR3          1        1.0         1.0
                                    OA1A          1        1.0         1.0
                                    OA1B          1        1.0         1.0
                                     OR2          2        1.0         2.0
                                    OR2A          4        1.0         4.0
                                    OR2B          4        1.0         4.0
                                    OR3C          1        1.0         1.0
                                     VCC          1        0.0         0.0
                                   XNOR2          1        1.0         1.0
                                   uartZ0         1      215.0       215.0

                              TOTAL             111                  323.0
-----------------------------------------------------------------------------
Report for cell uartZ0.netlist
     Instance path: SI.UA1
                                              Cell usage:
                                    cell      count       area    count*area
                                  DFN1C0          1        1.0         1.0
                                     GND          1        0.0         0.0
                                     VCC          1        0.0         0.0
                          clockdividerZ0          1       75.0        75.0
              clockdividerZ1_txrate_1             1        8.0         8.0
                      rxunit_Urxunit_1            1       63.0        63.0
                      txunit_Utxunit_1            1       68.0        68.0

                              TOTAL               7                  215.0
-----------------------------------------------------------------------------
```

```
Report for cell rxunit_Urxunit_1.netlist
  Original Cell name rxunit
     Instance path: SI.UA1.Urxunit
                                           Cell usage:
                              cell         count      area      count*area
                              AND2           1         1.0         1.0
                              AX1E           1         1.0         1.0
                            DFN1C0           4         1.0         4.0
                            DFN1C1           1         1.0         1.0
                          DFN1E0C0           1         1.0         1.0
                          DFN1E1C0          18         1.0        18.0
                               GND           1         0.0         0.0
                              MX2A           1         1.0         1.0
                              NOR2           1         1.0         1.0
                             NOR2A           3         1.0         3.0
                             NOR2B           9         1.0         9.0
                              NOR3           2         1.0         2.0
                             NOR3A           1         1.0         1.0
                             NOR3B           5         1.0         5.0
                             NOR3C           4         1.0         4.0
                              OAI1           1         1.0         1.0
                               OR2           2         1.0         2.0
                              OR2A           2         1.0         2.0
                              OR2B           1         1.0         1.0
                              OR3B           1         1.0         1.0
                               VCC           1         0.0         0.0
                              XOR2           4         1.0         4.0

                        TOTAL               65                    63.0
--------------------------------------------------------------------------
Report for cell txunit_Utxunit_1.netlist
  Original Cell name txunit
     Instance path: SI.UA1.Utxunit
                                           Cell usage:
                              cell         count      area      count*area
                               AO1           1         1.0         1.0
                              AO1B           1         1.0         1.0
                              AO1C           1         1.0         1.0
                              AO1D           1         1.0         1.0
                             AOI1B           1         1.0         1.0
                            DFN1C0           4         1.0         4.0
                          DFN1E0C0           8         1.0         8.0
                          DFN1E0P0           1         1.0         1.0
                          DFN1E1C0           9         1.0         9.0
                               GND           1         0.0         0.0
                               MX2           5         1.0         5.0
                              MX2B           1         1.0         1.0
                              MX2C           3         1.0         3.0
                              NOR2           1         1.0         1.0
                             NOR2A           1         1.0         1.0
                             NOR2B           3         1.0         3.0
                              NOR3           1         1.0         1.0
                             NOR3A           1         1.0         1.0
                             NOR3B           1         1.0         1.0
                               OA1           1         1.0         1.0
                              OA1B           2         1.0         2.0
                               OR2           1         1.0         1.0
                              OR2A           6         1.0         6.0
                              OR3A           1         1.0         1.0
                               VCC           1         0.0         0.0
                             XNOR2           1         1.0         1.0
                              XOR2           4         1.0         4.0
```

XOR3	2	1.0	2.0
synchroniser	1	6.0	6.0
TOTAL	65		68.0

Report **for** cell synchroniser.netlist
Instance path: SI.UA1.Utxunit.sync

Cell usage:

cell	count	area	count*area
DFN1C0	2	1.0	2.0
DFN1P1	1	1.0	1.0
GND	1	0.0	0.0
NOR2	1	1.0	1.0
OR2A	2	1.0	2.0
VCC	1	0.0	0.0
TOTAL	8		6.0

Report **for** cell clockdividerZ1_txrate_1.netlist
Original Cell name clockdividerZ1
Instance path: SI.UA1.txrate

Cell usage:

cell	count	area	count*area
AND2	1	1.0	1.0
DFN1C0	1	1.0	1.0
DFN1P0	2	1.0	2.0
GND	1	0.0	0.0
NOR3A	1	1.0	1.0
VCC	1	0.0	0.0
XOR2	3	1.0	3.0
TOTAL	10		8.0

Report **for** cell clockdividerZ0.netlist
Instance path: SI.UA1.rxrate

Cell usage:

cell	count	area	count*area
AND2	4	1.0	4.0
AO1	12	1.0	12.0
DFN1C0	7	1.0	7.0
DFN1P0	4	1.0	4.0
GND	1	0.0	0.0
NOR2	4	1.0	4.0
NOR2A	9	1.0	9.0
NOR2B	2	1.0	2.0
NOR3A	1	1.0	1.0
NOR3C	1	1.0	1.0
OR2	4	1.0	4.0
VCC	1	0.0	0.0
XNOR2	18	1.0	18.0
XOR2	9	1.0	9.0
TOTAL	77		75.0

Sorgente 27: Report dell'analisi di potenza

```

Power Report for design watchdog with the following settings:


 Vendor:     Actel Corporation
 Program:    Actel Designer Software, Release v8.6 SPB (Version 8.6.1.4)
             Copyright (C) 1989-2009
 Date:       Sat Oct 31 03:34:18 2009
 Version:    3.0



 Design:                    watchdog
 Family:                    ProASIC3L
 Die:                       M1A3P1000L
 Package:                   484 FBGA
 Temperature Range:         COM
 Voltage Range:             COM
 Operating Conditions:      Typical
 Operating Mode:            Active
 Data Source:               Advanced


Power Summary
+---------------+-------------+------------+
|               | Power (mW)  | Percentage |
+---------------+-------------+------------+
| Total Power   |     18.309  |     100.0% |
| Static Power  |      4.328  |      23.6% |
| Dynamic Power |     13.981  |      76.4% |
+---------------+-------------+------------+


Thermal Summary
+-------------------------+------------------------------------+
| Junction Temperature is | defined by operating conditions    |
+-------------------------+------------------------------------+
| Operating Conditions    | Typical                            |
| Temperature Range       | COM                                |
|                         |                                    |
| Junction Temperature    | 25.0 C                             |
| Junction Temperature is | Within range limits                |
+-------------------------+------------------------------------+


Clock Domains Summary: Frequency
+------------------------------------------------+--------+--------------+--------------+------
| Name                                           | Clocks | Register     | Set/
    Reset      | Primary        | Combinational  |
|                                                | (MHz)  | outputs      | nets
               | inputs         | outputs        |
|                                                |        | (MHz)        | (MHz
    )          | (MHz)          | (MHz)          |
+------------------------------------------------+--------+--------------+--------------+------
| SI/UA1/Urxunit/receiveLoad:Q                   | 0.000  | 0.000 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| SI/loadTxDataS:Q                               | 0.000  | 0.015 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
```

```
| SI/UA1/Urxunit/dataAvailable:Q                       | 0.000   | 0.015 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| PI/UA1/Urxunit/receiveLoad:Q                         | 0.000   | 0.000 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| PI/loadTxDataS:Q                                     | 0.000   | 0.015 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| PI/UA1/Urxunit/dataAvailable:Q                       | 0.000   | 0.014 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| uartClk                                              | 30.000  | 0.747 (4.98 %) |
    0.000 (0.00 %) | 1.500 (10.00 %) | 0.924 (6.16 %) |
| clk                                                  | 30.000  | 0.144 (0.96 %) |
    0.000 (0.00 %) | 1.500 (10.00 %) | 0.126 (0.84 %) |
| IncrementRBC_un6_interrupts_inferred_clock_RNO:Y     | 0.000   | 1.426 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.664 (0.00 %) |
| RC/reloadFlash/U1:Q                                  | 0.000   | 0.015 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 0.000 (0.00 %) |
| CS_G/readGoldenIndex:Q                               | 0.000   | 6.759 (0.00 %) |
    0.000 (0.00 %) | 0.000 (0.00 %)  | 4.369 (0.00 %) |
+------------------------------------------------------+---------+----------------+----------------+------

Clock Domains Summary: Probability
+------------------------------------------------------+---------+----------+----------+---------+------
| Name                                                 | Clocks | Register | Set/Reset
    | Primary | Combinational |
|                                                      | (%)    | outputs  | nets
    | inputs  | outputs       |
|                                                      |        | (%)      | (%)
    | (%)     | (%)           |
+------------------------------------------------------+---------+----------+----------+---------+------
| SI/UA1/Urxunit/receiveLoad:Q                         | 50.000 | 50.000   | 50.000
    | 50.000  | 50.000        |
| SI/loadTxDataS:Q                                     | 50.000 | 0.098    | 50.000
    | 50.000  | 50.000        |
| SI/UA1/Urxunit/dataAvailable:Q                       | 50.000 | 99.899   | 50.000
    | 50.000  | 50.000        |
| PI/UA1/Urxunit/receiveLoad:Q                         | 50.000 | 50.000   | 50.000
    | 50.000  | 50.000        |
| PI/loadTxDataS:Q                                     | 50.000 | 0.097    | 50.000
    | 50.000  | 50.000        |
| PI/UA1/Urxunit/dataAvailable:Q                       | 50.000 | 99.905   | 50.000
    | 50.000  | 50.000        |
| uartClk                                              | 50.002 | 48.491   | 50.000
    | 50.000  | 43.873        |
| clk                                                  | 50.002 | 49.535   | 50.000
    | 50.000  | 61.512        |
| IncrementRBC_un6_interrupts_inferred_clock_RNO:Y     | 50.000 | 50.074   | 50.000
    | 50.000  | 35.583        |
| RC/reloadFlash/U1:Q                                  | 50.000 | 0.098    | 50.000
    | 50.000  | 50.000        |
| CS_G/readGoldenIndex:Q                               | 50.000 | 49.691   | 50.000
    | 50.000  | 40.171        |
+------------------------------------------------------+---------+----------+----------+---------+------

Set Of Pins Summary: Frequency
+------+-------+
| Name | Data  |
|      | (MHz) |
+------+-------+
```

```
+------+------+
Set Of Pins Summary: Probability
+------+------+
| Name | Data |
|      | (%)  |
+------+------+
+------+------+


Activity Summary
+-----+-----+--------+-----------------+--------+
| Pin | Net | Domain | Frequency (MHz) | Source |
+-----+-----+--------+-----------------+--------+
+-----+-----+--------+-----------------+--------+


Activity Annotation Statistics Summary
+-----------------------+------------+
| Source                | Precentage |
+-----------------------+------------+
| VCD Import            | 0.00       |
| Manual Annotation     | 0.00       |
| Fixed Values          | 0.41       |
| Vectorless Estimation | 99.59      |
+-----------------------+------------+
```

Sorgente 28: Report dell'analisi dei tempi

```
SmartTime Version 3.0
Actel Corporation - Actel Designer Software Release v8.6 SPB (Version 8.6.1.4)

 Design                          watchdog
 Family                          ProASIC3L
 Die                             M1A3P1000L
 Package                         484 FBGA
 Temperature                     COM
 Voltage                         COM
 Speed Grade                     STD
 Design State                    Post-Layout
 Data source                     Silicon verified
 Analysis Min Case               BEST
 Analysis Max Case               WORST
 Scenario for Timing Analysis    Primary

 Using Enhanced Min Delay Analysis
Pin Description
+----------------------+----------+--------+-----------------+
| Name                 | Location | Type   | I/O Technology  |
+----------------------+----------+--------+-----------------+
| receiveChecksum      | V11      | Input  | LVCMOS33 (1)    |
| rxProgramInterface   | AA10     | Input  | LVCMOS33 (1)    |
| clk                  | M19      | Clock  | LVCMOS33 (1)    |
| uartClk              | L5       | Clock  | LVCMOS33 (1)    |
| Grst                 | M15      | Input  | LVCMOS33 (1)    |
| RstCpu               | AB8      | Output | LVCMOS33 (2)    |
| TckCpu               | W9       | Output | LVCMOS33 (2)    |
| txOut                | Y12      | Output | LVCMOS33 (2)    |
| TxProgramInterface   | W7       | Output | LVCMOS33 (2)    |
+----------------------+----------+--------+-----------------+


DC Electrical Characteristics
+---------------+------+----------+------------+--------+------+------+--------+
| Name          | Vcci | Resistor | Hot        | Output | Slew | Skew | Output |
|               | (V)  | Pull     | Swappable  | Drive  |      |      | Load   |
|               |      |          |            | (mA)   |      |      | (pF)   |
+---------------+------+----------+------------+--------+------+------+--------+
| LVCMOS33 (1)  | 3.3  | None     | no         |        |      |      |        |
| LVCMOS33 (2)  | 3.3  | None     | no         | 12     | High | no   | 5      |
+---------------+------+----------+------------+--------+------+------+--------+


AC Electrical Characteristics
+------------------+-------------------+--------+------------------+--------+--------+------+
| Description      |                   |        |                  | Min    |
   Max     | Unit |
+------------------+-------------------+--------+------------------+--------+--------+------+
| Clock frequency  | clk               |        |                  |        |
   60.150  | MHz  |
| Clock period     | clk               |        |                  | 16.625 |
           | ns   |
| Clock frequency  | uartClk           |        |                  |        |
   68.283  | MHz  |
| Clock period     | uartClk           |        |                  | 14.645 |
           | ns   |
```

```
53 | Setup time          | receiveChecksum     | before | uartClk (rise)      | 3.342   |
           | ns    |
54 | Setup time          | rxProgramInterface  | before | uartClk (rise)      | 5.184   |
           | ns    |
55 | Hold time           | receiveChecksum     | after  | uartClk (rise)      | 0.325   |
           | ns    |
56 | Hold time           | rxProgramInterface  | after  | uartClk (rise)      | 0.016   |
           | ns    |
57 | Recovery time       | Grst                | before | clk (rise)          | 3.765   |
           | ns    |
58 | Recovery time       | Grst                | before | uartClk (rise)      | 0.294   |
           | ns    |
59 | Removal time        | Grst                | after  | clk (rise)          | 0.223   |
           | ns    |
60 | Removal time        | Grst                | after  | uartClk (rise)      | 0.209   |
           | ns    |
61 | Propagation delay | clk (rise)          | to     | RstCpu              | 2.049   |
     7.214   | ns    |
62 | Propagation delay | clk (rise)          | to     | TckCpu              | 2.175   |
     7.519   | ns    |
63 | Propagation delay | uartClk (rise)      | to     | TxProgramInterface  | 2.176   |
     7.517   | ns    |
64 | Propagation delay | uartClk (rise)      | to     | txOut               | 2.369   |
     8.002   | ns    |
65 +---------------------+---------------------+--------+---------------------+---------+-------+
```

Bibliografia

[1] D. Roascio, *Sottosistema di comunicazione tollerante ai guasti per satellite modulare AraMiS*. POLITECNICO DI TORINO, Novembre 2008.

[2] A. Gasperin, *Advanced Non-Volatile Memories: Reliability and Ionizing Radiation Effects*. PhD thesis, Università degli Studi di Padova, December 2008.

[3] D. Binder, E. Smith, and A. Holman, *Satellite anomalies from galatic cosmic rays*, vol. NS-22, pp. 2675–2680. IEEE, 1975.

[4] *The Radiation Design Handbook*. ESA PSS-01-609 Issue 1, ESA, Maggio 1993.

[5] W. Kolasinski, J. Blake, J. Antony, W. Price, and E. Smith, *Simulation of cosmic-ray induced soft errors and latchup in integrated-circuit computer memories*, vol. NS-26, pp. 5087–5091. IEEE, 1979.

[6] T. May and M. Woods, *Alpha-particle-induce soft errors in dynamic memories*, vol. ED-26, pp. 2–9. IEEE, 1979.

[7] *Military Handbook Guidelines for developing Radiation Hardness Assurance Device Specification*. MIL-HDBK-816, Department of Defense, United State of America, December 1994.

[8] M. Borri and L. Reyneri, *Mechanical Subsystem of the AraMiS Architecture – Overview and application concepts, Version 1.0*, 2008.

[9] S. L. Vandelli, *Ricerca e sperimentazione virtuale di sistemi automatici per la schermatura e la protezione dalla radiazione solare per la strumentazione ottica di Simbio-Sys nella missione spaziale "BepiColombo"*. PhD thesis, UNIVERSITA' DEGLI STUDI DI PADOVA, Gennaio 2008.

[10] J. Wallmark and S. Marcus, *Minimum size and maximum packaging density of non- redundant semiconductor devices*, vol. 50, pp. 286–298. Proc.IRE, 1962.

[11] A. Waskiewicz, J. Groninger, V. Strahan, and D. Long, *Burnout of power MOS transistor with heavy ions of Californium-252*, vol. NS-33, pp. 1710–1713. IEEE, 1986.

[12] J. E. Mazur, "An overview of the space radiation environment." <http://www.aero.org/publications/crosslink/summer2003/02.html

[13] R. Munakata, *CubeSat Design Specification.* California Polytechnic State University, 12 ed., Agust 2009.

[14] "System-critical fpgas (product cattalog)." Actel, July 2009.

[15] Texsas Instrument, *MSP430FG43x MIXED SIGNAL MICROCONTROLLER*, slas380b ed., June 2007.

[16] Texas Instrument, *MSP430x4xx Family User's Guide*, slau056g ed., 2007.

[17] *MSP430 Memory Programming User's Guide*, February 2009.

[18] *Secuencias pseudoaleatorias para telecomunicaciones*, ch. 3, pp. 61–86. Edicions UPC, 1998.

[19] D. A. Kamp, A. D. DeVilbiss, S. C. Philpy, and G. F. Derbenwick, "Adaptable ferroelectric memories for space applications," *Wafer fabrication of the prototype radiation-hardened FeRAM was supported by the Air Force Research*, July 2004.