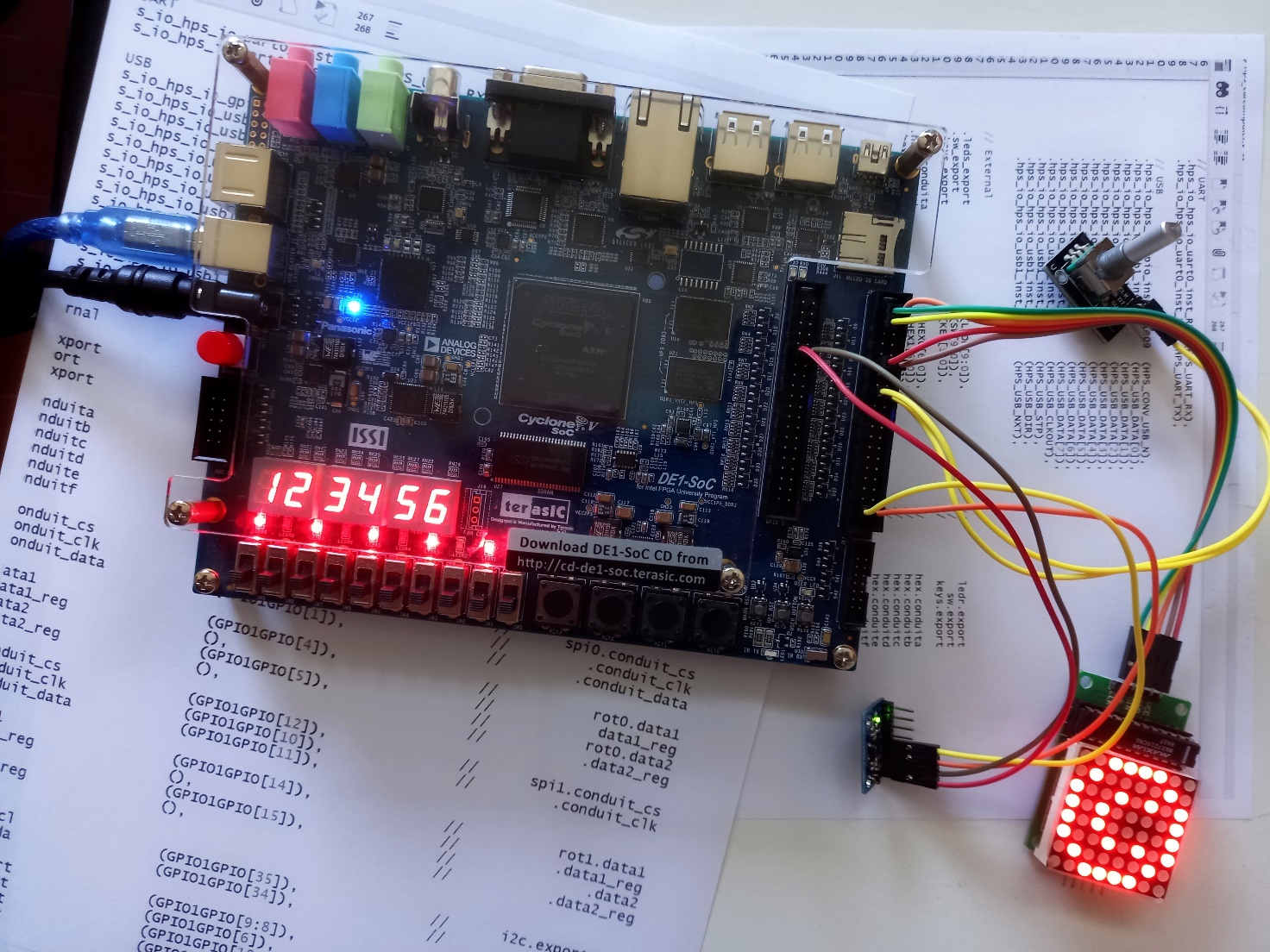
Progettazione di Sistemi Elettronici

# Prof. Marsi Stefano - Università di Trieste Anno Accademico 2023/24

Tutorial 6



**Sviluppo di un sistema di calcolo completo di interfacce basato su ARM.**  
**Hardware** impiegato: Terasic DE1-SoC Board  
**Software** Impiegato: Quartus 22.1

Tutorial 6

Realizzazione di un sistema di calcolo dotato di periferiche sia standard che custom, basato su processore ARM

Descrizione: In questo tutorial si andrà a realizzare un sistema composto da un processore ARM e di diverse periferiche sia custom che realizzate da terze parti.

Scopo: Lo scopo è quello di vedere come si possa realizzare un intero sistema di calcolo all’interno di una singola FPGA e come questo sistema possa essere impiegato per interagire col mondo esterno attraverso periferiche già sviluppate da terze parti oppure anche sviluppando in proprio la periferica più idonea per realizzare una certa interfaccia verso l’esterno.

Apprendimento previsto:

* Approfondimento del tool di Quartus “Platform Designer” per realizzare un processore insieme al suo sistema di interfaccie
* Configurazione del processore e delle periferiche sviluppate da terze parti
* Realizzazione di periferiche “ad hoc”
* Programmazione del processore ed esecuzione del codice in un processo “Hardware-Software” Codesign

# Premessa

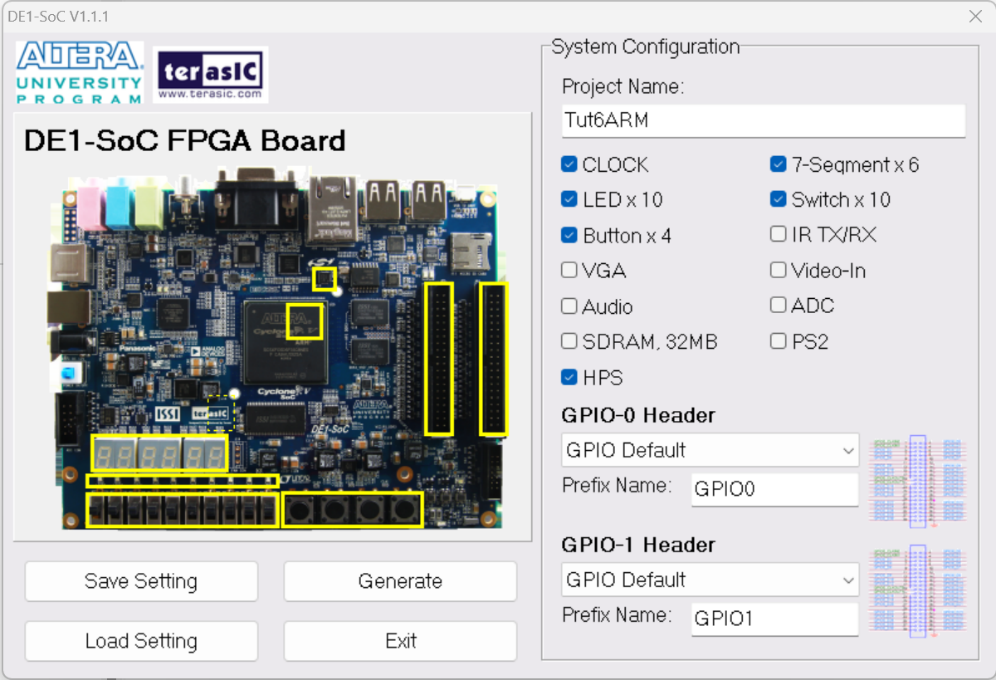
Una moderna FPGA contiene al suo interno sia risorse per realizzare Hardware dedicato, ma anche risorse specifiche che realizzano sistemi di calcolo, controllo ed interfaccia verso il mondo esterno secondo i più comuni protocolli in modo da realizzare quello che comunemente viene definito System On Chip (SoC). In particolare l’ FPGA montata sulla DE1-SoC ha già integrata in forma Hardware un Processore ARM dual Core oltre a EMACs, USB Controllers, I2C Controllers, UARTs, CAN Controllers,SPI Master Controllers,SPI Slave Controllers, GPIO Interfaces ecc. ecc.

In questo tutorial si vedrà come interagire con questi elementi per realizzare un sistema di calcolo e di interfaccia ad Hoc, scegliendo in particolare quali elaborazioni sia più opportuno vengano svolte in Hardware e quali in software.

# Realizzazione dell’ architettura

Si inizi realizzando lo “scheletro” di un sistema utilizzando il tool “System Builder” che preveda l’impiego di

* HPS (Hardware Processor System)
* LEDs
* Display a 7 segmenti
* Swithes
* Tasti
* Clock
* SystemID
* GPIO0 e GPIO1 (in modalità default)



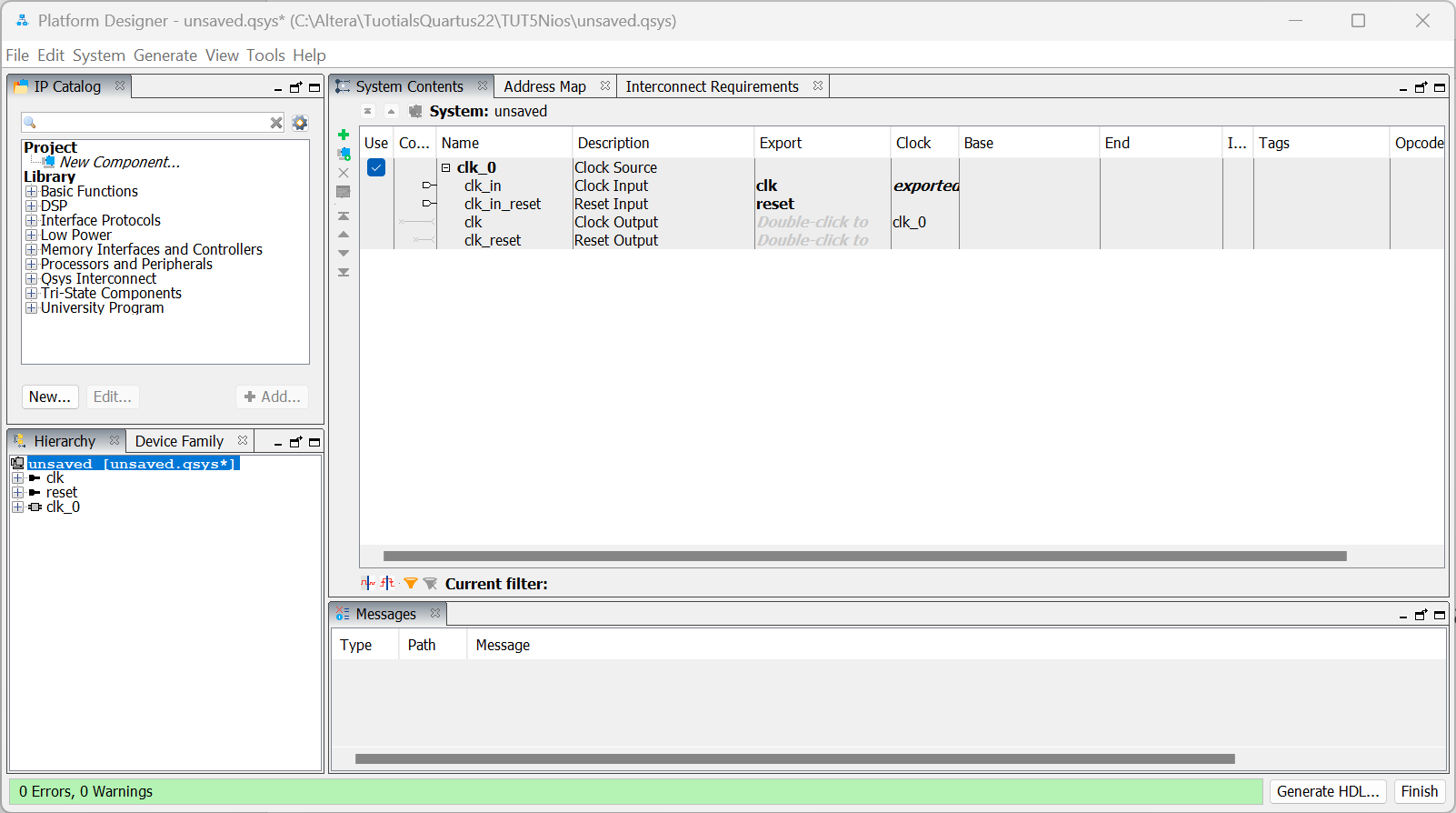
Non tutte queste periferiche verranno impiegate all’inizio, ma la loro presenza sarà utile più tardi senza dover riconsiderare il progetto dall’inizio.

Si Apra il progetto così generato all’interno di Quartus.

All’interno di Quartus:

Tools > Platform Design

Si apre un sistema che consente di configurare l’architettura in base a blocchi già sviluppati da terze parti.

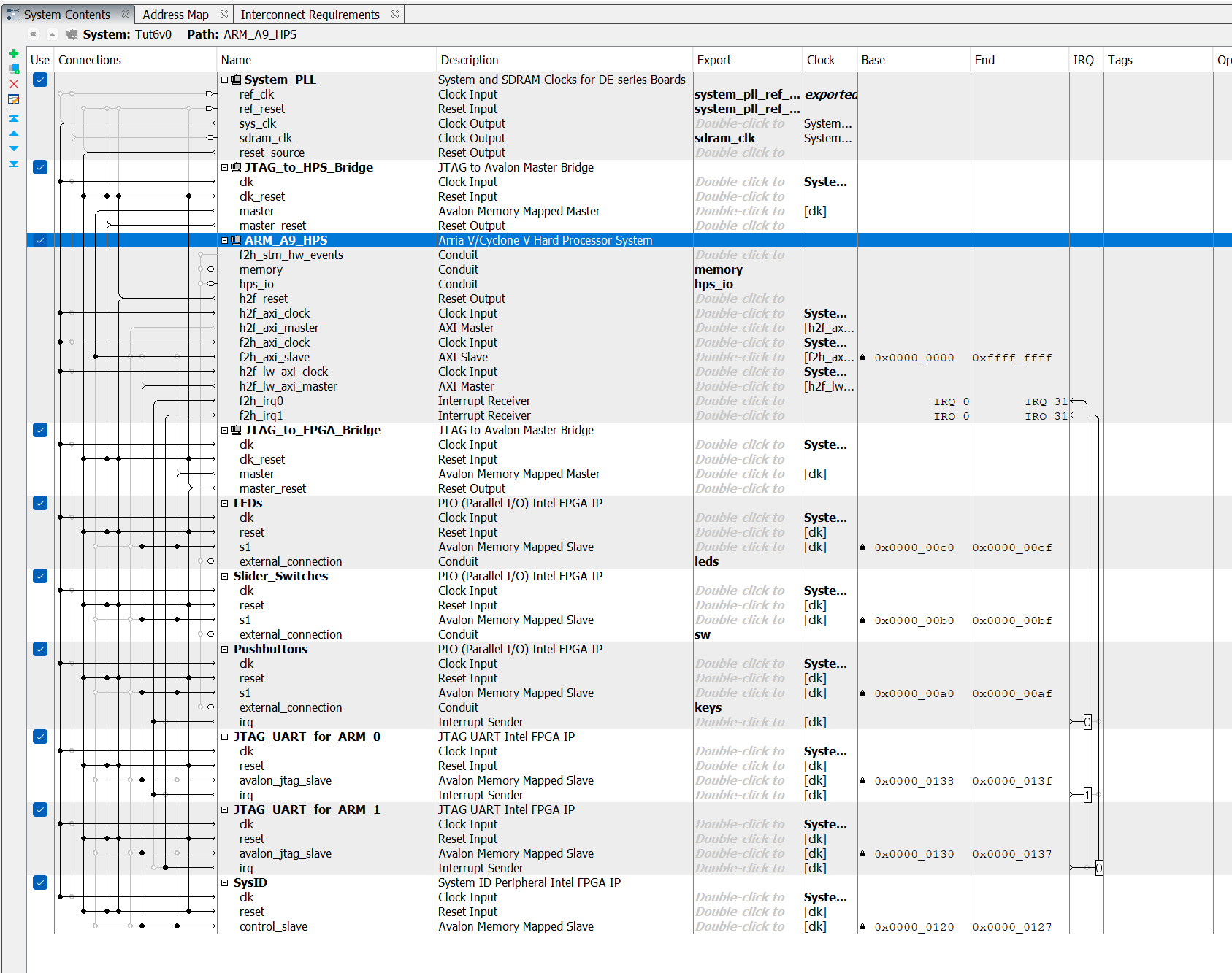


Un primo blocco è già presente ed è un blocco utile alla gestione del clock e del reset, ma questo verrà sostituito con un altro che potrà essere utilizzato anche per progetti che dovessero ad esempio utilizzare la memoria esterna (SDRAM) presente sulla DE1-SoC

Si Includano quindi i seguenti blocchi

* **System and SDRAM clock For DE1-series Boards** (il soft processor)
* **JTAG\_to\_HPS\_Bridge** (l’interfaccia che consente al JTAG di accedere alle risorse di HPS)
* **ARM\_A9\_HPS** (il processore ARM)
* **JTAG\_to\_FPGA\_Bridge** (l’interfaccia che consente al JTAG di accedere alle risorse dell’ FPGA)
* Due **JTAG UART Intel FPGA IP** (L’interfaccia seriale STDIO per dialogare col processore) per interfacciarsi con i due processori ARM
* Tre interfacce di tipo **PIO (Parallel I/O) Intel FPGA IP** opportunamente dimensionate (per interfacciarsi rispettivamente con Switches,LEDs e KEYs)
* **System ID Peripheral Intel FPGA IP** (identificatore del sistema – SysID)

La finestra con l’architettura si dovrebbe presentare più o meno così:



Si rinomino opportunamente i vari blocchi e si configurino le connessioni come segue:

* Il segnale di clock in uscita dal “System PLL” sia collegato a tutti i blocchi ed in particolare alle 3 interfaccie dell’ARM
* Il segnale di Reset in uscita dal clock generator sia collegato a tutti i blocchi

System > Create Global Reset Network

* La linea **h2f\_lw\_axi\_master** del ARM raggiunga tutti i blocchi impiegati come periferica   
  (ovvero l’ARM legge/scrive dati da tutti i blocchi attraverso la sua interfaccia lightweight a 32 bit)
* la Linea **h2f\_axi\_master** rimanga scollegata
* la Linea **f2h\_axi\_slave** sia collegata con la porta master di **JTAG\_to\_HPS\_bridge** (questo consentirà al tool di debugging di poter interagire col processore)
* I segnali **memory** e **hps\_io** vengano esportati verso l’FPGA
* Il segnale **f2h\_stm\_hw\_event** (se presente) può rimanere scollegato
* I segnali **f2h\_irq0** e **f2h\_irq1** siano collegati rispettivamente alle due interfaccie **JTAG\_UART\_For\_ARM**
* Verificare che tutte le interfacce (Led,Switches,Button,SysID, Bridge-x2, UART-x2) ricevano
  + Il clock da System PLL
  + Il Reset da:
    - SystemPLL
    - JTAG2HPS Bridge
    - JTAG2FPGA Bridge
    - ARM (dalla porta h2f\_reset)
* Verificare che tutte le interfacce verso l’esterno (Led,Switches,Button,SysID) ricevano
  + sulla porta slave (**s1**) i segnali provenienti
    - JTAG2FPGA Bridge
    - ARM (dalla porta h2f\_lw\_axi\_master)
  + Si esportino per esse verso l’esterno tutti i segnali di interfaccia (conduit) assegnado eventualmente un nome

Si vadano ora a definire (manualmente o automaticamente) gli opportuni indirizzi di memoria per i vari blocchi. Utile in tal senso può essere la procedura automatica

System > Assign Base Address

E si definiscano (manualmente o automaticamente) gli “interrupt numbers”

System > Assign Interrupt Numbers

L’architettura fin qui definita dovrebbe assomigliare a quella riportata in figura.

Si devono ora configurare attentamente tutti i vari blocchi ed in particolare il Processore ARM che rispecchia nella sua struttura l’impiego delle interfaccie già previste sulla scheda De1\_Soc.

In Particolare le schede

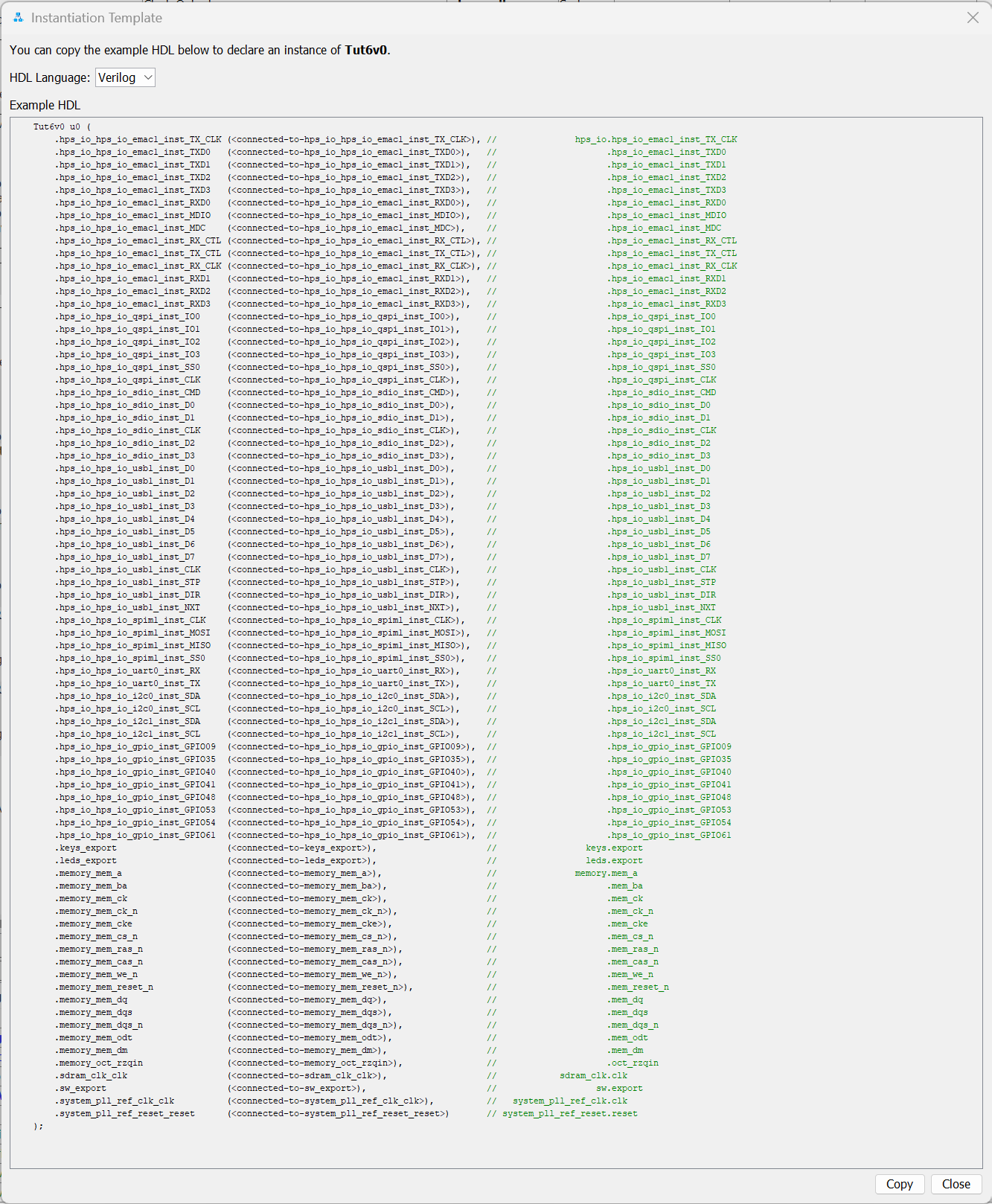
* **Fpga Interface** serve a configurare tutte le varie interfacce e gli interrupts derivanti dalle varie periferiche che costituiscono l’HPS
* **Peripherals Pins** vengono a definire quali periferiche di HPS vengono attivate e a quale settaggio predefinito di piedinatura afferire. Ad esempio: poiché nella scheda De1-SoC in corrispondenza ai piedini HPS55 e HPS56 è già installato un accelerometro la cui porta I2C afferisce proprio a quei piedini, ovviamente se lo si vuole utilizzare bisognerà garantire che l’I2C Controller I2C0 venga assegnato al set di I/O HPS I/O set0. Altrettanto dicasi ad esempio dei controller per accedere alla memoria SD o all’interfaccia USB o ancora alla porta Ethernet, ecc. ecc. Un altro esempio sono i piedini HPS53 e HPS54 che sono stati collegati rispettivamente ad un LED ed un pulsante accessibili solo attraverso HPS e che pertanto se si desidera impiegare devono essere esportati attraverso l’opportuna tabella delle connessioni.
* **HPS Clocks (suddiviso a sua volta in Input clocks e Output clocks)** definisce le frequenze dei clocks afferenti alle varie periferiche ed eventualmente li esporta
* **SDRAM (suddiviso a sua volta in sottocartelle)** definisce tutti i parametri per una idonea comunicazione verso la SDRAM.

Poiché il settaggio completo di tutti questi parametri, che peraltro deve rispecchiare le caratteristiche fisiche del cuircuito sul quale la FPGA è montata, può essere piuttosto oneroso si consiglia di trarre spunto dal file disponibile nella pagina Moodle del corso, oppure dal materiale software fornito iniseme alla scheda De1-SoC.

Una volta completata l’architettura del processore e delle periferiche si prenda nota degli indirizzi ai quali sono mappate le varie periferiche.

Potrebbe essere inoltre utile utile vedere un esempio di come instanziare il componente:

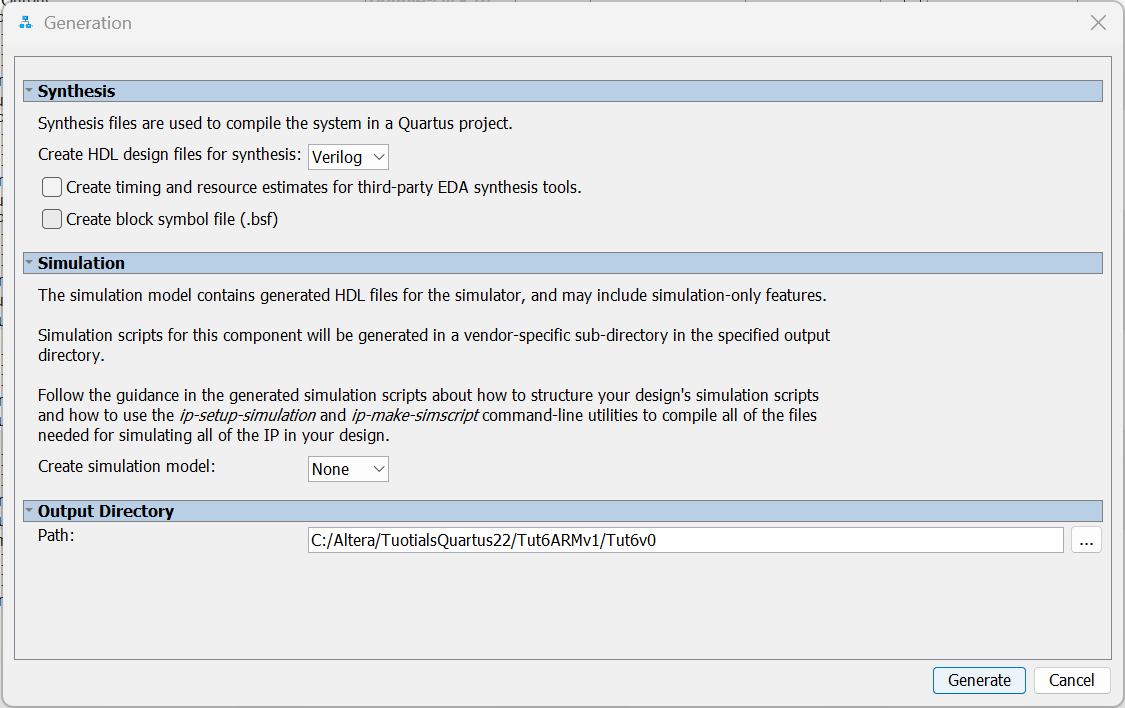
Generate > Show Instanstation Template



Che può essere preso come spunto per istanziare successivamente il componente all’interno della top level entity .

A questo punto si può generare l’architettura completa

< Generate HDL>



Scegliere eventualmente se si vuole creare anche il simbolo (utile all’interno di un eventuale schematico) oppure i files verilog per effettuarne una opportuna simulazione in Verilog – Ma entrambe le opzioni non verranno impiegate in questo tutorial

< Generate >

E successivamente

< Close >

< Finish >

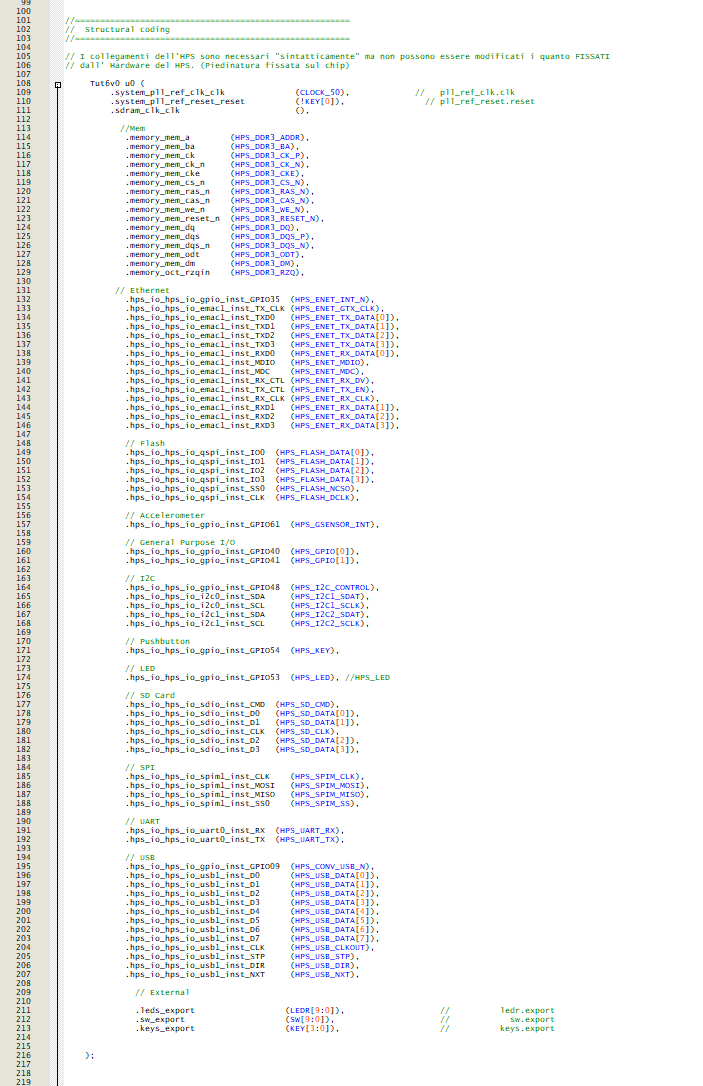
Probabilmente alla fine del processo comparirà una finestra che ricorda dove sono stati archiviati i files da includere manualmente al progetto.

# Istanziazione dell’ architettura all’interno del progetto

L’architettura generata al passo precedente deve essere adesso importata a livello di sistema globale ed interfacciata con eventuali altri blocchi e segnali.

Ad esempio bisogna far si che il clock ed un opportuno segnale di reset arrivi all’architettura ed i suoi segnali di I/O relativi a leds, keys e switches si colleghino ai corrispettivi piedini all’interno della scheda DE0-SoC. Inoltre bisogna garantire pure che i vari piedini di HPS dedicati al collegamento verso periferiche esterne (come ad esempio DDR3, USB, ENET , FLASH, I2C , GPIO , … ) che vengono condivisi verso l’esterno nell’istanziazione del componente precedentemente realizzato, trovino un opportuno collegamento fisico verso i corrispondenti componenti montati sulla scheda DE1-SoC.

Quindi a livello della “Top Level Entity” generata col tool “system builder” bisognerà creare tutti i collegamenti opportuni:



Un file di esempio con l’istanziazione appena riportata è disponibile nella pagina moodle relativa al corso, ma si deve fare particolare attenzione al fatto che i nomi dei segnali riportati sono coerenti con quelli assegnati ai segnali “esportati” all’interno del tool “Platform Designer” e laddove si fosse optato per nomi diversi, sarà opportuno modificare coerentemente anche il file di istanziazione. Si noti inoltre che il segnale impiegato per il reset in questo caso è il segnale relativo a KEY[0] **invertito** !

Da ultimo a livello di sistema bisogna ancora includere il file “.qip” generato nel progetto.

Project > Add/Remove files in project

Cliccare sui tre puntini

Andare a trovare il file laddove ci era stato indicato precedentemente dalla finestra di dialogo ed importarlo.

<OK>

A Questo punto l’intero sistema può essere compilato.

Si esegua **SOLAMENTE** il processo di “Analysys e Syntesis”

Successivamente bisogna eseguire due scripts per effettuare un’opportuna assegnazione dei parametri e dell’assegnazione dei piedini relativi a HPS che vanno ad imporre opportuni vincoli per quanto riguarda le fasi di *“Placement”, “Mapping” and “Routing”*

Tools > TcL scripts

E effettuare un doppio click su

hps\_sdram\_p0\_parameters.tcl

hps\_sdram\_p0\_pin\_assignments.tcl

tale procedura va eseguita solo una volta per progetto e non sarà necessario eseguirla nuovamente qualora si eseguissero modifiche all’architettura.

A questo punto si può completare la compilazione del progetto ed eseguire il download su scheda.

La Parte Hardware inerente alla realizzazione del sistema è ultimata e si può chiudere Quartus.

# Intel FPGA Monitor Program

Lanciare “Intel FPGA Monitor Program”

File > New Project

Definire una directory opportuna (si suggerisce all’uopo di creare, all’interno del progetto realizzato con Quartus un opportuno direttorio – denominato ad esempio “software”, nel quale far risiedere il progetto che per l’appunto prosegue ora nella realizzazione di un software opportuno di controllo del processore appena realizzato).  
Si assegni un nome al progetto e come architettura si scelga **ARM Cortex A9**

> Next

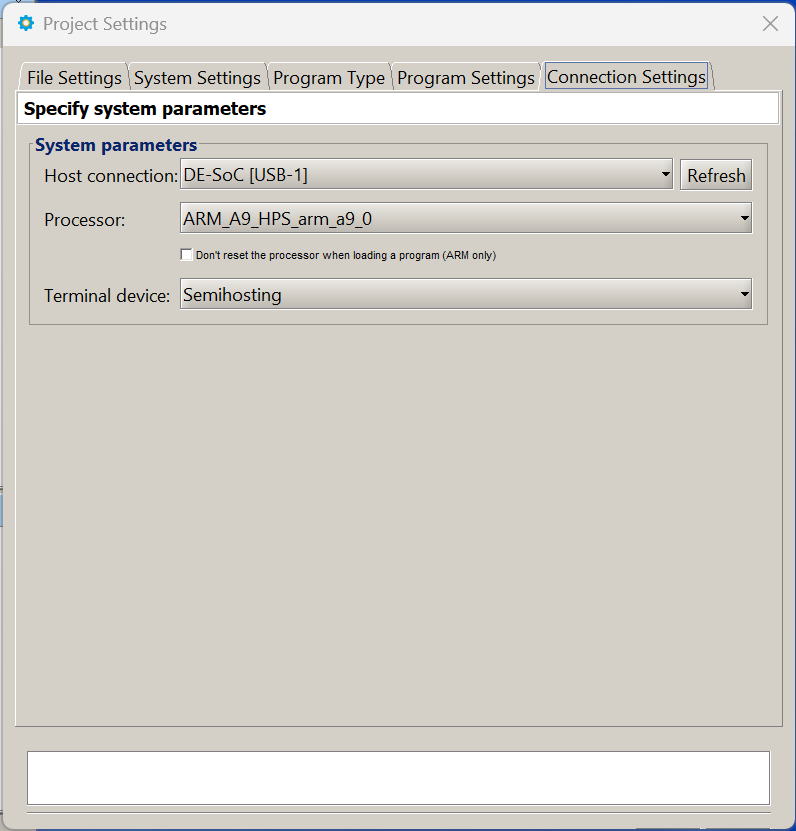
Come tipologia di architettura scegliere <Custom>, indicare poi i file che caratterizzano il sistema hardware .sopcinfo e .sof ubicati nel direttorio dove risiede l’hardware del progetto come pre-loader quello relativo alla scheda DE1-Soc e come preloader scegliere quello relativo alla scheda che si sta impiegando.

> Next

Come tipologia di programmazione al momento scegliere <No Program>

> Next

Definire quindi in “Connection Settings” processore ed interfaccia di comunicazione



> Save

Il sistema chiederà se si vuole caricare il file di configurazione sulla scheda, ma presentando un piccolo baco del sistema è meglio soprassedere su questo punto ed impiegare Quartus per programmare la scheda (come gia fatto).

Action > Connect to System

Accedere alla scheda Memory

Goto Address andare agli indirizzi in cui sono stati mappati LEDs, Swithes e SysID (ad es. 0xff200000), mettere la spunta su “Query Device”

> GO

A questo punto si possono vedere le varie periferiche attraverso i loro indirizzi di memoria:

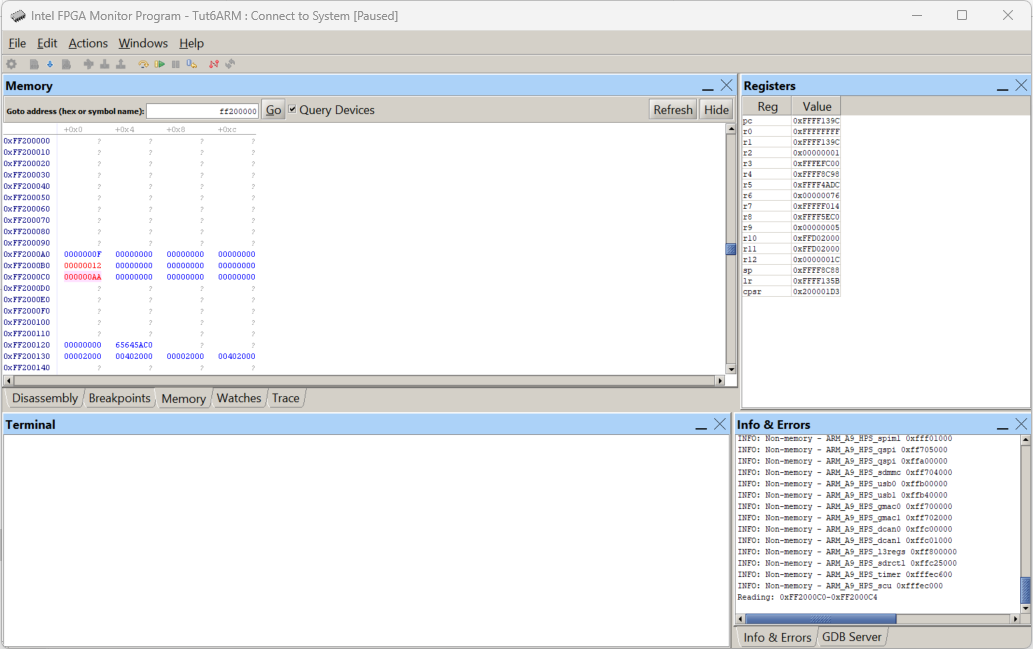
- scrivendo un qualche dato (es 0x2AA alla locazione relativa dei LED) si vedranno i led accendersi alternati)

- attivando una configurazione sugli switches e cliccando <Refresh> si vedrà tale configurazione mappata in memoria

- alla locazione del SysID si vedrà mappato il valore fissato via Hardware in fase di definizione dell’architettura

- Pigiando il tasto preposto al Reset (Key[0]) i led si spegneranno

In rosso vengono riportati i valori modificati dalla precedente lettura.



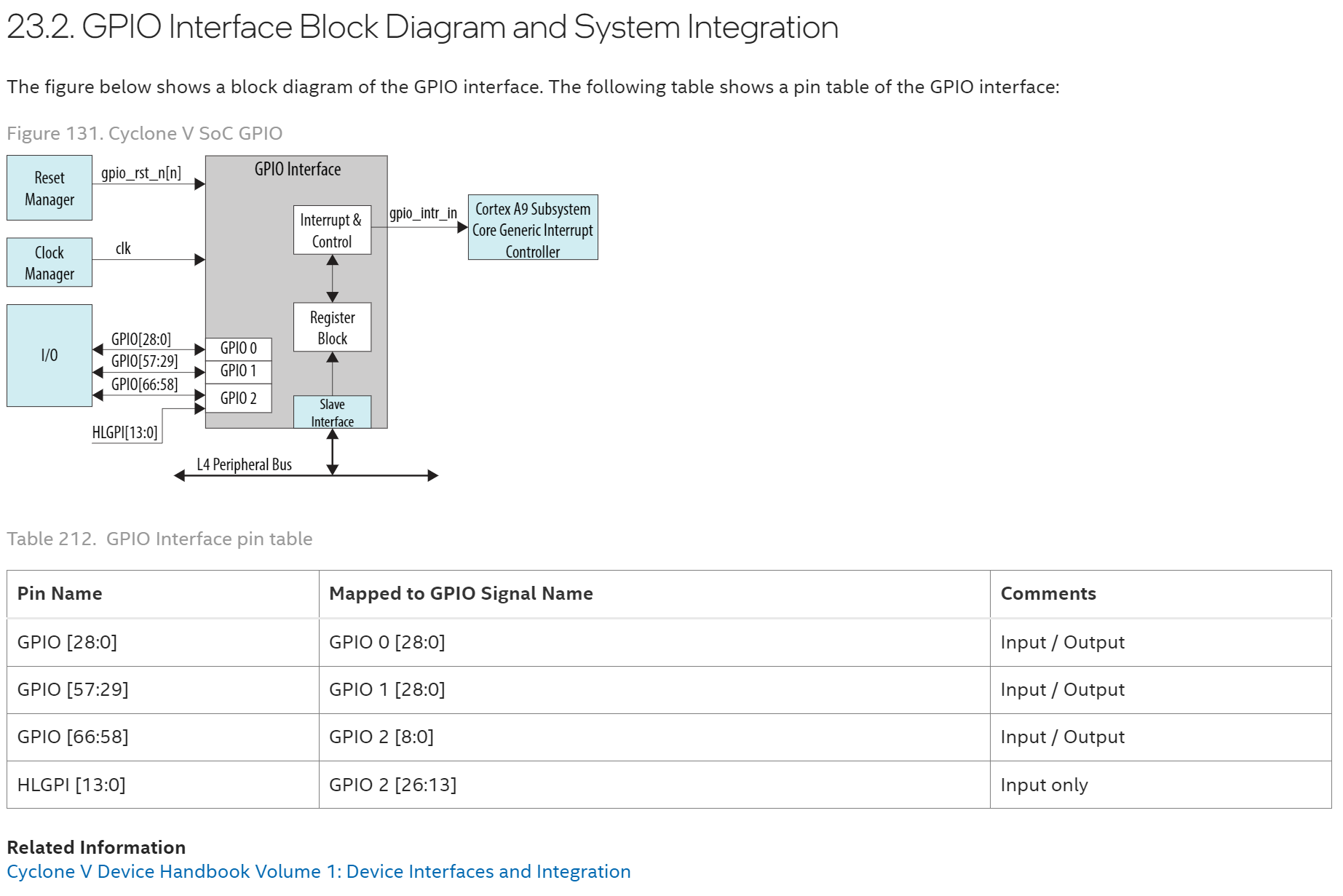
### Accesso ad altre periferiche embedded

Oltre alle periferiche realizzate su FPGA ed espressamente integrate utilizzando “Platform Design”, l’HPS integrato all’interno della CycloneV presenta, realizzate in hardware, molteplici altre periferiche (ENET, USB, UART, I2C, SPI, ….) di cui alcune sono state connesse ad opportuni dispositivi compatibili sulla scheda DE1-SoC, altre sono rimaste inutilizzate. In particolare vediamo che due piedini GPIO controllati ESCLUSIVAMENTE dall’ HPS (i piedini *hps\_io\_hps\_io\_gpio\_inst\_GPIO53* e *hps\_io\_hps\_io\_gpio\_inst\_GPIO54)* sono stati connessi rispettivamente ad un Led e ad un pulsante dedicati disposti sulla scheda DE1-SoC.

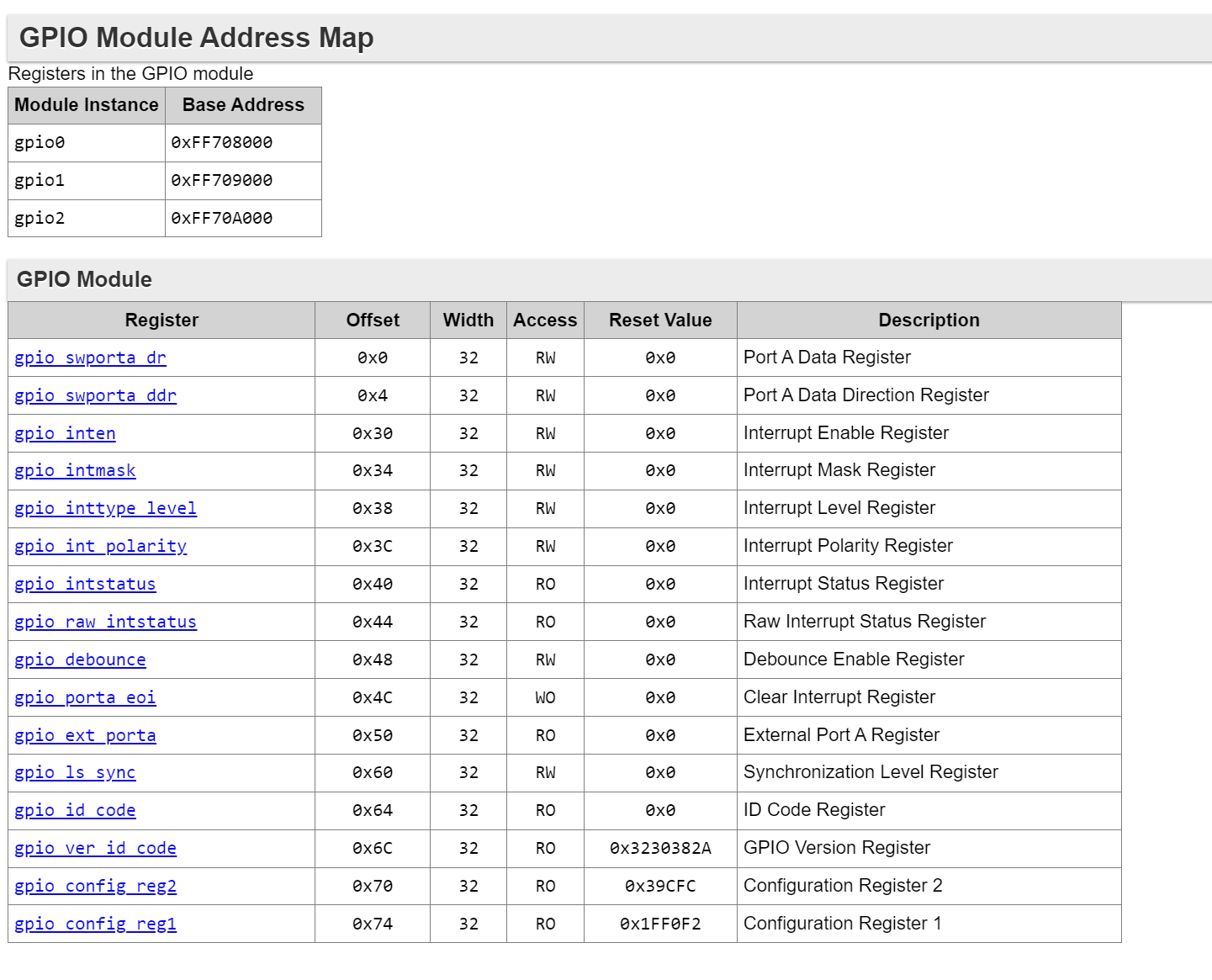
Essi sono pertanto visibili ad un opportuna locazione di memoria e controllabili/osservabili attraverso una certa procedura.

Informazioni relative sulle varie periferiche di HPS ed in particolare sulle periferiche GPIO sono reperibili nei documenti:

* Cyclone® V Hard Processor System Technical Reference Manual
* Cyclone V HPS Register Address Map and Definitions

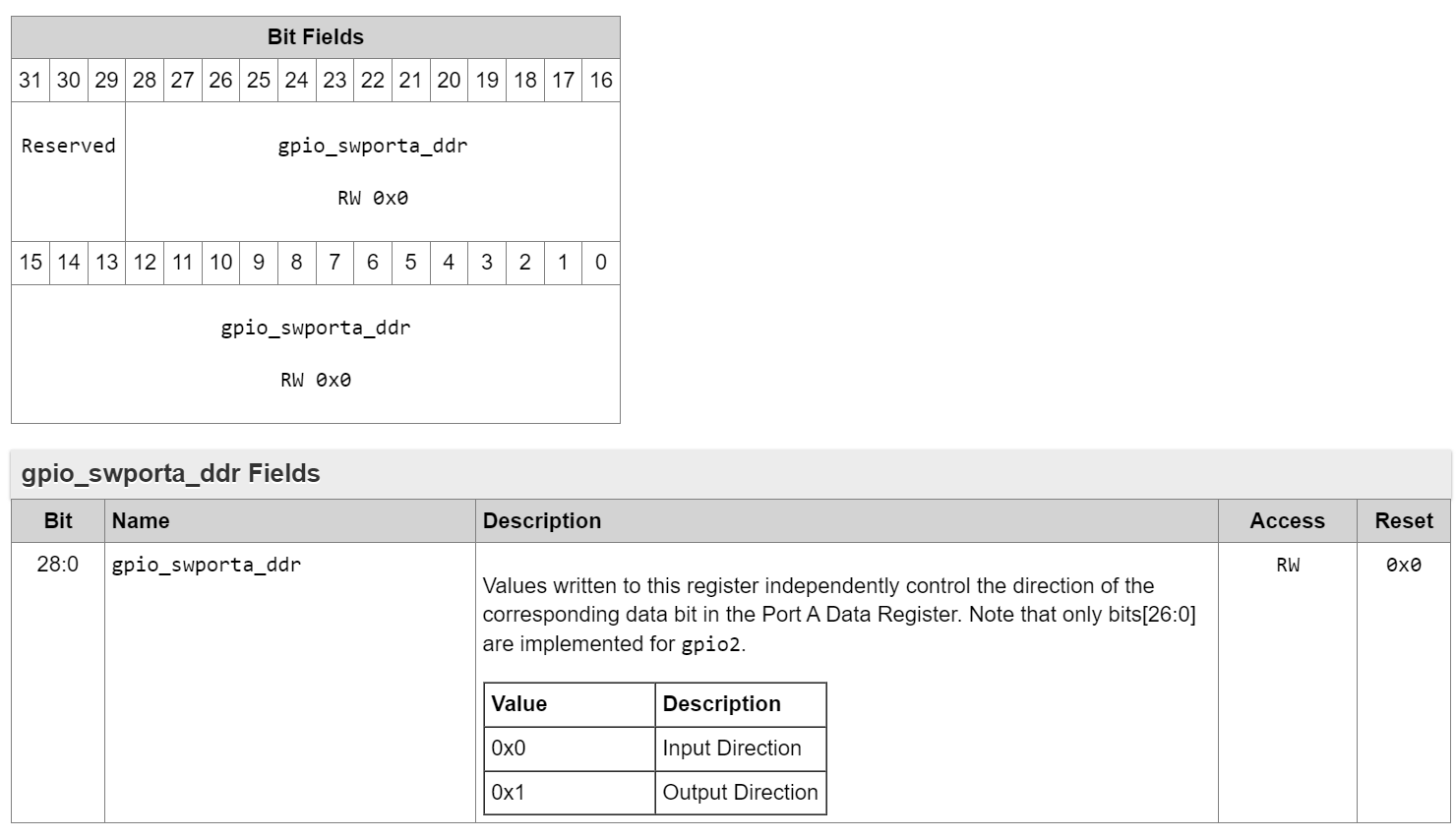


In particolare in essi si trova che i diversi piedini di GPIO accessibili attraverso HPS sono raggruppati in tre diversi banchi il cui driver è accessibile in una precisa locazione di memoria



Mentre i vari registri utilizzati per controllare il driver sono accessibili in locazioni di memoria attigue.

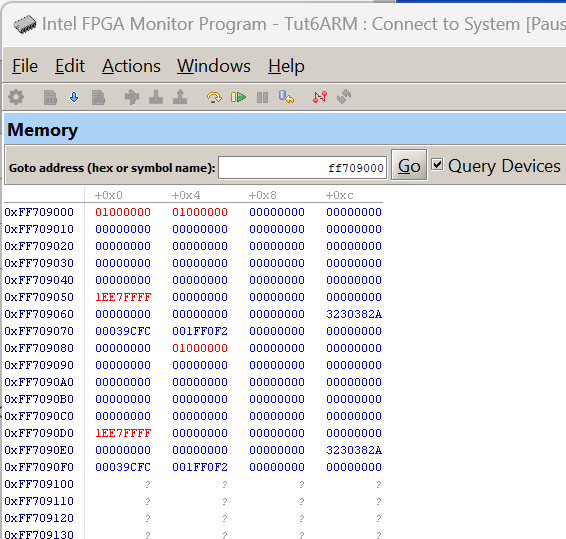
Prendendo ora in esame il LED, connesso al HPS\_GPIO53, si può desumere che esso è accessibile attraverso il driver gpio1 mappato alla locazione 0xff709000 e successive.Dalla documentazione si deduce che per accendere il LED si debba innanzitutto definire la direzione della porta di I/O relativa a HPS\_GPIO53 e configurarla come uscita.



Ciò può essere fatto attivando il bit di posizione 24 all’indirizzo 0xff709004 e successivamente attivando il bit 24 all’indirizzo 0xff709000 (ovvero scrivendo in entrambe le locazioni il valore 0x0**1**000000).

Si noti innanzitutto che questa operazione modifica lo stato del bit 24 anche alla locazione 0xff709050 (dedicata alla lettura dei GPIO) che passa dal valore 0x1**F**E7FFFF al valore 0x1**E**E7FFFF (dopo aver attivato la modalità output ma col bit24 ancora a 0) per poi tornare al valore 0x1**F**E7FFFF dopo aver acceso il LED.

Mentre la lettura dello stato del pulsante può essere operata similmente in base al valore assunto dal bit di posizione 25 sempre nella locazione preposta alla lettura del GPIO 0xff709050 il cui valore passa da 0x1FE7FFFF quando il pulsante non è premuto a 0x1DE7FFFF quando si preme il pulsante.

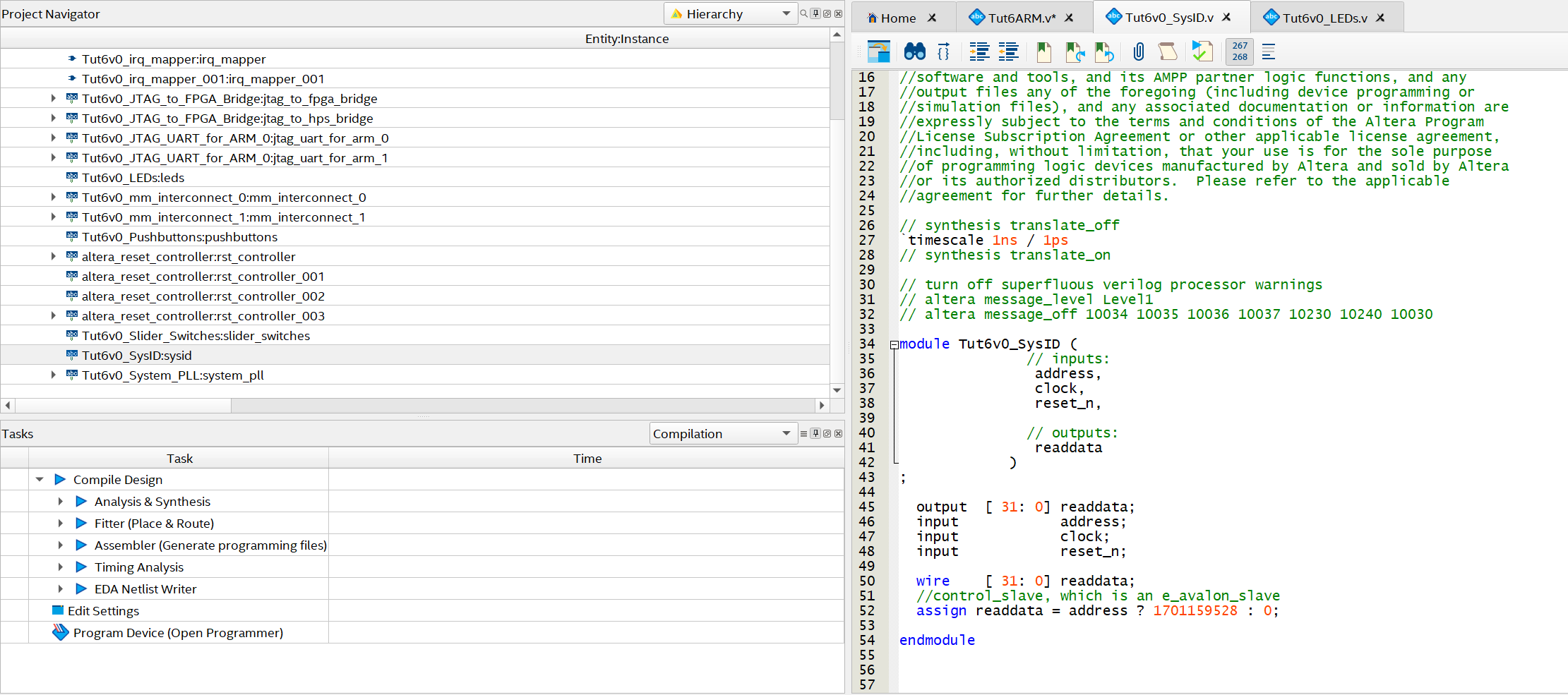


### Realizzazione di periferiche generiche da interfacciare col processore.

Così come sono state inserite nell’architettura del processore delle periferiche dedicate, sviluppate da terze parti, reperibili nell’ “IP Catalog” del tool “Platform Design”, così si possono integrare nel processore delle periferiche Custom realizzate ad hoc per gestire problemi specifici. Tali periferiche utilizzaranno come mezzo di comunicazione verso il processore il bus “AvalonMM” e verso il mondo esterno dei collegamenti di tipo “conduit”. Per trarre spunto su come scrivere una periferica dedicata si può trarre spunto dai codici relativi alle periferiche finora impiegate.

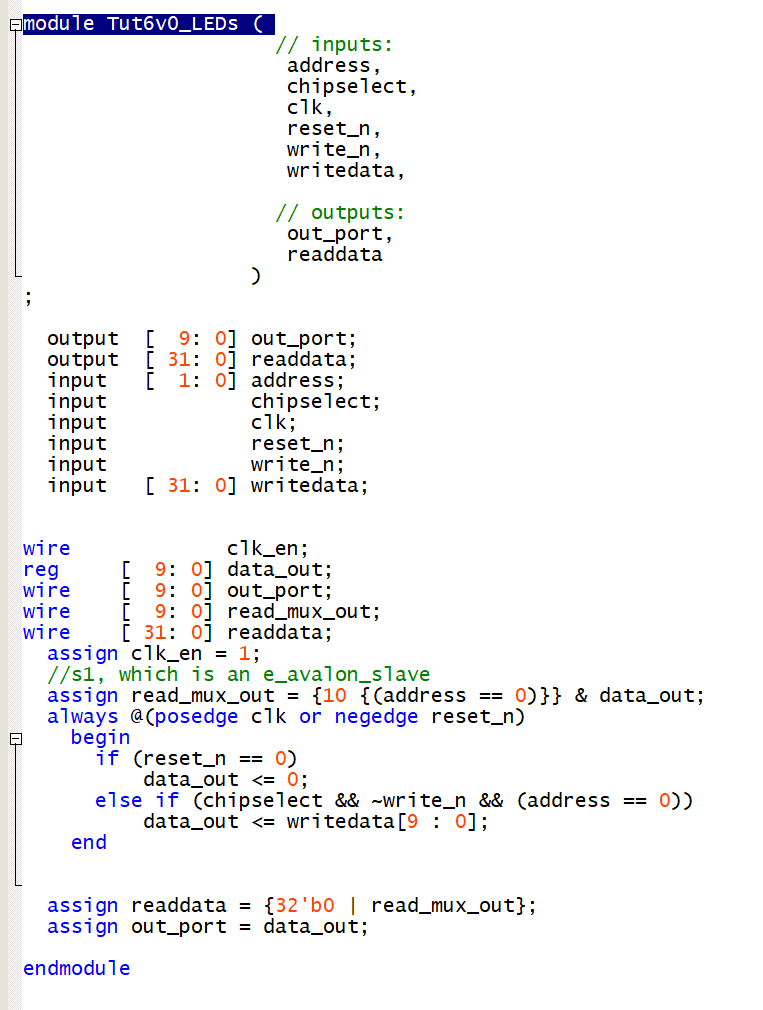
Si apra Quartus importando il progetto precedente.

All’interno della finestra “Project Navigator” messo in modalità “Hierarchy”, aprendo i sottoblocchi costituenti il processore sviluppato al passo precedente vi si riconoscono taluni blocchi di interfaccia (Led, Switch, SySID). Cliccando su di essi si accede al loro codice che può essere utile come ispirazione per vedere quali siano i segnali coinvolti e come essi vengano gestiti per ottenere i risultati dedicati.



Soffermandoci sul SysID si nota che i segnali con i quali il processore interroga tale periferica sia essenzialmente il bus di indirizzo (che in questo caso è un singolo bit) e che quando questo bit è posto a 1 l’interfaccia interpreta questo valore come un interrogazione, fornendo di conseguenza sul bus “readdata” il valore stesso del SysID.

Un po’ più complesso è la periferica destinata ad accedere alla fila di LED. Questa prevede qualche segnale di controllo in più che consenta tanto di scrivere quanto di leggere sulla periferica, quanto anche il segnale che va a collegarsi fisicamente ai LED (*out\_port)*

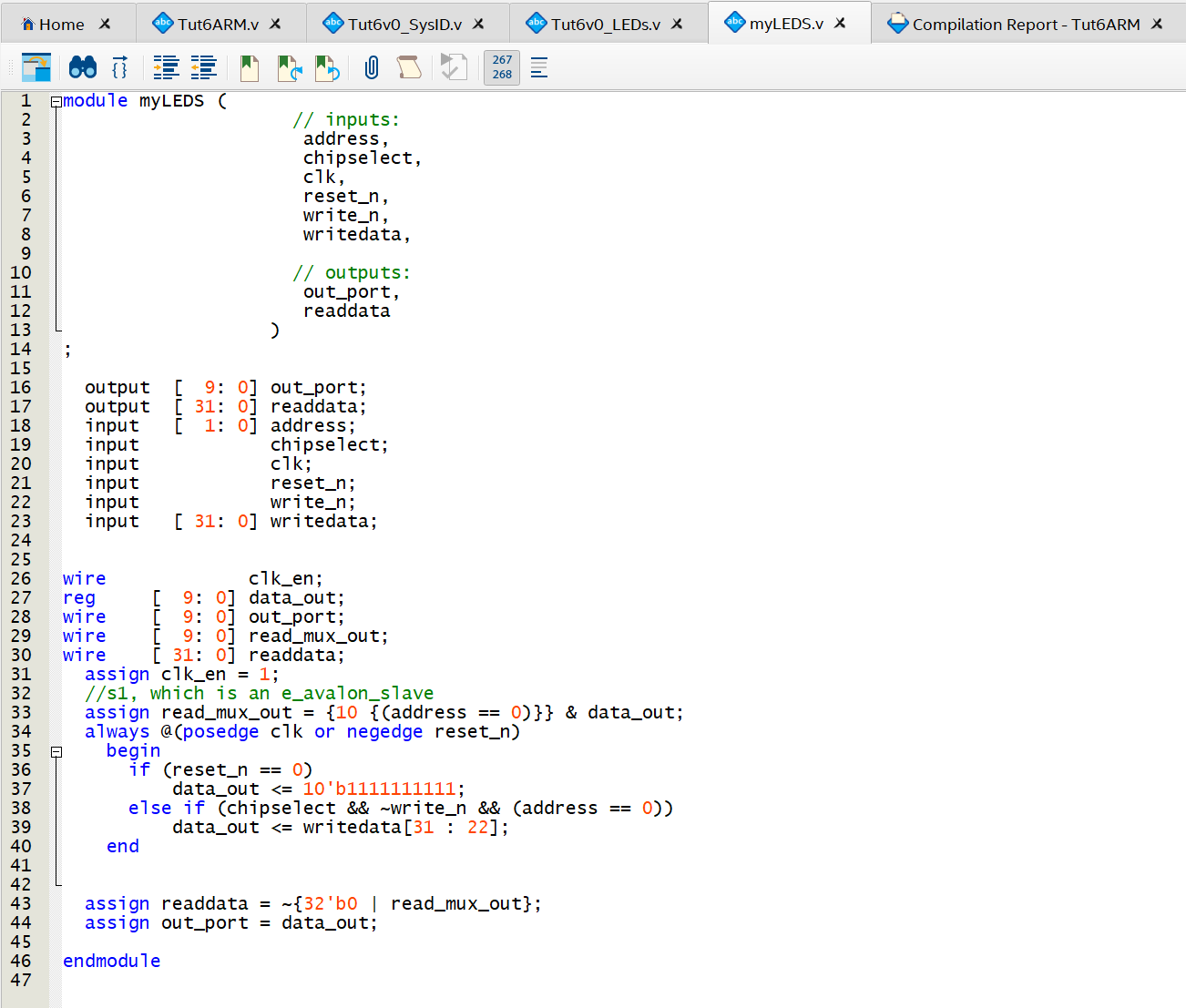


Fondamentalmente per poter “scrivere” sui LED il sistema verifica che sia attivato “*chipselect*”, che il segnale di “*write\_n*” sia appunto al livello basso (indicando l’operazione di scrittura) e che “*address*” sia pari a zero, il tutto ovviamente sincronizzato sul clock e con un opportuno segnale di reset. Quando tutte le condizioni sono verificate il dato presente sulla linea writedata (di 32bit) relativo ai bit [9:0] viene riportato sulle linee intermedie “*data out*” e da questa ad “*out\_port”.* Al contempoi dati che il processore viene a leggere sul bus *readdata* sono quelli *read\_mux\_ou*t estesi a 32bit con l’aggiunta di un numero opportuno di zeri. Ove il segnale *read\_mux\_ou*t stesso è composto (se l’indirizzo è pari a 0) dal segnale intermedio “*data out*” appena calcolato, altrimenti esso sarà nullo.

Una pratica sicuramente interessante da un punto di vista didattico-educativo potrebbe essere quella di modificare questa periferica per crearne una leggermente diversa da sostituire con quella impiegata nella parte precedente del tutorial.

Ad esempio si realizzi una periferica dove il valore col quale controllare i LED risieda nelle posizioni più significative del registro che ne controlla l’accensione (anziché nelle meno significative), il segnale di reset anziché spegnere i LED, li accenda e dove il valore letto sia l’opposto (ovvero la negazione bit a bit) di quanto riporterebbe nel caso attuale.

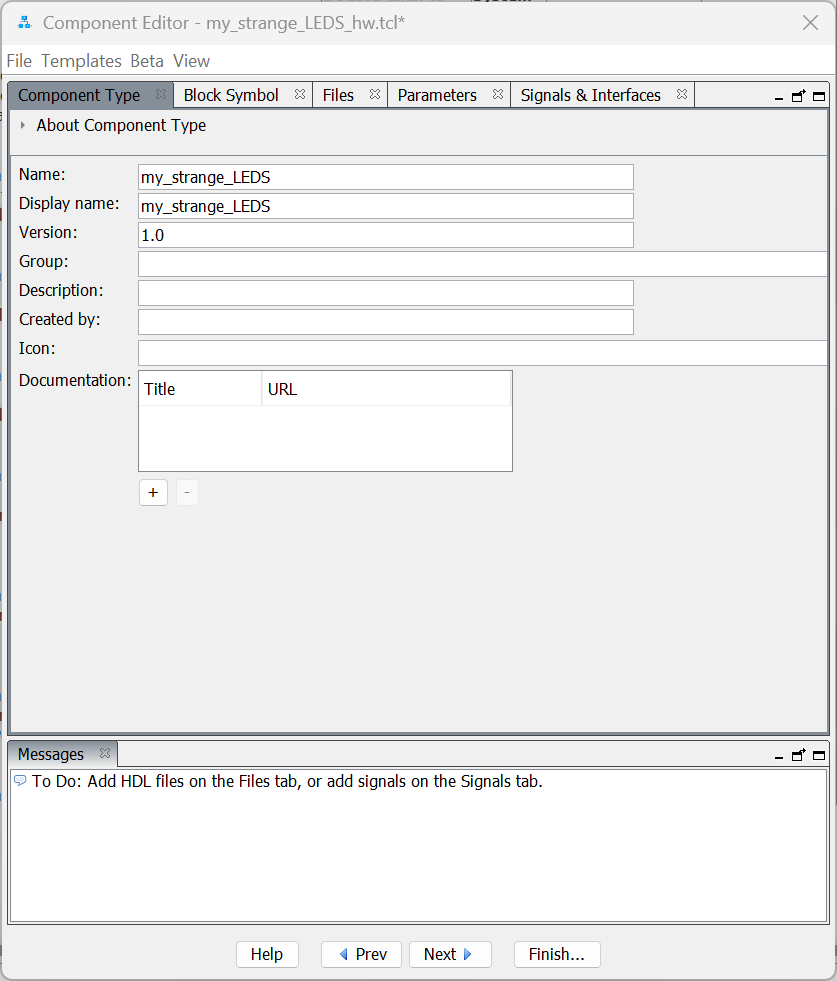
Si crei un nuovo file Verilog e vi si ricopi la struttura della periferica appena analizzata. Si modifichi il nome del modulo e si operino le variazioni suggerite rispettivamente alle righe 37,39 e 43



Si salvi opportunamente il file.

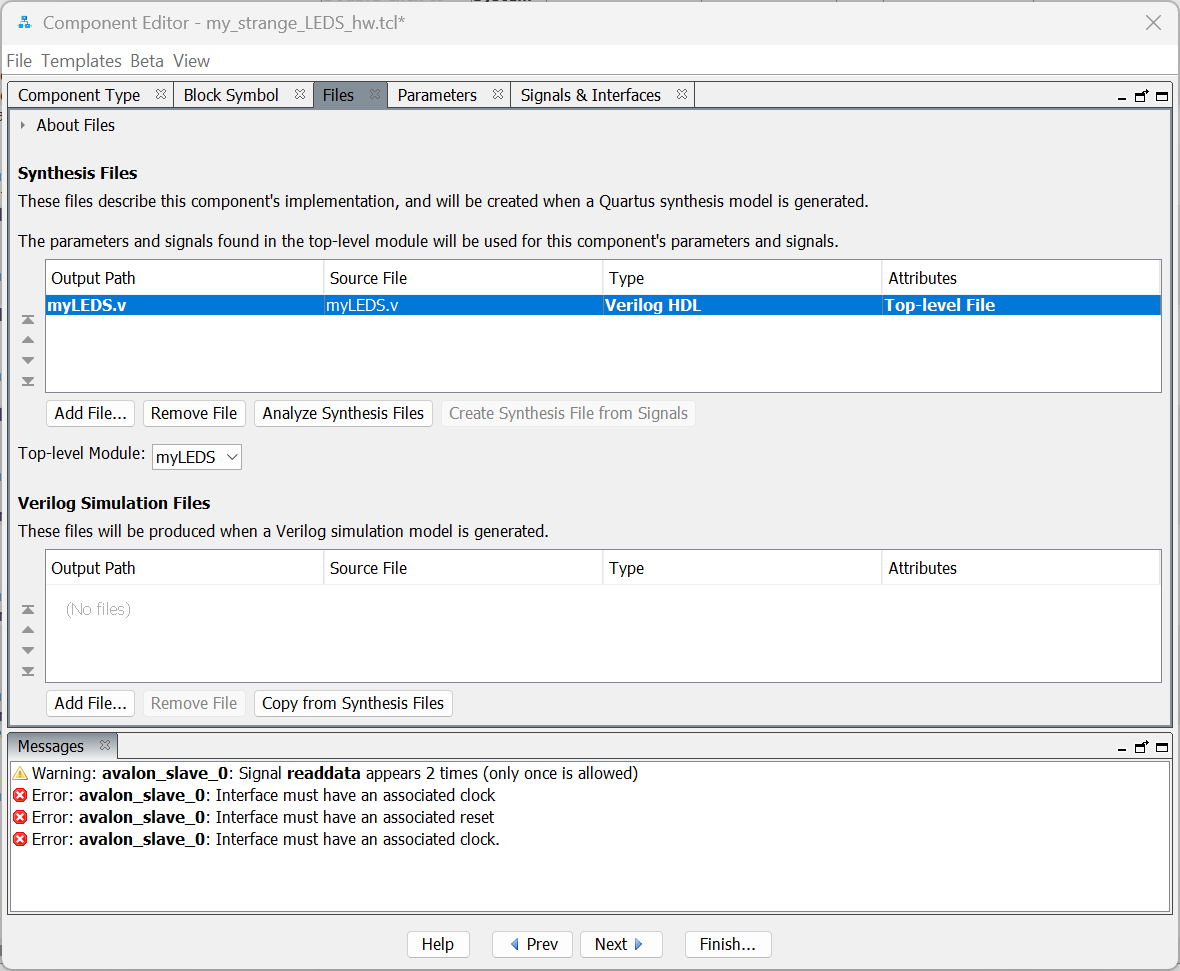
Si apra quindi il tool “Platform Design”

Nella Finestra di sinistra si operi un doppio click su “New Component”

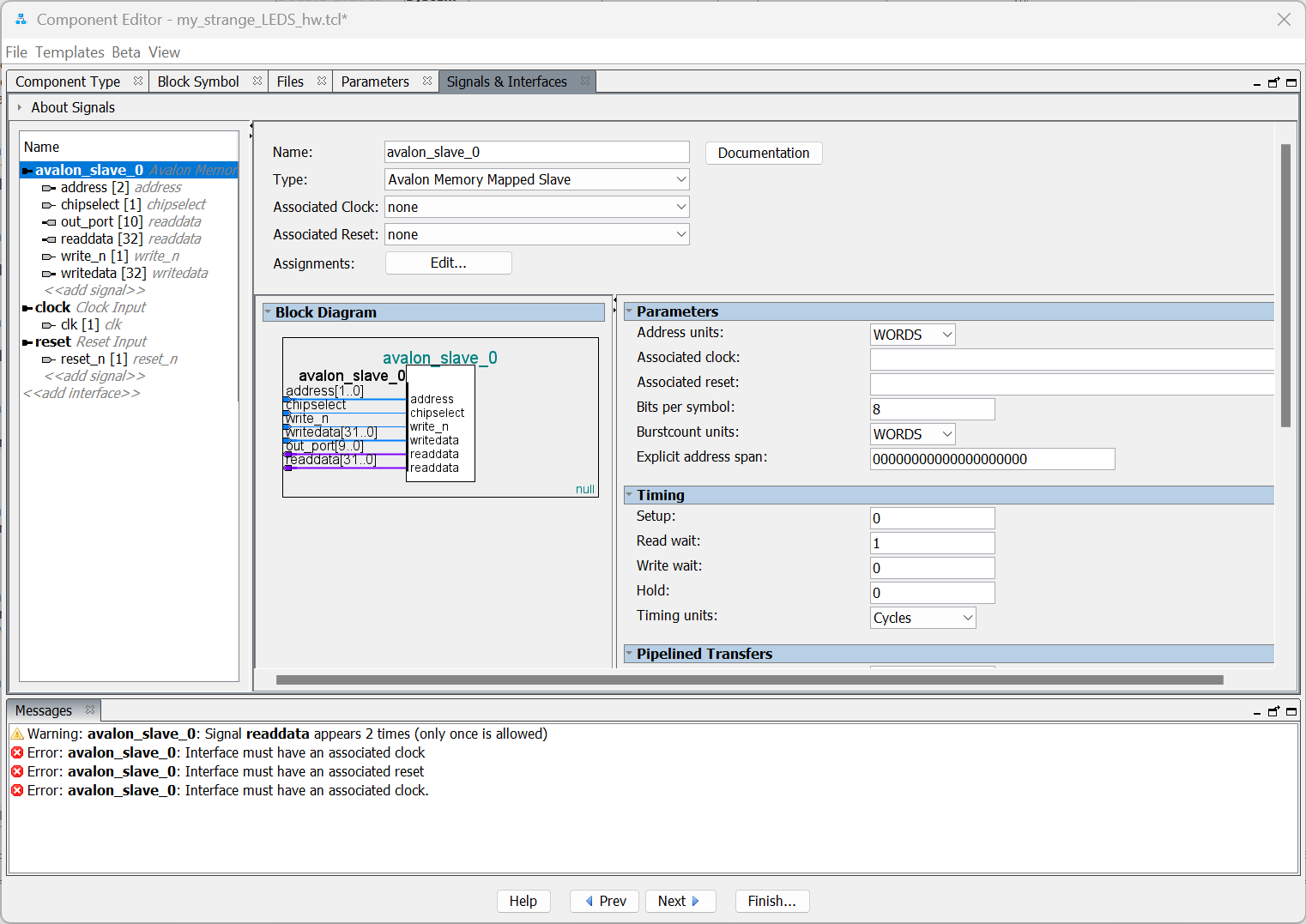


Nella finestra si fornisca un nome al componente (interfaccia) che si va a realizzare.

Nella scheda Files, cliccando su <AddFile> si aggiunga il File appena realizzato e successivamente si Clicchi su <AnalyzeFile>. Qualora vi fossero errori sintattici qui dovrebbero comparire messaggi di errore



Compaiono invece alcuni errori relativi alle Interfacce. Ci si sposti quindi nella scheda relativa alle interfacce e si nota come il sistema abbia già in automatico riconosciuto, grazie ai loro nomi alcuni segnali di interfaccia ed abbia ad essi associato interfacce specifiche. Se si usassero altri nomi, è plausibile che l’assegnazione delle interfacce e della tipologia di segnali potrebbe dover essere fatta a manualmente.

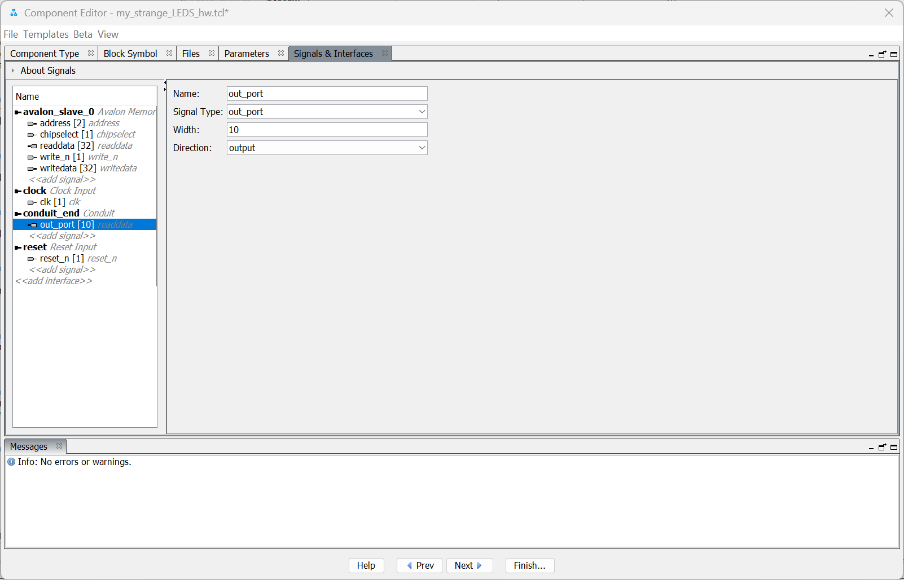


Ci sono rispettivamente al momento tre interfacce:

* avalon\_slave\_0 (con diversi segnali)
* clock
* reset

Si noti però che la porta out\_port è stata (erroneamente) riconosciuta come bus dati in lettura

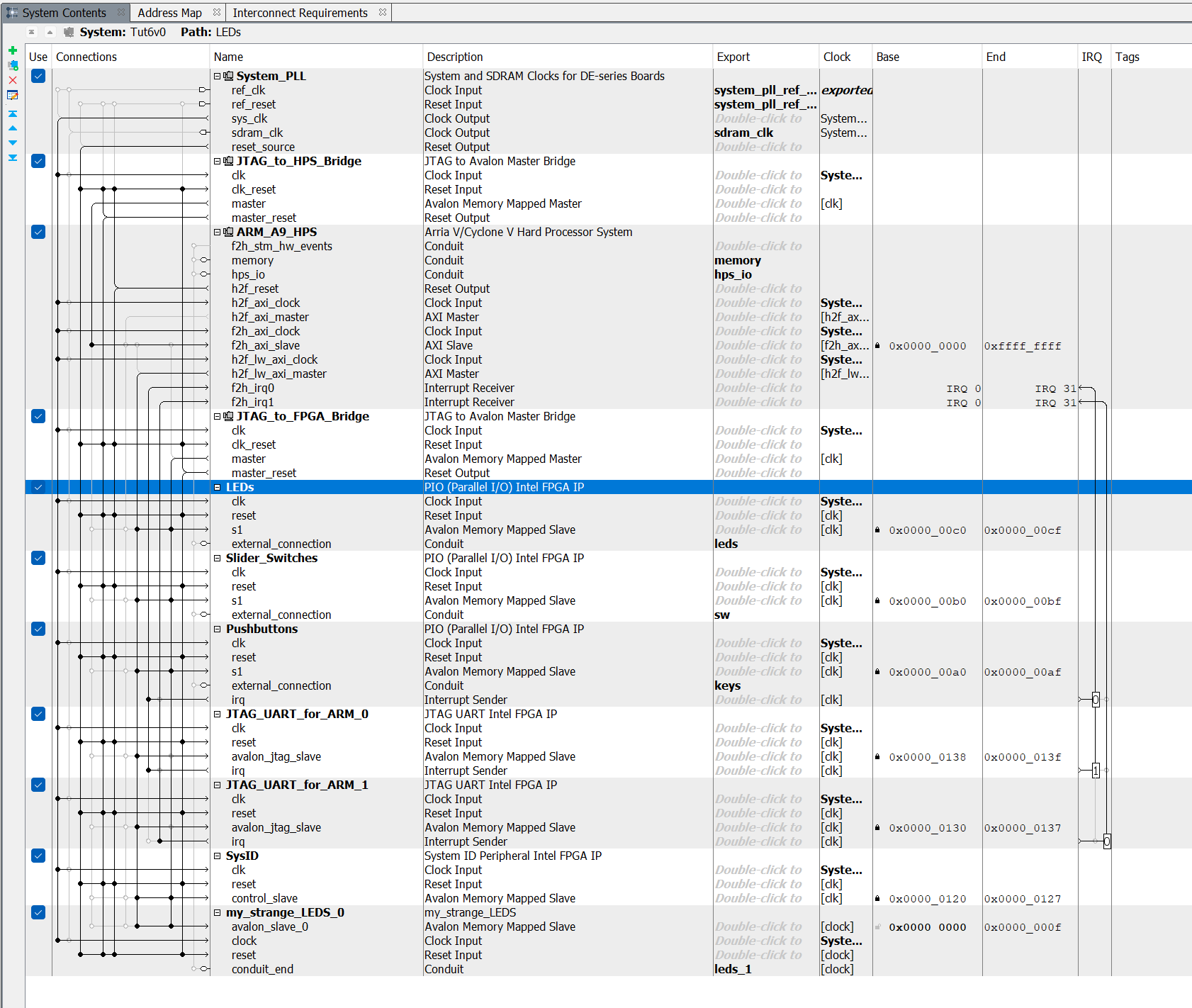
Bisogna pertanto Creare una nuova interfaccia di tipo “Conduit”, su di essa trascinare il segnale port\_out



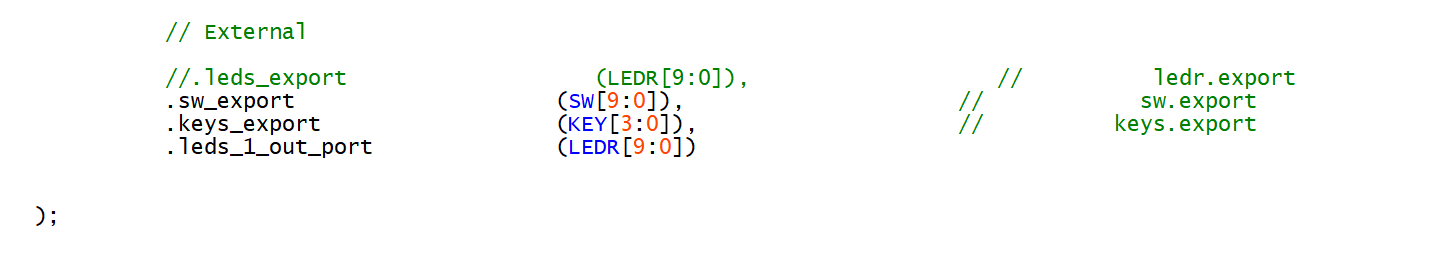
E nella finestra di destra stabilire la “tipologia” di segnale, ad esempio creando un nome ad hoc, in questo caso out\_port (che poi sarà il nome col quale il segnale verrà localizzato all’interno dell’istanza).

Inoltre cliccando sulle varie Interfacce stabilire quali siano i segnali di reset e di clock che le controllano.

Una volta ultimato il procedimento il nuovo componente sarà disponibile tra le interfacce da includere nel processore. E potrà essere integrato nell’architettura. Nell’immagine che segue il nuovo componente viene aggiunto a quelli già presenti, opportunamente collegato all’AvalonMM ed i segnali di interfaccia verso l’esterno esportati e rinominati.



A questo punto il processore può essere generato ed istanziato a livello di Top Level Entity, ma alterando i collegamenti in modo da rispecchiare il collegamento dei leds con questa nuova interfaccia:



Operando il download sulla scheda si noterà subito che in fase di reset tutti i led risultano accesi, inoltre i dati visualizzati sui led sono relativi questa volta ai bit più significativi e la lettura che avviene automaticamente dopo la scrittura restituisce un valore diverso da quanto scritto.

Nell’esempio sotto riportato, alla scrittura di 0xAAAAAAAA si nota che il valore letto simultaneamente risulta essere 0xFFFFFD55, ovvero i 22 bit più significativi sono posti tutti a 1, i 10 bit meno significativi sono riportati invertiti e sulla barra di led nella board si accendono i led in modo alternato.

## Realizzazione di altre periferiche da interfacciare col processore.

Si voglia corredare il processore di tre ulteriori periferiche: una che piloti il display a sette segmenti e visualizzi un codice esagesimale in base ai dati ricevuti dal processore, un timer che incrementi il valore ogni microsecondo, ma con la possibilità dio impostare il valore di partenza, un driver per encoder rotativo che incrementi/decrementi il valore di un registro in base agli step eseguiti dall’encoder in un senso o nell’altro

I codice di queste periferiche sono disponibili sulla pagina moodle del corso

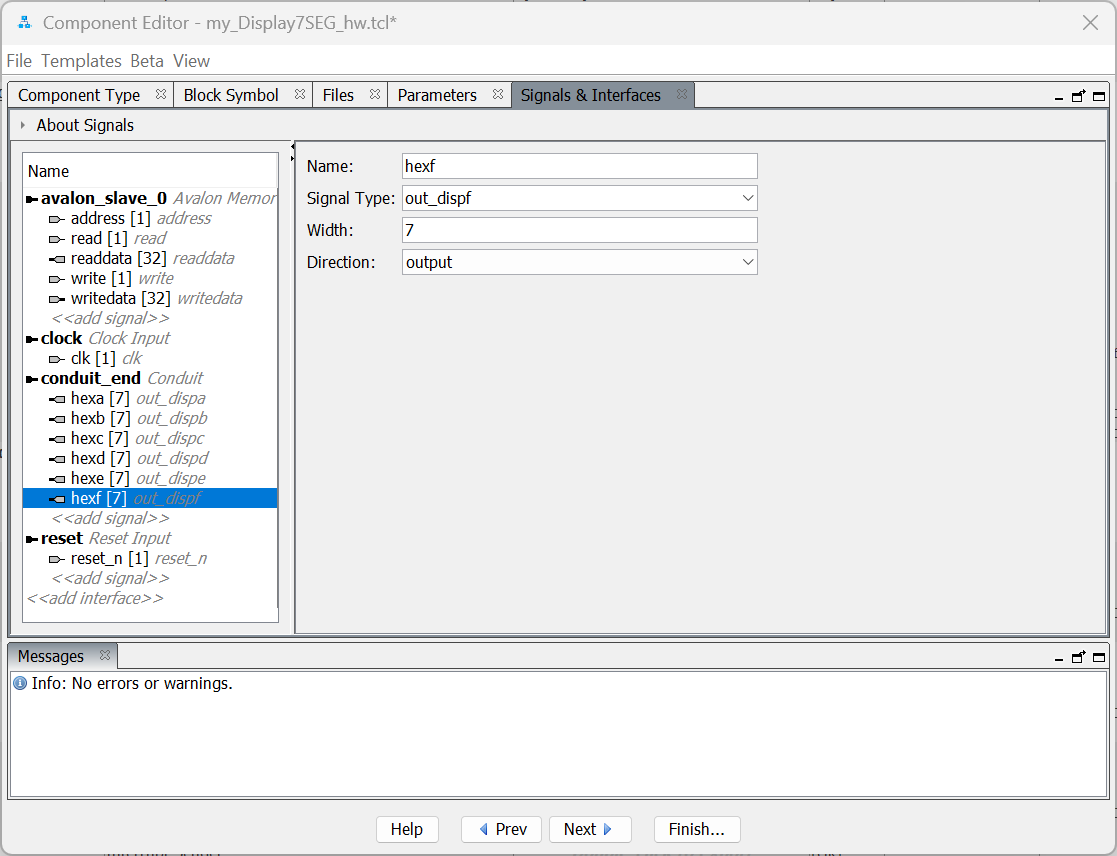
#### Display a 7 segmenti

Il codice si compone di due moduli:

* SEG7\_LUT è un circuito sincrono che riceve in input un segnale a 4 bit e genere in uscita i 7 segnali per visualizzare la cifra corrispondente su un singolo display
* my\_led7seg è un sistema a più alto livello che quando riceve il segnale write salva i dati in ingresso in un registro interno (reg\_data). I dati di questo registro, raggruppati 4 a 4 forniscono l’input per il blocco precedente. Inoltre forniscono il dato in lettura quando la linea read è attiva. In uscita ci sono 6 bus separati a 7 segmenti, uno per ciascun display.

Si importi il componente all’interno del “Platform Design” seguendo la procedura definita precedentemente: aggiungendo il file Verilog ed analizzandolo, ma questa volta, poiché il file contiene due moduli bisogna definire nella medesima finestra quale è il modulo più alto in gerarchia (ovviamente “my\_led7seg”).

Nella scheda “interface” creare una nuova interfaccia di uscita nella quale portare i 6 bus destinati ai 6 displays e a ciascuno di essi dare un “Type” univoco. Inoltre collegare clock e reset a tutte le interfacce che lo necessitano.



#### Timer

Questo componente non necessita nessun collegamento al di fuori del processore, ma può essere molto utile per stimare il tempo intercorso tra due punti di esecuzione del programma. Esso è sostanzialmente un contatore che viene incrementato a cadenza costante Inoltre attraverso il parametro SUBSAMP si può stabilire la cadenza con la quale il contatore viene incrementato.

Come fatto precedentemente

* si crei un nuovo componente
* si importi il file Verilog di descrizione e lo si analizzi
* si configurino le interfacce
* sulla scheda parameters viene evidenziato come il componenete sia stato realizzato lasciando regolabili 3 parametri : DATA\_WIDTH, ADDR\_WIDTH, SUBSAMP. Ma siccome per un corretto interfacciamento al processore è necessario che il bus dati sia a 32 bit si può deselezionare tale parametro dai parametri regolabili. Anche il ADDR\_WIDTH è piuttosto ridondante per il funzionamento, ma lo possiamo lasciare per vedere come si comporta l’interfaccia se andassimo a modificarlo. Da ultimo SUBSAMP è necessario per settare la frequenza del timer.

#### Encoder rotativo

Questo componente è in realtà un contatore con la possibilità di Incrementare/decrementare il conteggio analizzando due segnali periodici in ingresso ed il loro sfasmento relativo. Il conteggio può inoltre essere settato ad un determinato valore usando l’operazione di scrittura sul bus avalonMM, mentre in lettura restituisce il valore del contatore stesso.

Come nel caso del Timer, anche questo componente prevede 3 parametri regolabili che però all’atto pratico modificano in minima parte il comportamento del sistema.

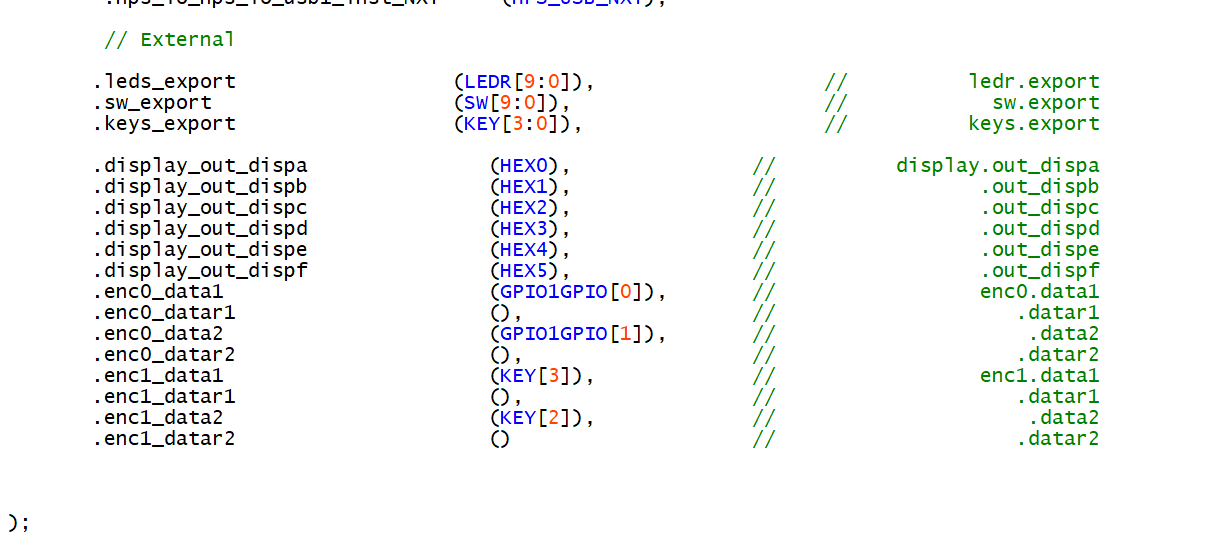
Come nei passi precedenti si realizzi un nuovo componente importando il file Verilog disponibile.

#### HPS complessivo

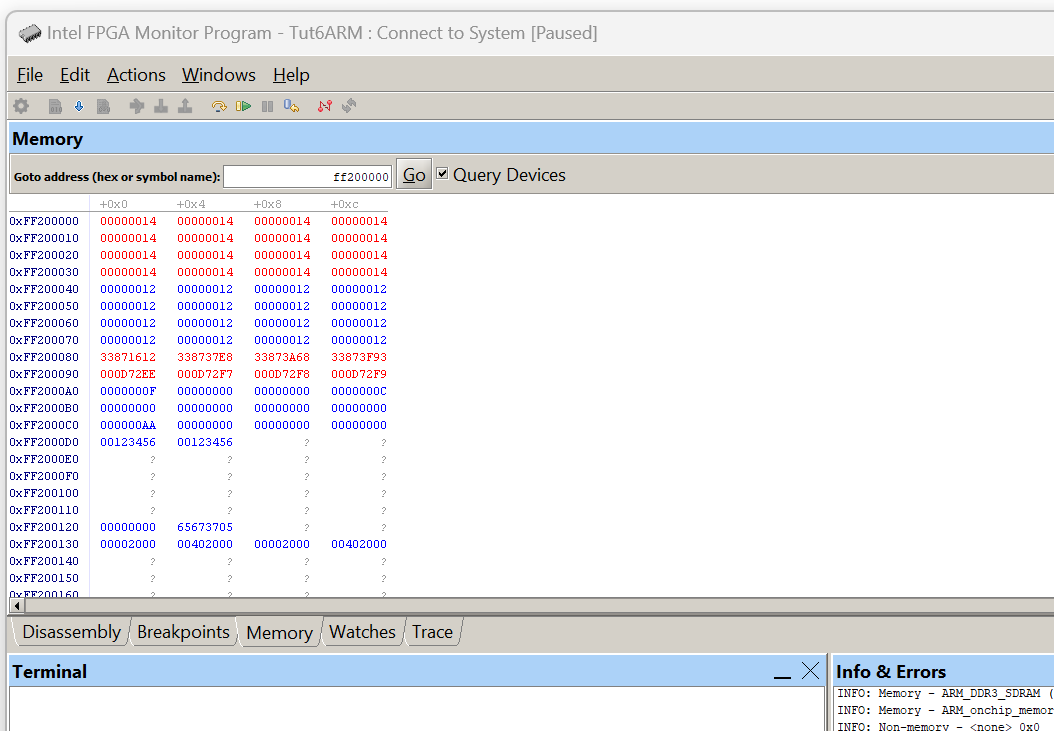
Si importino ora attraverso Platform Designer

* due Timers e si regolino i parametri in modo che uno abbia il periodo di 1millisecondo (SUBSAMP = 50000) e l’altro di un microsecondo (SUBSAMP=50)
* un display a sette segmenti
* due encoder rotativi – Ne vengono realizzati due in modo che mentre l’uno prende i segnali attraverso la GPIO da un reale encoder rotativo esterno, l’altro prenda i segnali dai pulsanti (KEY) in moda da emularne il funzionamento attraverso la pressione combinata dei tasti.

A Livello di Top-Level-Entity si istanzi il componenete aggiungendo tutte le opportune connessioni col mondo esterno.



Una volta compilato completamente il progetto ed eseguito il download su scheda vi si acceda attraverso “Altera Monitor Program” ed alle locazioni di memoria opportune vi si tovano tutti i dati delle interfacce



Si noti in particolare

* I dati relativi agli encoder rotativi il cui ADDR\_WIDTH è di default pari a 4 occupano di fatto 16 posizioni di memoria, sebben con dati tutti uguali
* I Dati relativi ai Timers sono diversi ad ogni lettura ed anche quando vi si scriva un dato il solo tempo intercorso tra scrittura e lettura fa sì che il dato si diverso da quanto scritto
* I dati in posizione 0xFF2000A0 sono relativi ai pulsanti
* I dati in posizione 0xFF2000B0 sono relativi agli interruttori
* I dati in posizione 0xFF2000C0 sono relativi ai LED
* Il dato scritto in posizione 0xFF2000D0 viene riportato nella stessa forma sul display a sette segmenti.

#### Software

A questo punto si può scrivere in C il codice di sistema che interfacciandosi alle periferiche realizzi la funzione desiderata.

<Continua>