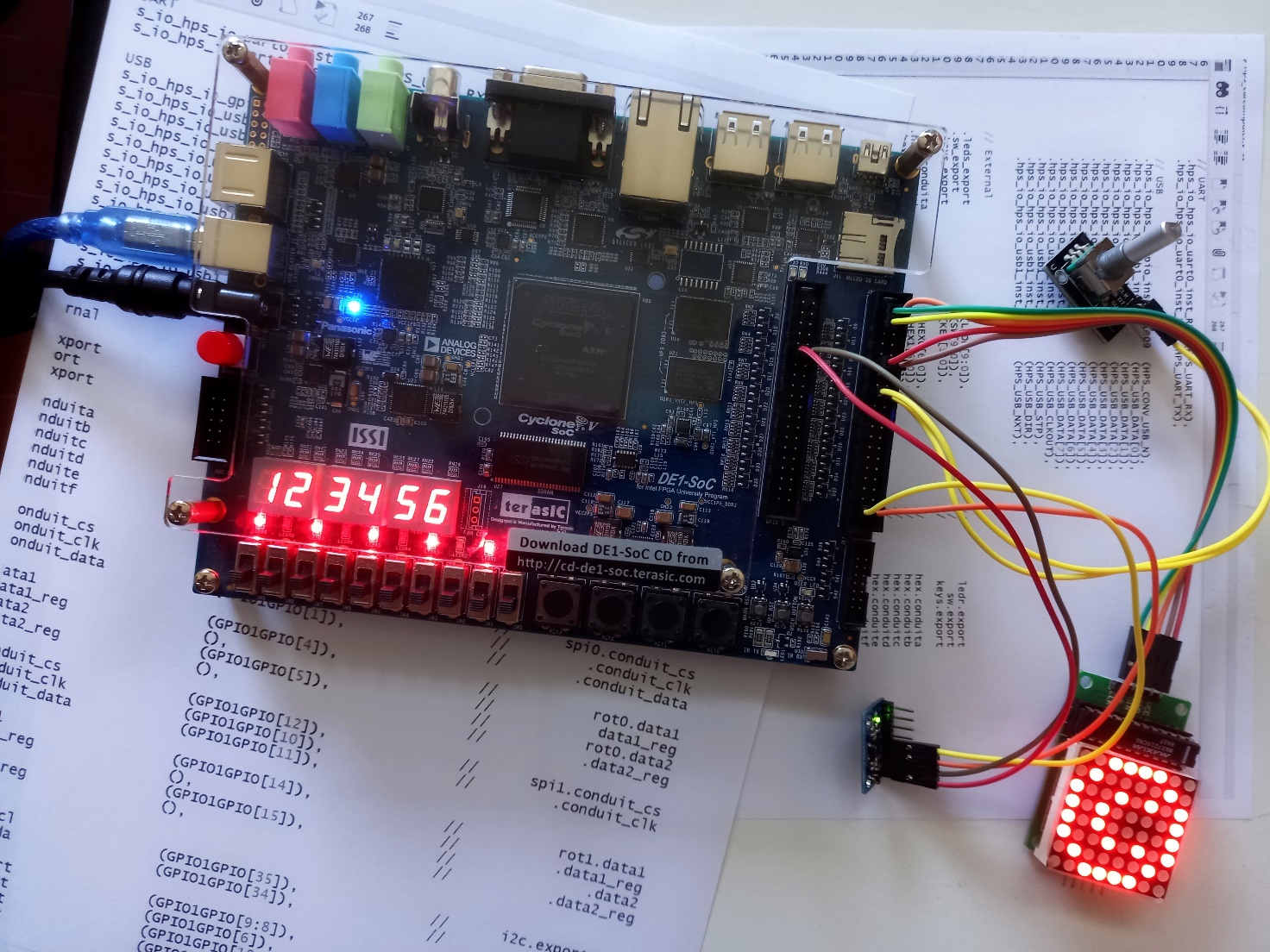
Progettazione di Sistemi Elettronici

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

Tutorial 1



**Introduzione allo sviluppo di un semplice progetto completo.**  
**Hardware** impiegato: Terasic DE1-SoC Board  
**Software** Impiegato: Quartus 22.1

Tutorial 1A

Realizzazione di un semplice circuito logico su DE1-SoC

Descrizione: Si realizzi un semplice circuito logico su DE1-SoC [] con ingressi pilotati dagli switches ed uscita su LED.

Scopo: familiarizzare col tool di sviluppo, con la scheda di sviluppo, apprendimento dei passi fondamentali da seguire per portare a termine un progetto completo.

Apprendimento previsto:

* Primi rudimenti per navigare all’interno del tool quartus
* Compilazione di un progetto descritto in forma schematica
* Compilazione di un progetto descritto tramite Verilog VHDL
* Imposizione di alcuni vincoli realizzativi (Pin position)
* Simulazione del circuito
* Visualizzazione del layout del circuito
* Download e running del circuito su scheda DE1

# Procedimento

1. Fase introduttiva

Si apra il tool quartus tramite icona sul desktop oppure dall’elenco dei programmi istallati

Si crei un nuovo progetto o ciccando sull’apposito tasto oppure tramite

*File > New Project Wizard*

*(Next)*

Si definisca

* il direttorio nel quale salvare i files di sistema (sconsigliato l’uso di spazi o caratteri strani – per cui evitare ad esempio direttori quali “My Documents”)
* il nome del progetto, e definire (provvisoriamente)
* il nome del blocco più alto in graduatoria del progetto (top level devign entity) – tipicamente questo prende il nome stesso del progetto e comunque può essere modificato in seguito.

(Next)

Ove si volessero importate files o settaggi da progetti precedenti si può agire su questa pagina usando il tasto Use Exsisting Project Settings. In questo primo esempio si può passare alla pagina successiva

(Next)

Si definisca se il progetto viene sviluppato ex-novo oppure da template pre-caricato. In questa prima esercitazione iniziamo un progetto vuoto

* Empty Project

(Next)

Eventualmente si possono includere Files pre-esistenti che in questo primo caso NON sono stati ancora sviluppati

(Next)

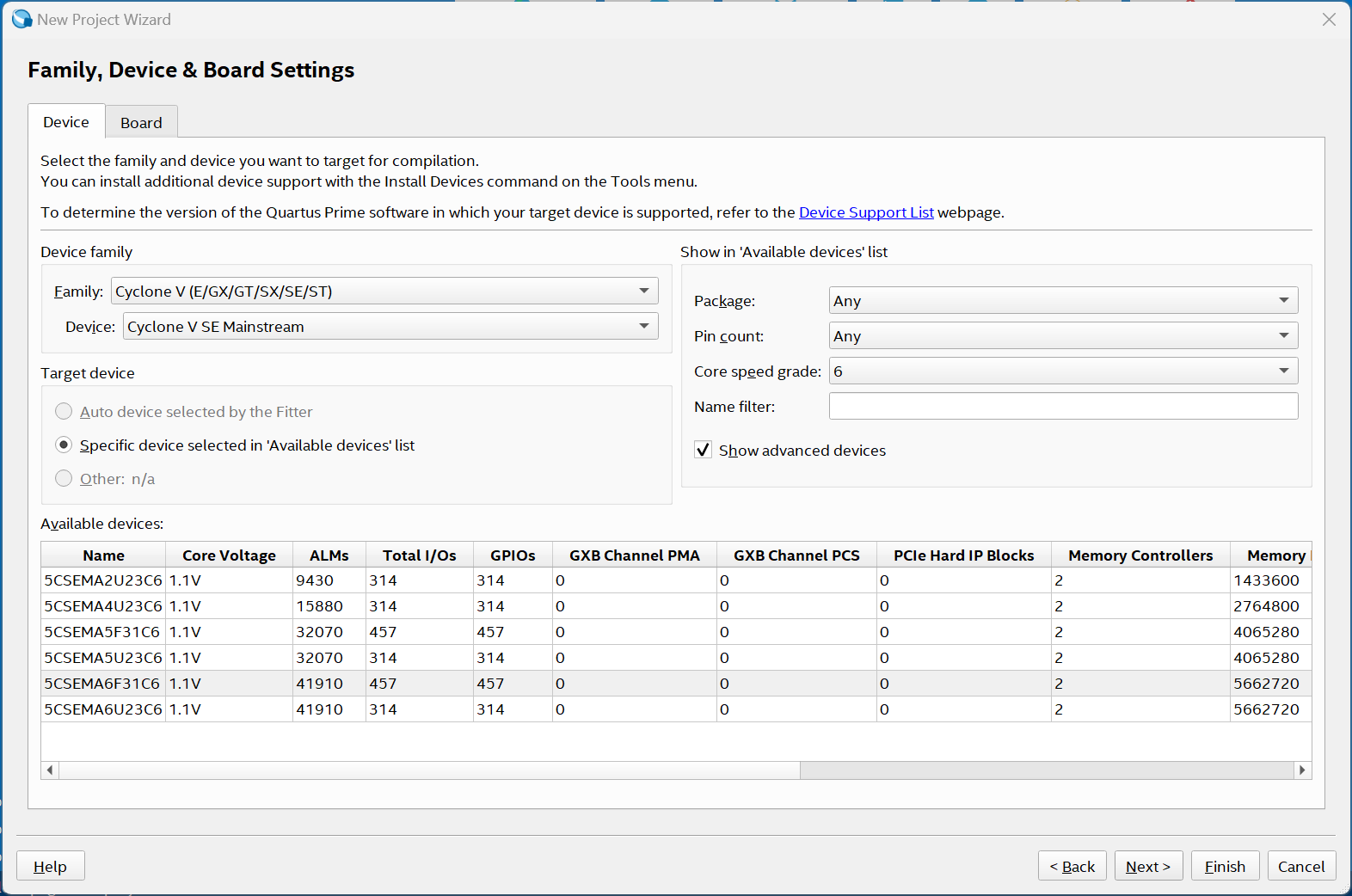
Si definisca il dispositivo (FPGA) per il quale sviluppare l’intero progetto. Ovviamente il progetto stesso, le risorse disponibili e la struttura stessa dipendono dal dispositivo utilizzato – esistono centinaia di FPGA diverse per famiglia, caratteristiche, prestazioni, piedinatura ecc. **E’ FONDAMENTALE** scegliere l’FPGA corretta [] . Nel nostro caso sulla DE1-SoC è montata una 5CSEMA5F31C6 della famiglia Cyclone V pertanto sceglieremo la:

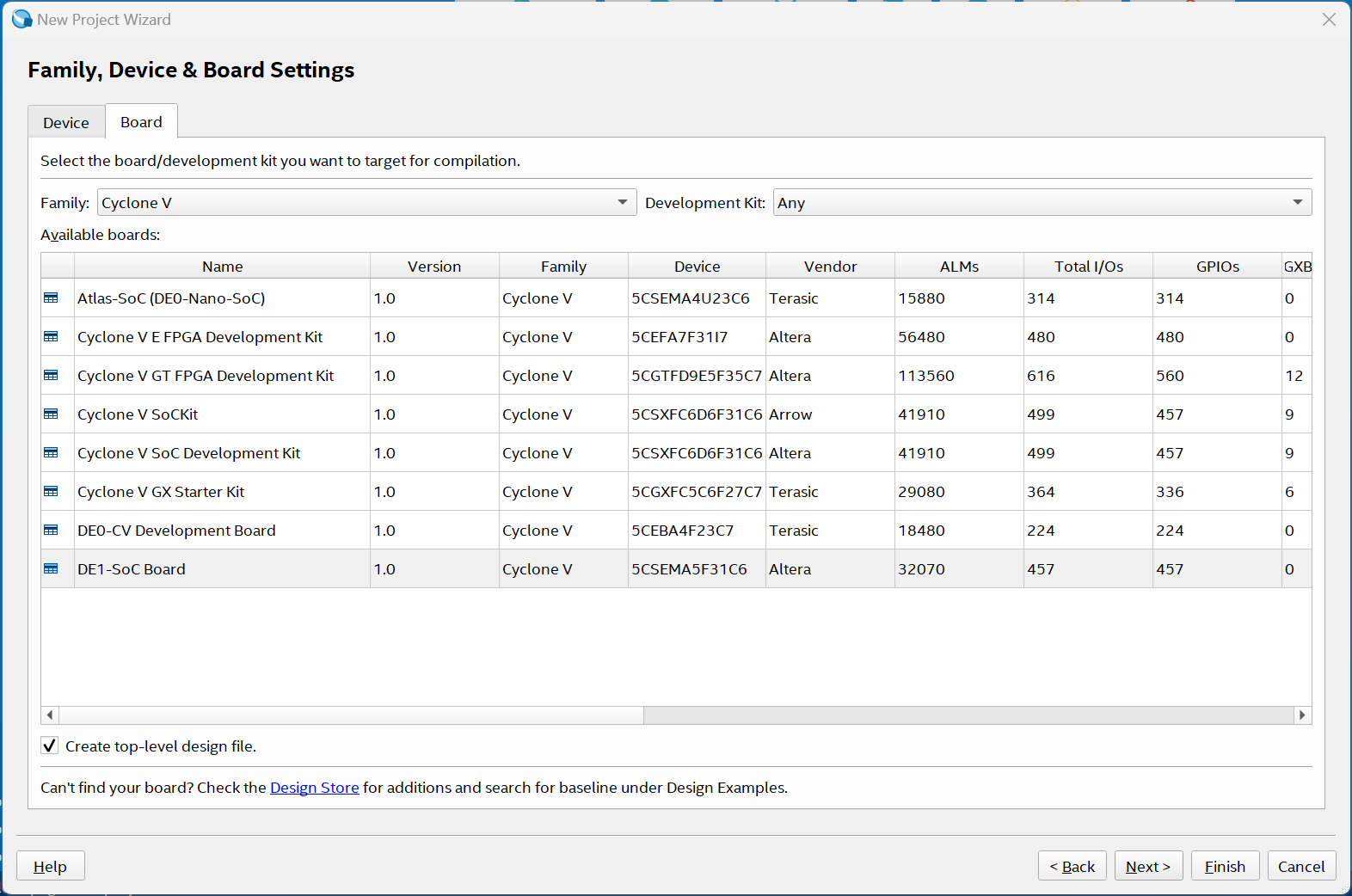
Family: Cyclone V (E/GX/GT/SX/SE/ST)

Device: Cyclone V SE Mainstream

* 5CSEMA5F31C6

Ci si può agevolare nella ricerca impiegando opportuni filtri, oppure usando la sottocartella Board (Disponibile ove si sia installato Intel Accademy University Program – che per alcune schede di sviluppo propone di default il dispositivo montato su di esse.

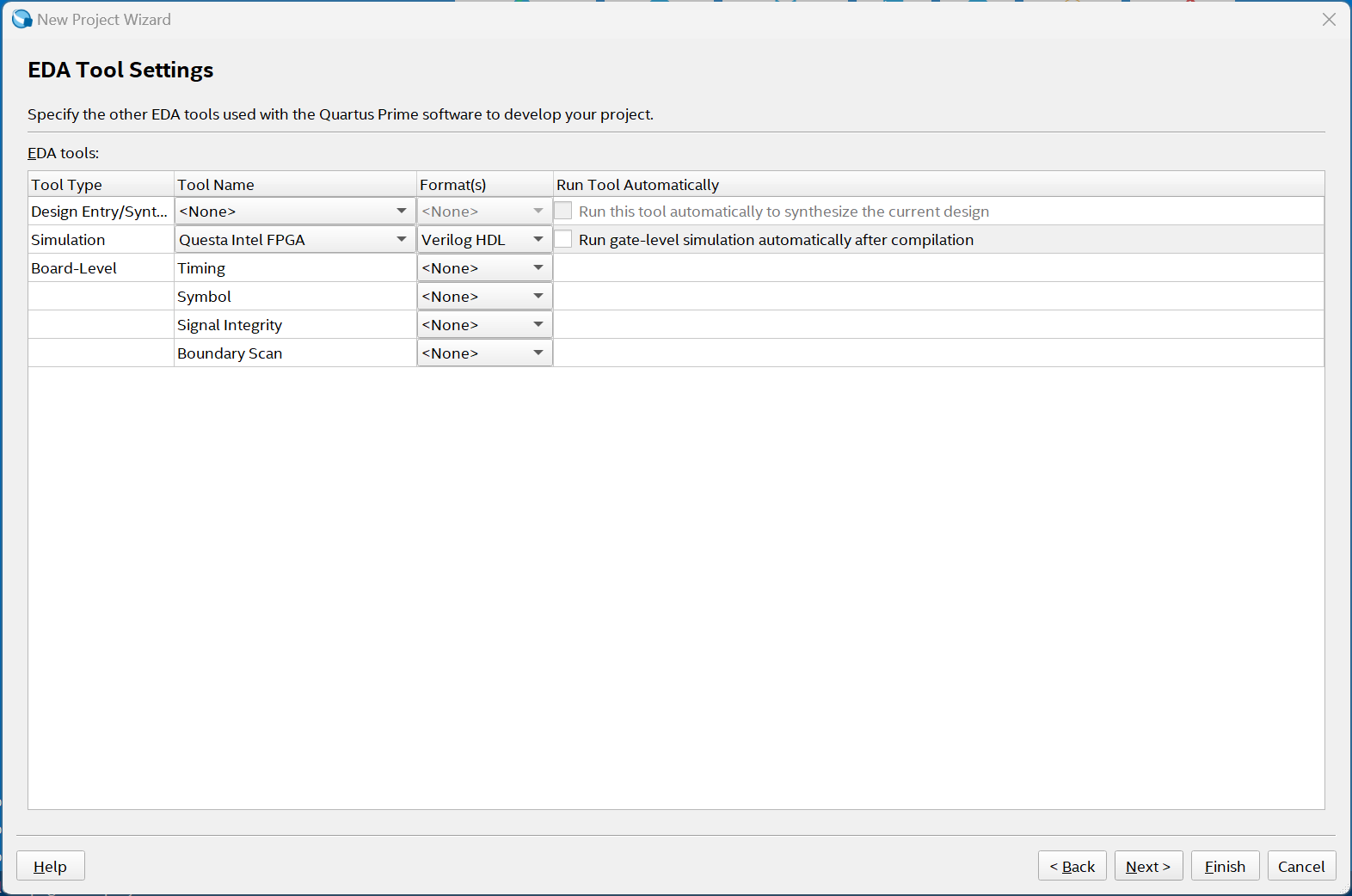




La tabella presente in questa pagina consente di paragonare le prestazioni dei vari dispositivi (celle logiche, memoria a disposizione, moltiplicatori integrati, PLL, risorse di I/O, linee di clock dedicato, Cores HPS ) … questo potrebbe essere utile laddove non sia stato definito ancora il dispositivo sul quale realizzare il sistema.

(Next)

In questa pagina si possono fissare programmi di terze parti da usare per la sintesi, la simulazione e l’analisi temporale. In questo caso si definisca come tool per la simulazione Questa Intel FPGA e come linguaggio da adottare Verilog HDL – disattivare “Run Gate-level simulation automatically after compilation” .



(Next)

Una pagina riassuntiva elenca le scelte fatte, onde verificarne la coerenza

(Finish)

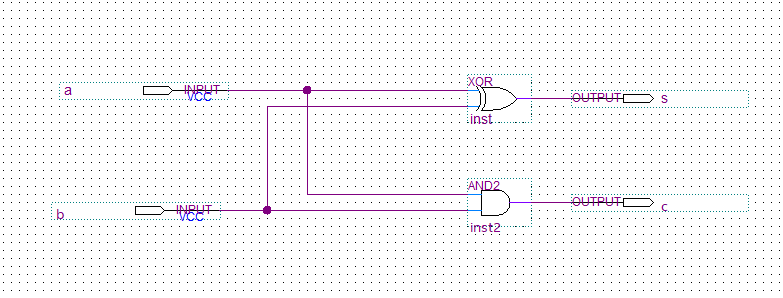
#### Definizione dell’ architettura sistema (Tramite schematico) [5-6]

File > New

Block Diagram / Schematic file (OK)

Supponiamo di voler sviluppare un “Half Adder” in grado di generare somma e riporto per due bit in ingresso

Nella nuova pagina grafica che compare, tracciare lo schematico del sistema da realizzare simile a quanto riportato in figura.

Si impieghino i tasti  e  per accedere alla libreria delle porte logiche e per creare i vari collegamenti. 

In particolare si suggerisce:

- di prendere visione delle diverse porte logiche e strutture hardware disponibili suddivise tra

Megafunctions: funzioni complesse ad alto livello di astrazione (verranno utilizzate in seguito)

Others: funzioni particolari legate ad una particolare famiglia di FPGA

Primitives: funzioni generiche

- di analizzare le varie opzioni grafiche disponibili (ad esempio quella di poter modificare la posizione dei gate mantenendo inalterati i collegamenti o viceversa spostare i gate senza i collegamenti)

- dare un nome mnemonico convenuto alle varie porte logiche e soprattutto ai pin di ingresso ed uscita.   
NOTA BENE: utilizzare dei nomi abbastanza specifici ed evitare possibili parole “riservate” quali ad esempio: “in”, “out”, “inout”, “and”, “inv”, …. , così come parole magari più improbabili come “for”, “while”, “goto” ecc. ecc.

Si salvi lo schematico e se ne verifichi la correttezza del circuito attraverso un processo di analisi.

Processing > Start > Analisys and Elaboration

Si noti che esistono eventualmente delle scorciatoie nella finestra dedicata ai Tasks.

Si correggano eventuali errori (un doppio click sul messaggio d’errore aiuta ad evidenziare la parte interessata)

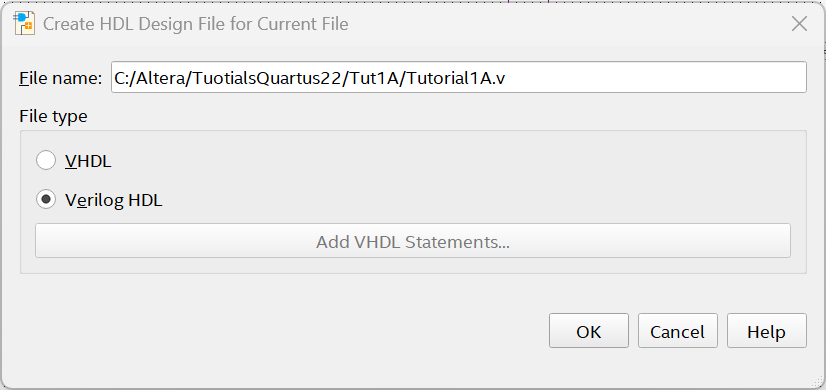
#### Simulazione funzionale

Nel corso degli anni si sono succedute diverse versioni di simulatori e di modalità per eseguire la simulazione del sistema da analizzare. In questo Tutorial vedremo come sviluppare la simulazione utilizzando un tool esterno a Quartus, ovvero Modelsim/Questa, rimandando però il lettore al tutorial specifico relativo alla simulazione per quanto riguarda la generazione dei segnali di stimolo ed alla visualizzazione dei risultati.

**Preparazione:**

Poiché il Tool di simulazione utilizza in ingresso sistemi descritti in forma testuale attraverso un opportuno linguaggio HDL (VerilogHDL o VHDL) la prima operazione è trasformare il sistema finora generato nel suo equivalente descritto in HDL.

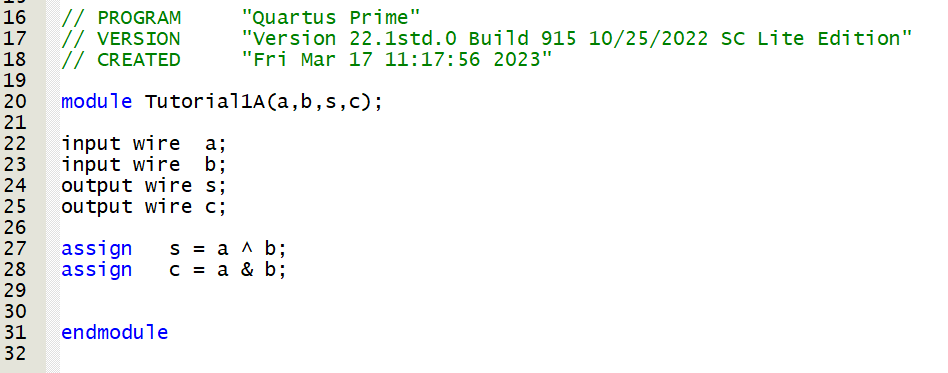
File > Create/Update > Create HDL Design File from current File



Si scelga un nome per il file di uscita

(OK)

Cercando il file generato nella cartella si trova un file più o meno simile a questo:



Che descrive testualmente il semisommatore sviluppato in maniera alternativa allo schematico

Per evitare confusione a questo punto è opportuno scambiare i due files nel progetto: ovvero includere il file verilog ed escludere lo schematico.

Project > Add/Remove Files in Project

Cliccando sul tasto con tre puntini andare a scegliere nel direttorio opportuno il file verilog testè generato

(Apri)

E Contestualmente rimuovere dall’elenco lo schematico (.bdf) – evidenziare il file e  
  
 (Remove)



(OK)

Nella Cartella dei files di progetto: Click destro sul file VerilogHDL appena importato e

Set As TOP Level Entity (Ctrl – Shift – V)

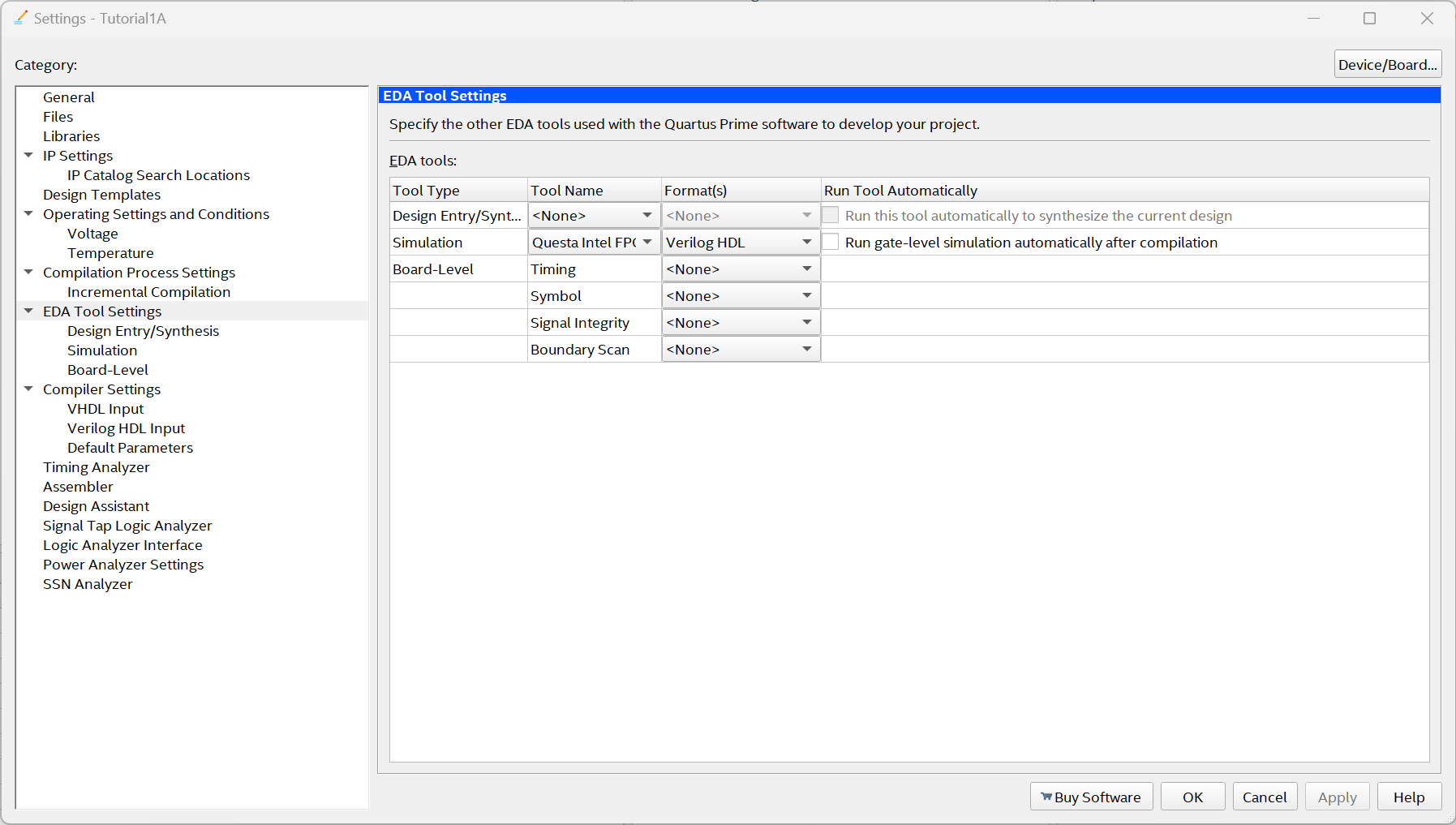
Rilanciare “Analisys and Elaboration”

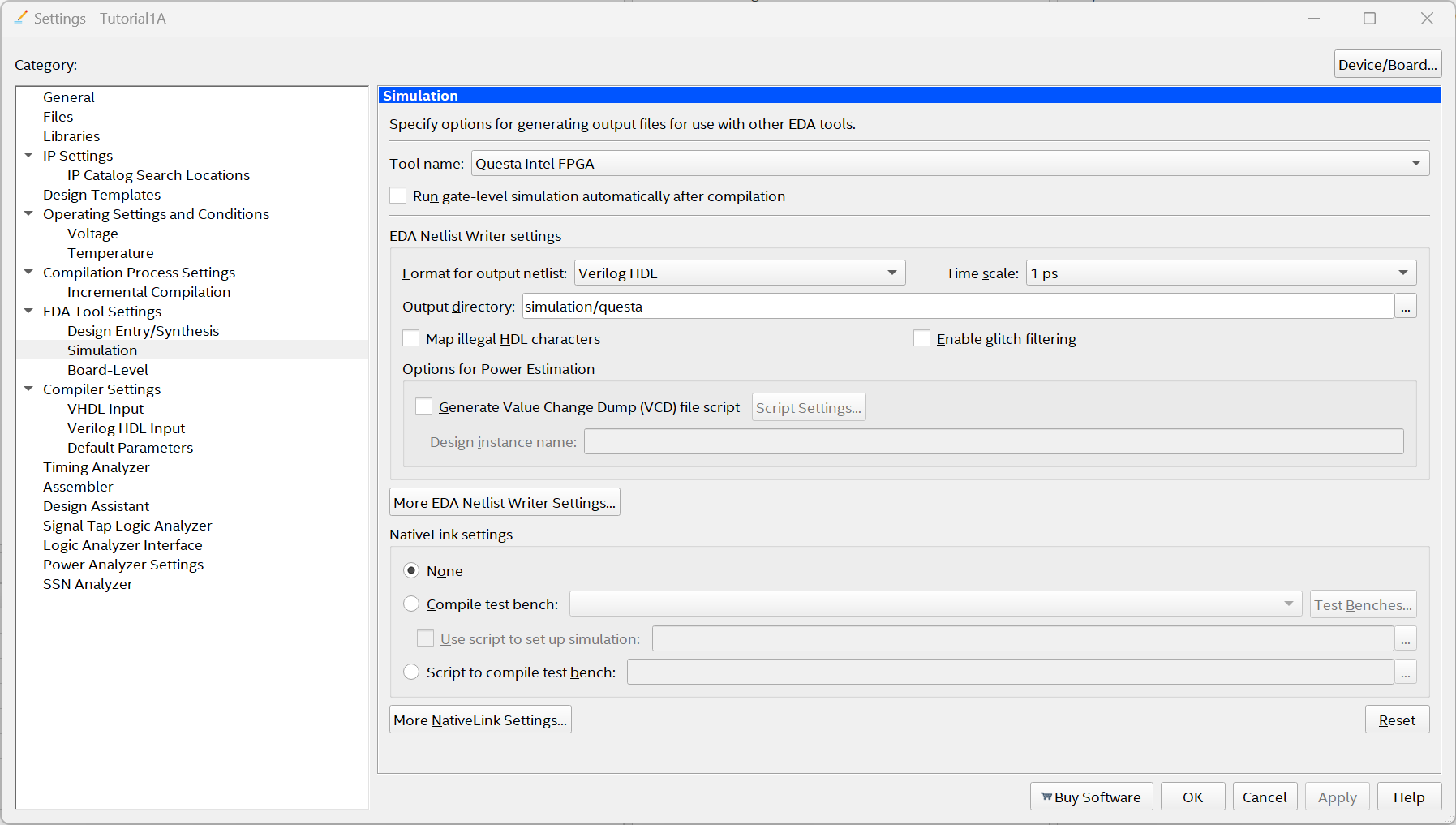
**Simulazione**

A questo punto si può far partire il simulatore esterno

Assicurarsi di aver definito il simulatore da utilizzare ed il suo settaggio accedendo ai vari sotto-menù

Assignment Setting > Settings (Ctrl – Shift – E )



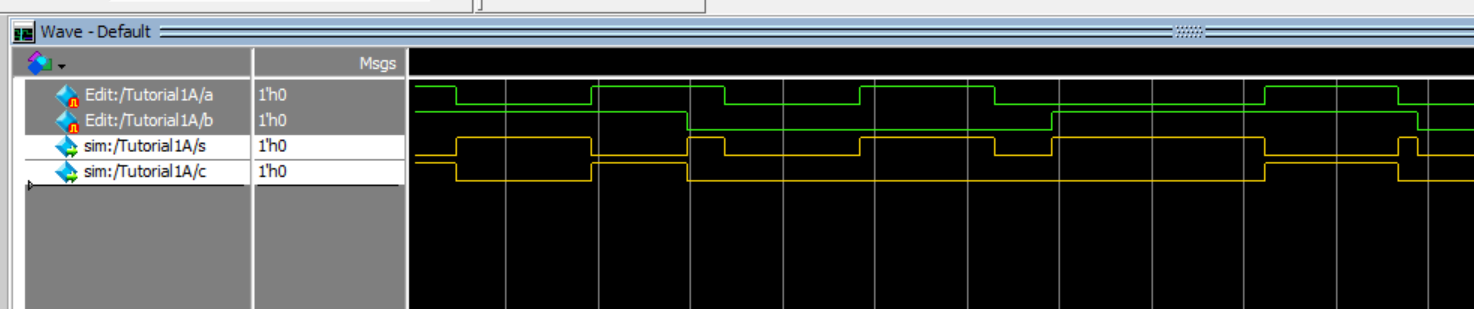


E lanciare il tool di Simulazione

Tools > Run Simulation Tool > RTL Simulation

Sommariamente:

1. Nella Cartella Library aprire la libreria “work”
2. All’nterno trovare il file da simulare
3. Click destro – Create Wave
4. Simulate > Start Simulation
   1. Scegliere nella libreria work il file da simulare (TOP Level Entity)
   2. Tra le opzioni, se necessario scegliere “Apply Full visibility …”
5. Nella finestra Wave includere i segnali da visualizzare (trascinandoli dalla cartella objects) e cancellare i segnali di stimoli NON configurati
6. Click destro sui segnali di stimoli   
   Edit > Wave Editor > Create / Modify Waveform
7. Definire i segnali desiderati
8. Per editare i segnali utilizzando il mouse   
   Wave > Mouse mode > Edit Mode
9. Modificare i segnali di stimoli a piacere
10. Lanciare la simulazione   
    Simulate > Run > Run -all
11. Eventualmente modificare le proprietà delle forme d’onda per rendere i risultati maggiormente leggibili



Se la simulazione fornisce risultati corretti si può procedere

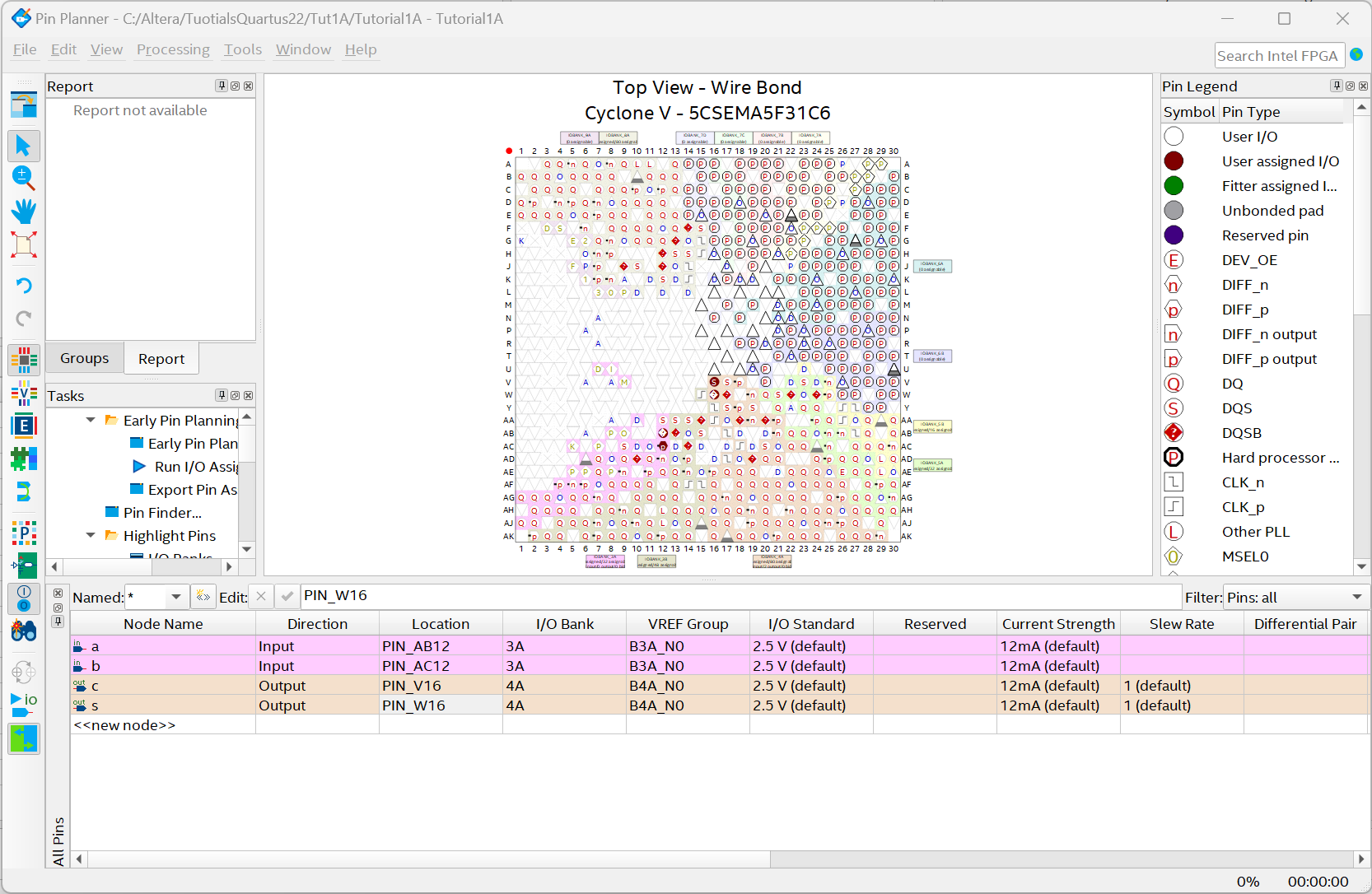
#### Definizione dei vincoli [8-9]

Al momento ci si limiterà a fissare dei semplici vincoli di posizionamento per i pin di ingresso e di uscita, onde fare in modo che gli ingressi siano associati rispettivamente agli switches 0 e 1 e che le uscite siano collegate al led rosso numero 0 e 1:

Dal manuale utente della DE1-SoC [] si evince che SW[0] è collegato al piedino **PIN\_AB12**, mentre SW[1] è collegato al piedino **PIN\_AC12**. Inoltre i LED sono collegati al **PIN\_V16** e **PIN\_W16**. Si tratta pertanto di associare queste posizioni ai piedini di ingresso e di uscita del nostro circuito (provare eventualmente altre associazioni, ma sempre col vincolo che ovviamente gli switches possono essere impiegati **esclusivamente** come ingressi. Sebbene sia la scheda che l’FPGA hanno delle protezioni, da un punto di vista generale, assegnare un valore logico ad un segnale che poi viene esternamente forzato ad un valore logico diverso crea un conflitto che potrebbe **danneggiare** il dispositivo stesso.

Assignments > Pin Planner (ctrl-shift-N)

Editare opportunamente la tabella come da figura:



Chiudere la finestra

Nota1: analogo risultato è ottenibile anche utilizzando il tool “assignment editor” che può essere impiegato anche per assegnare vincoli di altra natura..

Assignments > Assignments Editor (ctrl-Shift-A)

Nota2: Un altro modo più comodo per l’assegnazione dei vincoli di piedinatura (utile soprattutto quando il numero di piedini da assegnare è piuttosto elevato è quello di utilizzare “System Builder”: un tool messo a disposizione dal produttore della Board che in forma grafica ed in base ai dispositivi presenti sulla scheda che si vogliono adottare propone uno scheletro di un progetto da modificare a piacere, ma già corredato di tutti i vincoli di “locazione”. Questo tool verrà utilizzato nei prossimi tutorials.

A questo punto si può procedere con la compilazione completa del progetto. Questa prevede diversi passi “Analisi”, “Sintesi”, Place & Route”, “ Generate programming file”, “Time Quest Timing analysis”, ecc. ecc. che però vengono automaticamente svolti in sequenza:

Processing > Start Compilation (ctrl-L).

#### Simulazione temporale:

A questo punto il circuito è completamente realizzato e perfettamente definito in tutta la sua struttura interna all’FPGA, si potrebbe pertanto effettuare una ulteriore simulazione al più basso livello di astrazione che metta in luce di tutti i ritardi delle varie porzioni di circuito.

Si noti infatti che all’interno del direttorio dedicato ai files di simulazione adesso è presente un direttorio “gate work”

<project name>\simulation\questa\gate\_work

Pronto per la simulazione a livello di Gate, e nel direttorio superiore il file

<nomefile>.vo

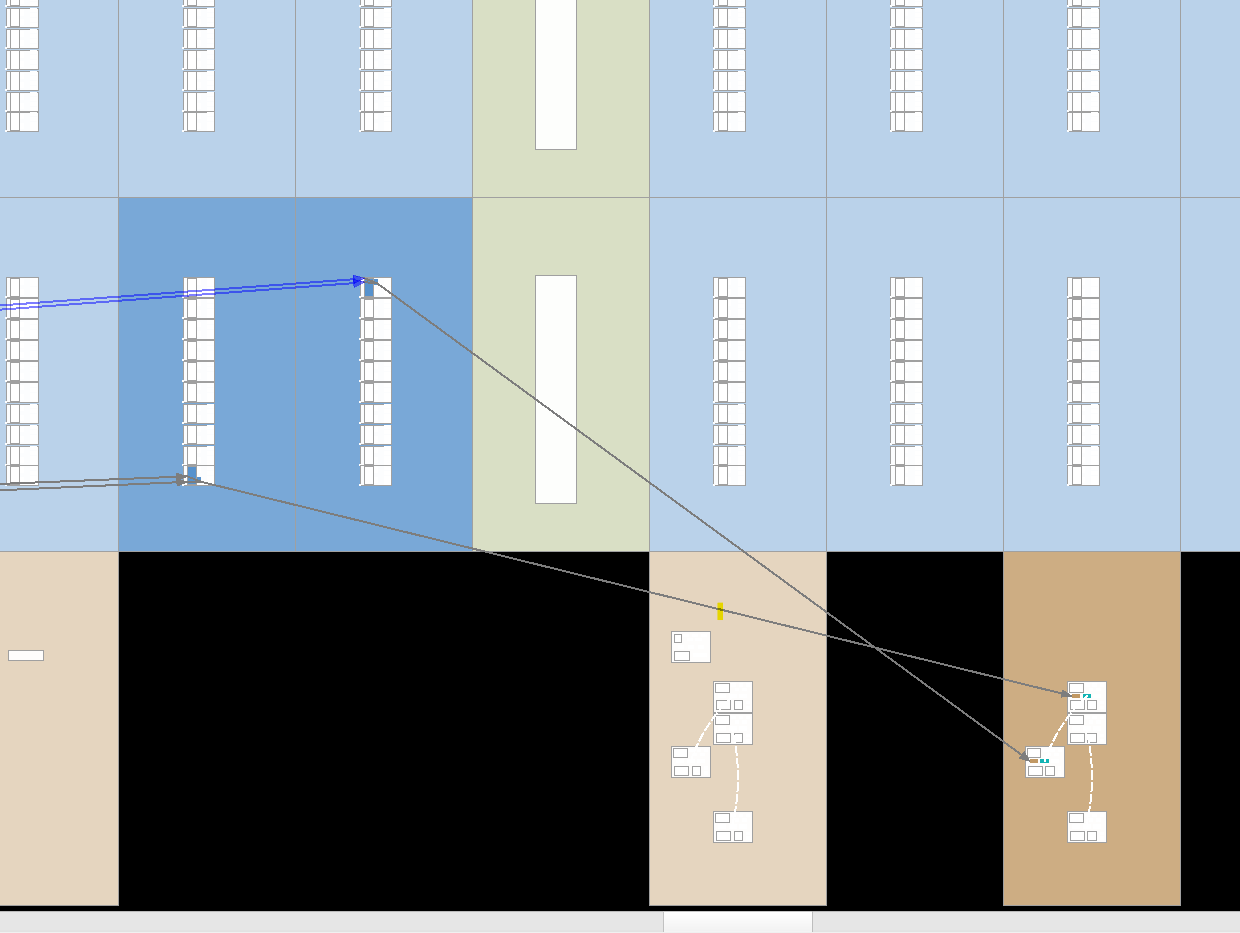
Che descrive a livello strutturale, tutti i blocchi funzionali e le porte di I/O che compongono il circuito finale a livello di Gate.

Purtroppo però per la Famiglia Cyclone V la casa produttrice **NON** ha fornito le librerie che definiscano le caratteristiche fisiche delle varie celle e dei singoli dispositivi che compongono l’FPGA ,fornendo invece come strumento per l’analisi delle tempistiche il “Timing Analyzer” che verrà analizzato in un tutorial successivo.

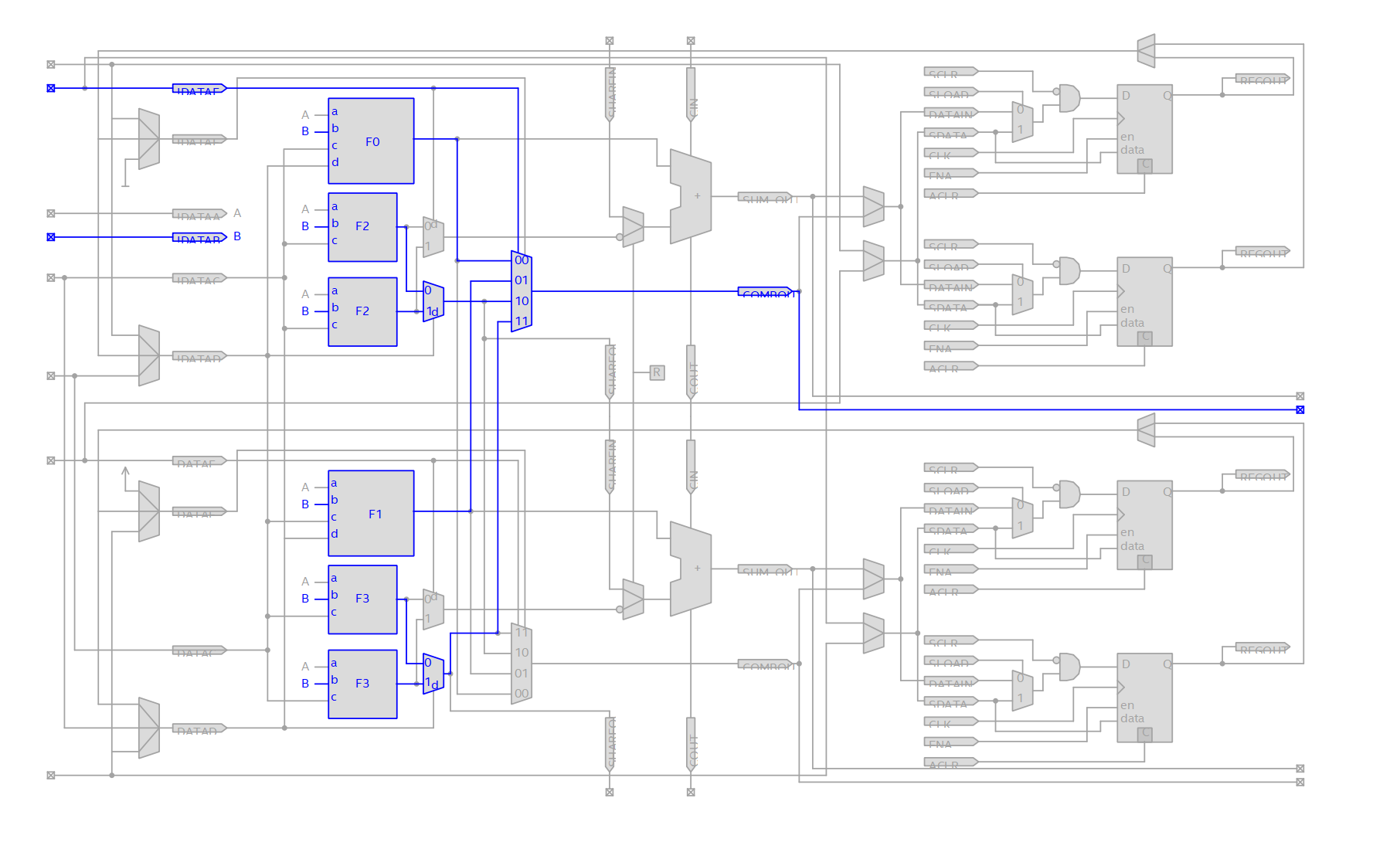
#### Visualizzazione del layout del circuito [10]:

Tools > Chip Planners

Utilizzare lo zoom per evidenziare piccole parti del circuito e tramite doppio click sulla singola cella evidenziarne la configurazione specifica.



Evidenziare in particolare i LE coinvolti dal progetto, i loro collegamenti, le porte di I/O interessate, i collegamenti e le loro configurazioni interne

  
Questo tool consente peraltro anche di modificare la posizione all’interno dell’FPGA dei vari LE impiegati ad esempio per riorganizzare i collegamenti e di conseguenza modificare i ritardi.  
Ma su circuiti di notevole complessità è bene delegare al sistema di sintesi automatica tale funzione.

#### Visualizzazione ed analisi dei vari report di processo:

Molto probabilmente la finestra contente tutti i reports di processo dovrebbe essere già aperta, altrimenti si può richiamare con

Processing > Compilation Reports (ctrl-R)

I vari reports sono organizzati per sezioni così come lo sono le varie fasi del processo di compilazione. Analizzarli nel dettaglio, in particolare soffermarsi sulle risorse impiegate all’interno dell’FPGA

Analysis & Settings > Resources Usage Summary

sulla congruenza della piedinatura

Fitter > Pin-out File

#### Analisi dettagliata dei ritardi

Per effettuare un’analisi dettagliata dei ritardi di propagazione dei vari percorsi del segnale si può utilizzare uno strumento dedicato a tale scopo, denominato “Timing Analyzer” []

Tools > Timing Analyzer

Nella nuova finestra si creino le informazioni relative alle temporizzazioni

Netlist > Create Timing Netlist

E si scelga la modalità di analisi (**Post-fit**) ed il modello da adottare (**Slow**).

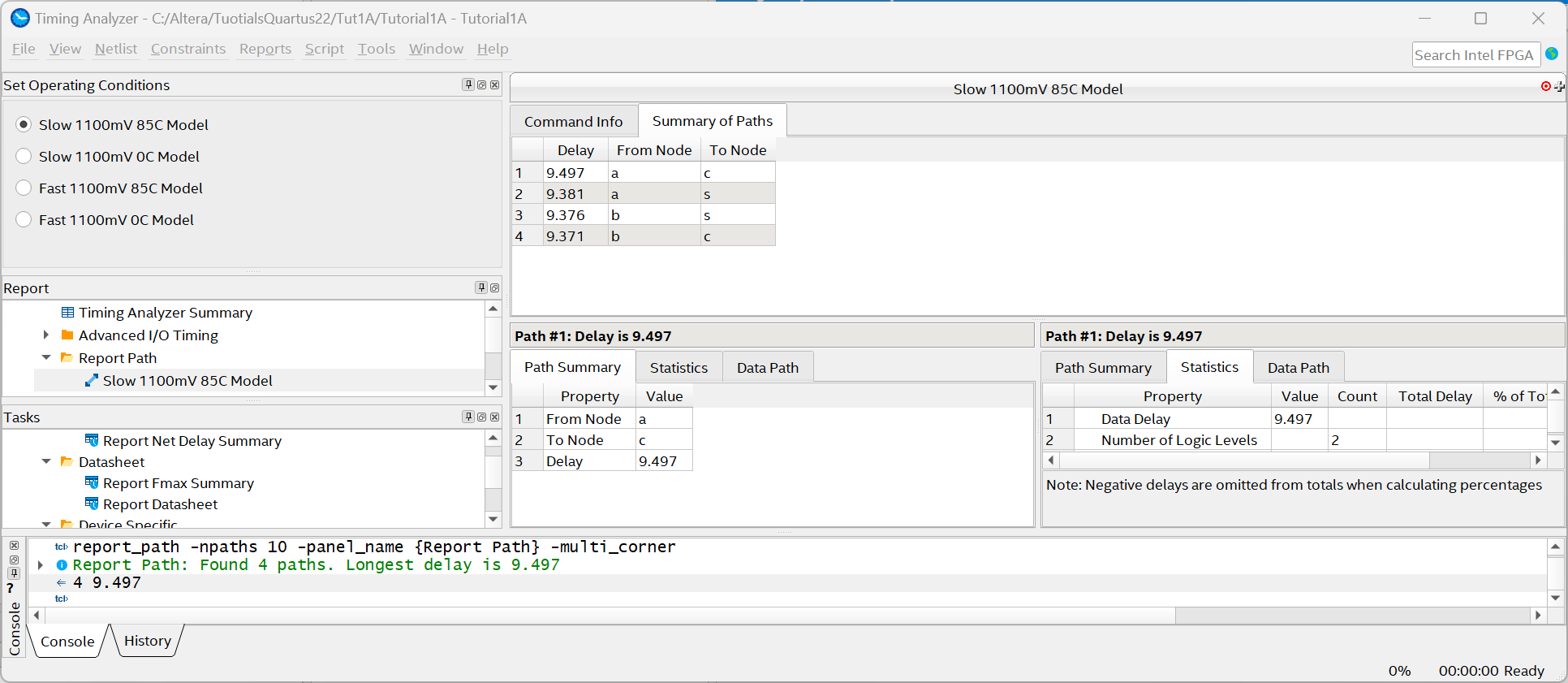
A questo punto bisognerebbe fissare il clock del sistema, la sua frequenza di lavoro ed i vincoli entro i quali il sistema dovrebbe sottostare, ma data la semplicità del progetto in questione si può passare alla fase successiva:

Netlist > Update Timing Netlist

Ora si visualizzi il report di interesse

Reports > Custom Reports > Reports Path

La finestra che si apre ha funzione di filtro onde visualizzare, all’interno di un progetto complesso solo i paths di nostro interesse, ma nel caso in esame si possono visualizzare tutti i paths essendo essi solamente quattro. Si porti pertanto la variabile “**Reports Number of Paths**” ad un valore superiore a 4 e si clicchi su “**Report Paths**”

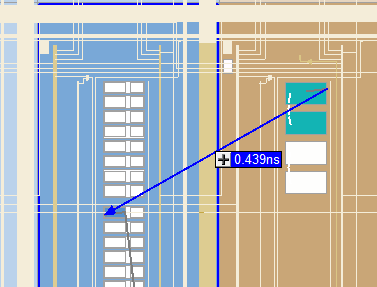


Nella finestra che così si apre si avrà un report dettagliato in tutti i passaggi dei quattro paths che compongono il nostro circuito dall’ingresso del segnale fino alle due uscite di quest’ultimo. Inoltre cliccando col tasto destro sul path di nostro interesse e scegliendo “**locate path**” si può individuare il path in oggetto in alcune delle possibili visualizzazioni del circuito.

Ad esempio visualizzandoli all’interno del “Chip Planner”, e sfruttando il comando interno

View > Show Delays

Si possono visualizzare il loco i ritardi dei vari step del percorso.



Si noti inoltre che lo strumento “Timing Analyzer” non ha valenza solo di controllo, ma può essere impiegato per fissare dei vincoli da passare al sintetizzatore, in maniera che in fase di realizzazione del circuito possa verificare se detti vincoli sono rispettati o, in caso contrario provare altre soluzioni.

#### Test su Scheda

Giunti a questo punto non rimane che scaricare il bitstream (che contiene tutte le informazioni sulla configurazione dell’FPGA) sulla FPGA medesima

* Verificare che i micro-switches MSEL sul retro della scheda siano in configurazione   
  MSEL [1:6]= 6’b01001-
* Collegare il cavo di download USB
* Accendere la scheda

Dopo un istante la scheda si configurerà in base al file di configurazione residente nella flash che di default prevede la realizzazione di pattern grafici sui led.

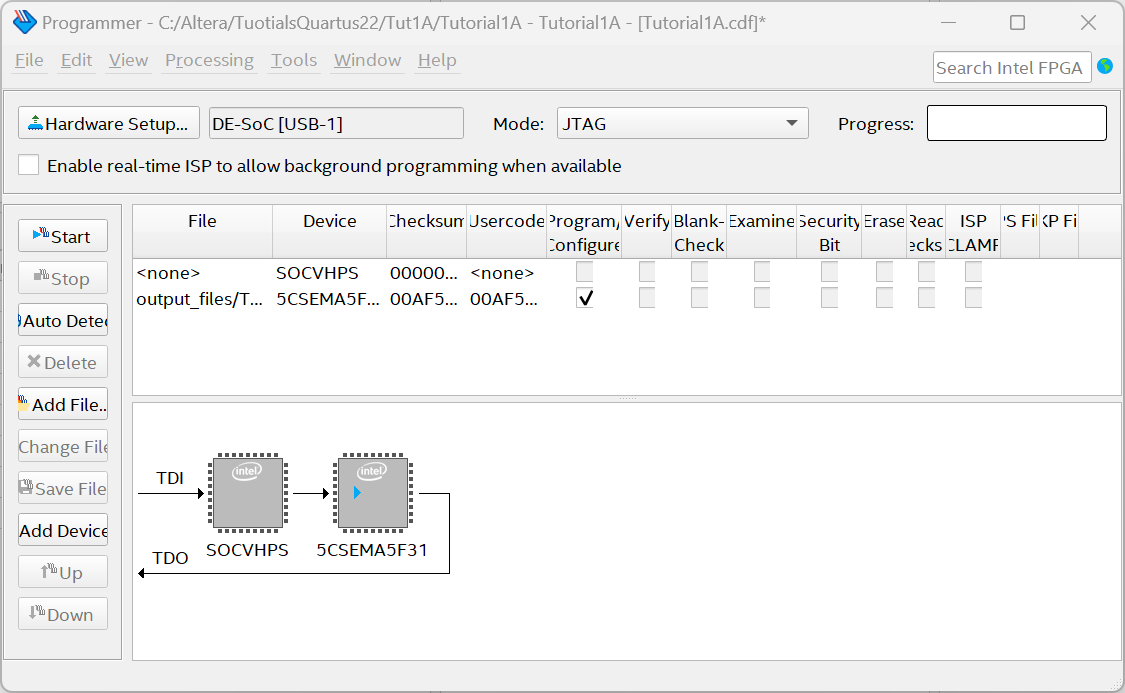
Tools > Programmer

Eventualmente - Click on “Hardware setup” e scegliere DE-SoC

Eventualmente – Auto Detect e scegliere 5CSEMA5

Add File – nella cartella “output\_files” scegliere “<nome>.sof

La catena di programmazione deve essere costituita da due elementi in serie (SOCVHPS e 5SEMA5F31) ol file di programmazione. Se ve ne fossero altri si possono rimuovere



Click on Start

Verificare il corretto funzionamento del sistema

## Bibliografia

[1] Terasic : “**DE1-SoC User Manual – Terasic**” *https://www.terasic.com.tw/cgi-bin/page/archive\_download.pl?Language=China&No=836&FID=ae336c1d5103cac046279ed1568a8bc3*

[2] Altera Corporation: **“Cyclone II FPGA Starter Development Board -Reference Manual**” [*http://www.altera.com/literature/manual/mnl\_cii\_starter\_board\_rm.pdf*](http://www.altera.com/literature/manual/mnl_cii_starter_board_rm.pdf)

[3] Altera Corporation: **“Quartus II Handbook Version 13.1 - Volume 1: Design and Synthesis**”

[*http://www.altera.com/literature/hb/qts/quartusii\_handbook.pdf*](http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf)

[4] Altera Corporation: “**Cyclone II Device Handbook**”   
<http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf>

[5] Altera Corporation: “**Documentation on Cyclone II Devices**”  
<http://www.altera.com/literature/lit-cyc2.jsp>

[5] Altera Corporation: “**My First FPGA Design Tutorial**” [*http://www.altera.com/literature/tt/tt\_my\_first\_fpga.pdf*](http://www.altera.com/literature/tt/tt_my_first_fpga.pdf)

[6] Altera Corporation: “**Quartus II Introduction Using Schematic Designs**” [*ftp://ftp.altera.com/up/pub/Altera\_Material/13.0/Tutorials/Schematic/Quartus\_II\_Introduction.pdf*](ftp://ftp.altera.com/up/pub/Altera_Material/13.0/Tutorials/Schematic/Quartus_II_Introduction.pdf)

[7] Altera Corporation: “**Quartus II Handbook Version 13.1-Volume 3: Verification**”  
<http://www.altera.com/literature/hb/qts/qts_qii5v3.pdf>

[8] Altera Corporation: “**Using the Assignment Editor in the Quartus II Software**”  
<http://www.altera.co.jp/literature/wp/wp_asigned.pdf>

[9] Altera Corporation: “**Constraining Designs**”  
<http://www.altera.com/literature/hb/qts/qts_qii52001.pdf>

[10] Altera Corporation: “**Analyzing and Optimizing the Desig Floorplan with the Chip Planner**”[*http://www.altera.com/literature/hb/qts/qts\_qii52006.pdf*](http://www.altera.com/literature/hb/qts/qts_qii52006.pdf)

[11] Altera Corporation: “**TimeQuest Timing Analyzer - Quick Start Tutorial**”  
[*http://www.altera.com/literature/ug/ug\_tq\_tutorial.pdf*](http://www.altera.com/literature/ug/ug_tq_tutorial.pdf)

[12] Altera Corporation: “**Quartus II TimeQuest Timing Analyzer Cookbook**”   
[*http://www.altera.com/literature/manual/mnl\_timequest\_cookbook.pdf*](http://www.altera.com/literature/manual/mnl_timequest_cookbook.pdf)

[13] Altera Corporation: “**The Quartus II TimeQuest Timing Analyzer**”   
[*http://www.altera.com/literature/hb/qts/qts\_qii53018.pdf*](http://www.altera.com/literature/hb/qts/qts_qii53018.pdf)[14] Altera Corporation: “**Quartus II Introduction Using Verilog Design**” [*ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital\_Logic/tut\_quartus\_intro\_verilog.pdf*](ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_quartus_intro_verilog.pdf)

[15] Stuart Sutherland: “**Verilog® HDL - Quick Reference Guide**”   
[*http://www.sutherland-hdl.com/online\_verilog\_ref\_guide/verilog\_2001\_ref\_guide.pdf*](http://www.sutherland-hdl.com/online_verilog_ref_guide/verilog_2001_ref_guide.pdf)

[16] Synopsys Corporation: “**FPGA Compiler II / FPGA Express Verilog HDL Reference Manual**”  
[*http://classes.soe.ucsc.edu/cmpe225/Fall01/synver.pdf*](http://classes.soe.ucsc.edu/cmpe225/Fall01/synver.pdf)

[16] “**IL LINGUAGGIO HDL VERILOG PER LA SINTESI E LA. SIMULAZIONE DI CIRCUITI DIGITALI**”  
[*http://unina.stidue.net/Sistemi%20Elettronici%20Programmabili/Materiale/Manuale%20Verilog.pdf*](http://unina.stidue.net/Sistemi%20Elettronici%20Programmabili/Materiale/Manuale%20Verilog.pdf)

[17] Carlo Brandolese - Politecnico di Milano: “**Introduzione al linguaggio VHDL - Aspetti teorici ed esempi di progettazione”,** [*http://www.fabiopanozzo.it/download/Reti\_logiche/VHDL%20(Dispensa%20a%20cura%20di%20Carlo%20Brandolese).pdf*](http://www.fabiopanozzo.it/download/Reti_logiche/VHDL%20(Dispensa%20a%20cura%20di%20Carlo%20Brandolese).pdf)

**[] FPGAcademy** [**https://fpgacademy.org/tools.html**](https://fpgacademy.org/tools.html)