BREVE GUIDA ALL' UTILIZZO DI QUARTUS II PER LO SVILUPPO DI PROGETTI VHDL
a.a 2010-2011

Introduzione

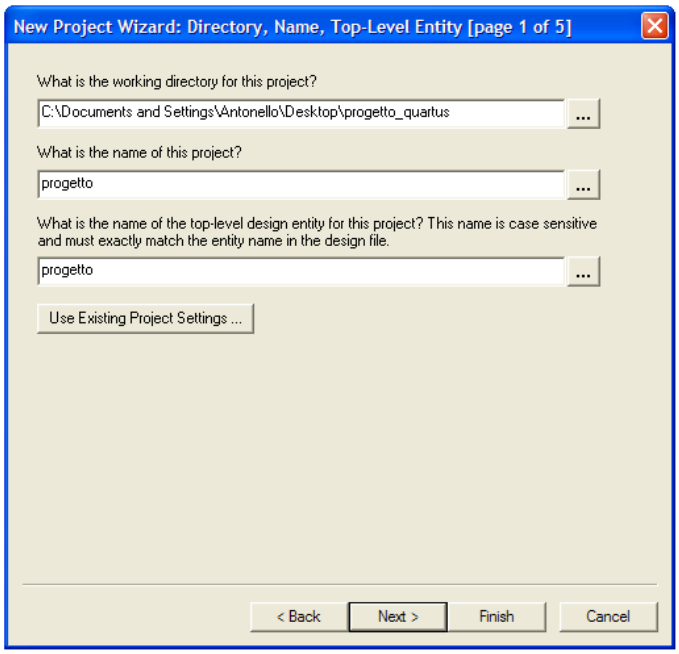
QUARTUS è un tool utilizzabile per effettuare, nell'ambito della progettazione di circuiti digitali:

1. Descrizione a livello RTL del circuito
2. Simulazione funzionale
3. Sintesi logica su dispositivi FPGA della famiglia Altera
4. Simulazione post-sintesi ("timing") e analisi delle prestazioni

In questa guida verranno illustrati e commentati i vari passi che sarà necessario compiere per il progetto di circuiti digitali tramite linguaggio VHDL. QUARTUS è composto da diversi tools (Compiler, Simulator, Text Editor, etc.) ognuno dei quali serve per una fase specifica del flusso di progetto.

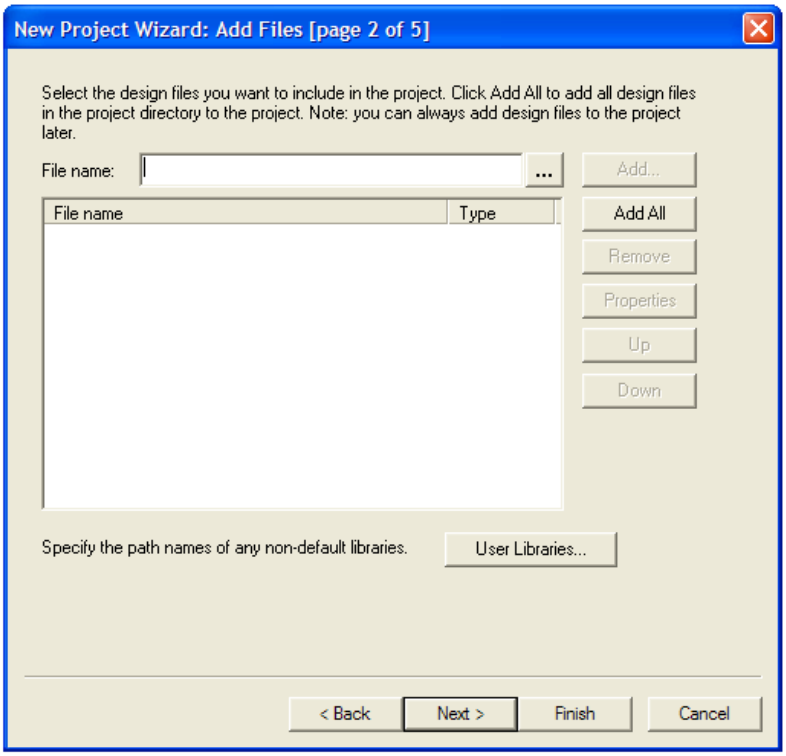
CREAZIONE DI UN PROGETTO

Il primo passo da svolgere consiste nella definizione un nuovo progetto. Con Progetto si intende un insieme di files (es. file VHDL .vhd, file con forme d'onda per la simulazione .vwf, file contenenti i reports forniti dai differenti tool) che vengono raggruppati in un direttorio comune di lavoro che deve essere specificato dall'utente all'atto della creazione del progetto stesso (imp: mai direttamente nel desktop).



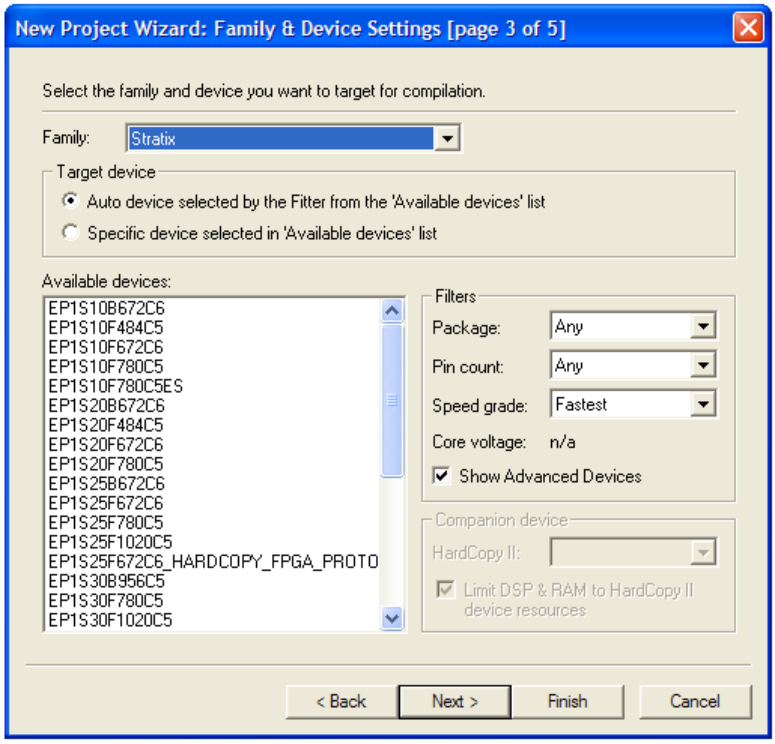
Per poter generare un nuovo progetto selezionare il menù di Quartus: ***File>NewProjectWizard***. Nella prima schermata che appare occorre specificare un direttorio di lavoro (working directory), il nome da assegnare al progetto ed il nome della Top-level entity del progetto stesso. Si noti che quest'ultima voce può successivamente essere modificata per consentire al progettista di analizzare differenti parti del proprio design senza necessariamente dover lavorare sull'intero progetto. Di default il sistema attribuisce al progetto e alla entità a top-level (e quindi al circuito da simulare) il nome del direttorio di lavoro.

Il passo successivo consente di associare al progetto alcuni file (sia .vhd che .vwf). Poiché questo manuale vuole essere una guida per la creazione di un nuovo progetto ignoriamo tale passo ipotizzando di non avere alcun file da inserire. Si noti comunque che è possibile aggiungere nuovi file al proprio progetto in un secondo momento specificando quali sono i file in uso dal menù *Assignment-> Settings*.



Il passo 3 è dedicato alla famiglia logica sulla quale si vuole mappare il circuito che si vuole progettare. E' evidente che per selezionare il dispositivo più adatto occorre innanzitutto avere una conoscenza delle famiglie logiche messe a disposizione da ALTERA ed una stima di occupazione d'area in termini di GATE utilizzati e del numero di PIN necessari a realizzare il proprio modulo circuitale. Oltre ciò bisogna ovviamente tenere in considerazione le prestazioni desiderate per il circuito progettato.

Poiché in questo momento non abbiamo le specifiche che ci consentirebbero di fare una scelta oculata del dispositivo lasciamo tutte le impostazioni di default e terminiamo il processo di creazione del nuovo progetto (premendo il tasto *Finish*) ignorando i passi 4 e 5.



Si potrà accedere al Project appena creato in altre sessioni di lavoro con Quartus, specificando il nome del progetto con *File> Open Project.*

1. SCRITTURA e VERIFICA DEL CODICE VHDL

Per creare un nuovo file, dal menù ***File>New*** > *Design File* selezionare la voce *VDHL File*. Prima di iniziare ad editare, salvare il file appena creato assegnandogli un nome pertinente. Selezionando la voce del menù *File>Save as* specifichiamo il nome voluto per il nostro file (*adder4.vhd*). Si noti che l'estensione viene automaticamente assegnata e che il file viene aggiunto al progetto lasciando selezionata la voce *Add file to current project* nel menù di salvataggio del file. Si potrà accedere a file precedentemente scritti specificandone il nome dal menù *File> open*.

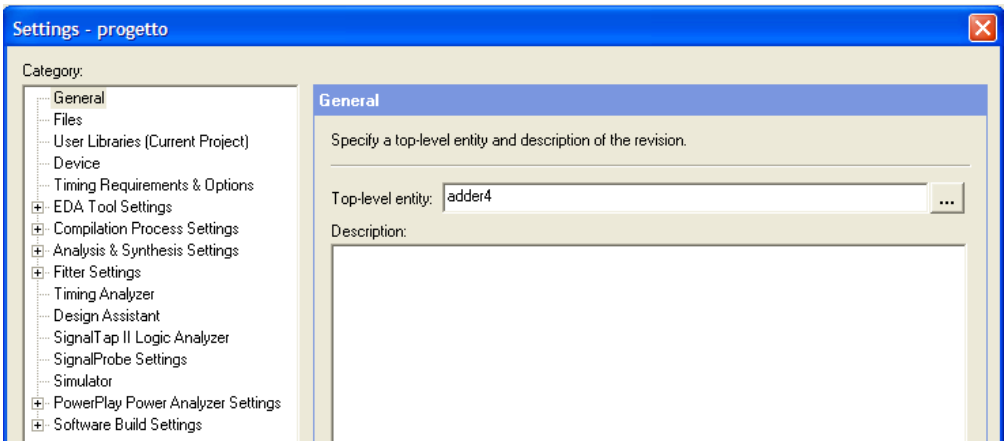
Completata la procedura di salvataggio il file viene aggiunto al progetto ed è possibile vederlo nella finestra di navigazione di progetto (Project Navigator, collocata solitamente alla sinistra dell'area di lavoro) sotto la voce *Files*. Qualora questa finestra non fosse visibile nell'area di lavoro la si può richiamare dal menù *View>Utility Windows>Project Navigator*.

Verifica del codice

Prima di effettuare una simulazione funzionale, selezioniamo nel menù la seguente voce: ***Processing>Start>Start Analysis&Elaboration.*** Questo passo verifica solo la correttezza della sintassi del codice VHDL.

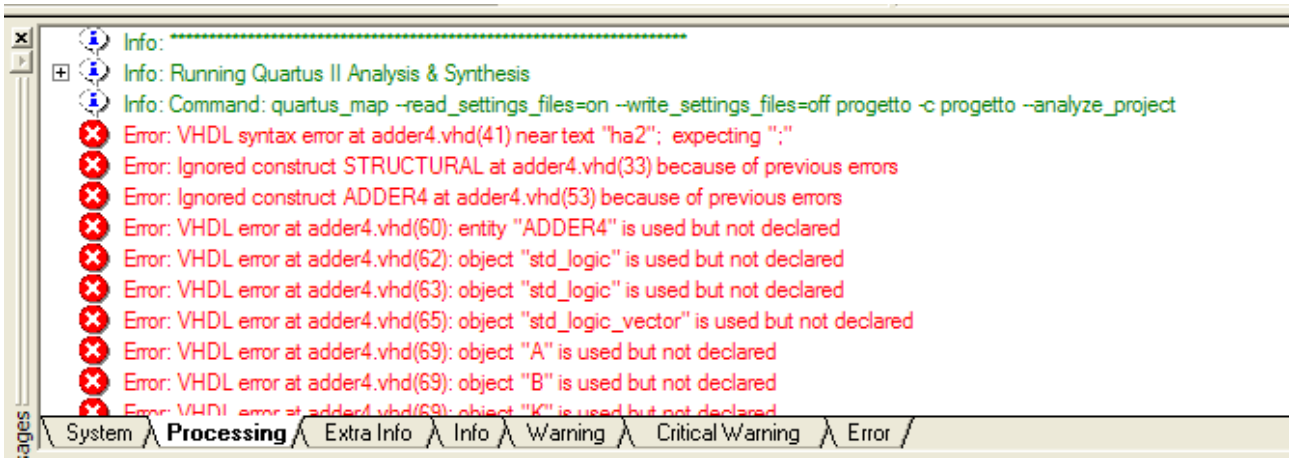
Verificare che la Top-level entity sia correttamente specificata. Per verificare ciò accedere al menù Assignments> Settings. Nella categoria *General* specifichiamo alla voce *Top-level Entity* il nome dell'entità che vogliamo compilare. Nel caso in questione la nostra entità a

livello più alto della gerarchia sarà inizialmente HA, poi FA e infine **adder4**. La gerarchia di progetto è visibile dalla finestra Project Navigator (Hierachy).



Una volta effettuata la sintesi con successo appare il seguente messaggio:

Qualora invece la sintesi fallisca a causa di qualche errore o comunque si presentino un numero di warning non nullo è necessario analizzare il risultato della compilazione leggendo i messaggi di errore/warning nella finestra in basso dei messaggi.



E' possibile, facendo doppio clic sulla linea corrispondente all'errore farsi portare (nel Text Editor) alla riga corrispondente all'errore.

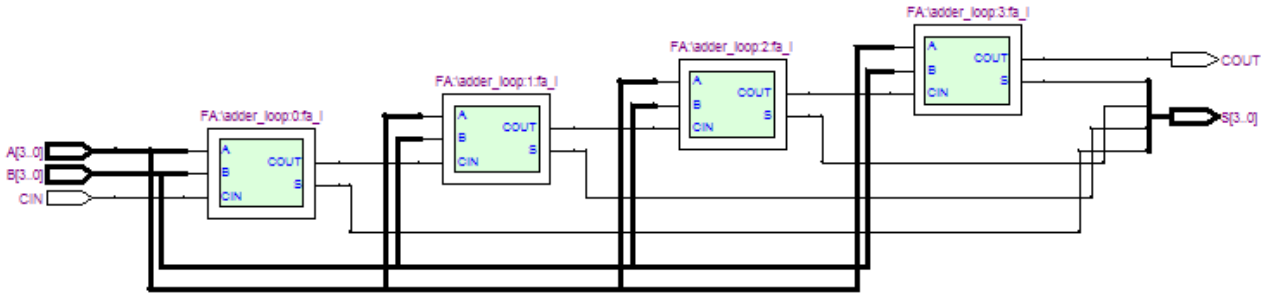
Attenzione! Poiché alcuni errori ne generano altri (si può avere il tipico effetto “a valanga”) è sempre bene risolvere gli errori cominciando dal primo (numero di riga di codice più basso).

Attenzione! Questo passo verifica solo la correttezza della sintassi del codice VHDL. Vengono però generati dei messaggi (warnings) che aiutano a segnalare la presenza di errori logici prima della simulazione: per esempio il messaggio che indica che alcune uscite sono stuck-at 0 o 1 (cioè fisse al valore logico alto o basso) o quello che segnala che alcuni ingressi non sono connessi a nessun nodo del circuito.

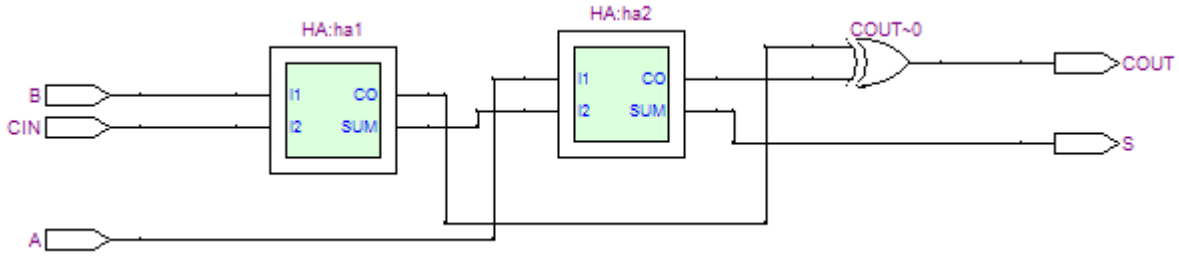
Visualizzazione dello schematico dalla descrizione RTL

Al fine di verificare che il codice rispecchi effettivamente il modulo circuitale progettato, è possibile utilizzare la vista RTL, che altro non è che la rappresentazione tramite schematico gerarchico del circuito descritto a livello *Register-Transfer-Level*.

Per richiamare tale funzione occorre dal menù di QUARTUS lanciare il seguente comando: ***Tools>netlist viewers > RTL Viewer***. La figura sottostante rappresenta la vista RTL del nostro sommatore, ove ogni blocco celeste altro rappresenta un FULL_ADDER con riporto.



E’ possibile inoltre esplorare la gerarchia e, per esempio, come ogni FA è internamente implementato. A tale scopo basta cliccare col tasto sinistro del mouse due volte su uno dei quattro FA per ottenere il seguente risultato,che mostra come ogni FA si compone di due HALF_ADDER ed un OR:



Si noti che questo strumento non è sempre di facile utilizzo soprattutto in progetti di complessità maggiore. Ad ogni modo la sua utilità è indubbia soprattutto per scongiurare i più comuni errori che portano alla generazione di Latch parassiti nel circuito.

2 . SIMULAZIONE FUNZIONALE

Creazione della netlist

La fase di sintesi logica può essere di due tipi:

1. **Functional.** Questo tipo di sintesi logica, genera una netlist "ideale", cioè priva di ritardi. Non tiene conto dei ritardi di propagazione nei blocchi logici, né nelle interconnessioni. E' più rapida della sintesi Timing, quindi è necessario utilizzare questo tipo di sintesi per verificare la correttezza funzionale della rete (cioè: la rete fa esattamente quello che era richiesto?).

   Per indicare che si intende eseguire una sintesi e quindi una simulazione di tipo funzionale, è necessario selezionare dal menù ***Processing>Generate Functional Simulation Netlist***

2. **Timing.** Questo tipo di sintesi genera una netlist "reale", cioè composta da blocchi logici e interconnessioni caratterizzati da ritardi di propagazione definiti. E' computazionalmente più pesante della sintesi Functional, quindi varrà la pena utilizzarla solo quando saremo sicuri che il circuito ha il comportamento "funzionale" desiderato, e dovremo caratterizzare le prestazioni (frequenza, area, potenza, etc.) della rete digitale. In QUARTUS questo tipo di sintesi logica comprende anche la fase di Place and Route su dispositivo FPGA e la determinazione dei ritardi. Esamineremo questi passi nella sezione 3.

Creazione del file con gli stimoli in ingresso

Effettuata la sintesi funzionale, occorre verificare se il circuito funziona correttamente. Gli strumenti da utilizzare sono il Waveform Editor e il Simulator. Prima di tutto occorre creare un file di forme d'onda per imporre gli stimoli al circuito e verificare che si comporti come desiderato. Per prima cosa creare un nuovo file di tipo "Vector Waveform File" .vwf dal menù ***File>New>verification/debugging***:

Prima di editare, salvari il file appena creato assegnandogli un nome pertinente. Selezionando la voce del menù *File>Save as* specifichiamo il nome voluto per il nostro file (*adder4.vwf*).

Vediamo ora come procedere.

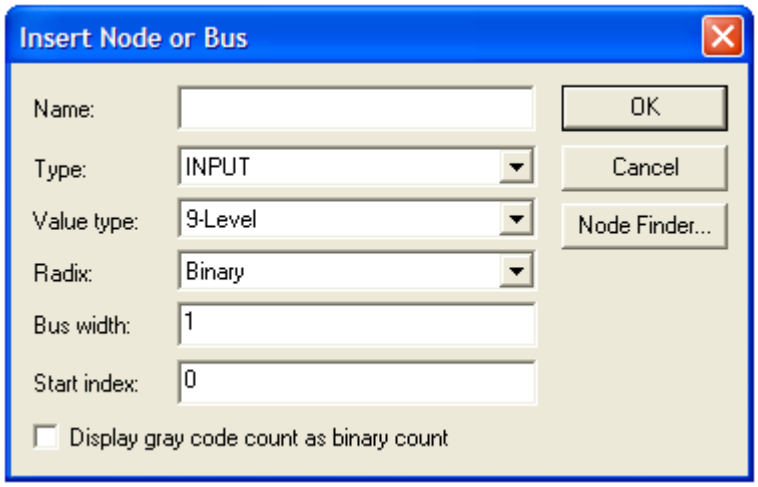
1. Innanzitutto occorre decidere la "risoluzione" (grid size) delle forme d'onda, cioè definire il passo di discretizzazione in cui verranno calcolati i valori dei segnali: nelle reti digitali questo valore deve essere pari almeno semiperiodo di clock, poiché il clock commuta proprio ogni mezzo periodo. E' bene scegliere una risoluzione adeguata alla velocità della rete: se la rete fosse sincrona, e andasse a 1MHZ (Tck = 1 usec), sarebbe inutile specificare una risoluzione di 1ns!!!

   L'ADDER4 che stiamo progettando è una rete combinatoria e non ha quindi un segnale di clock. Per impostare la "risoluzione" selezionare dal menù la voce: *Edit>Grid size* e impostare per esempio `100ns` (il valore di default è 10 ns). IMP: negli esercizi che svolgeremo converrà sempre assumere grid size pari a metà del periodo di clock .

2. Ora dobbiamo decidere per quanto tempo portare avanti la simulazione (cioè l'istante finale). Facciamo i conti con la griglia che abbiamo scelto, sapendo che il numero di passi che il simulatore dovrà fare sarà $T_{finale}/T_{gridsize.}$

Selezioniamo dal menù la seguente voce: *Edit>End time* ed impostiamo per esempio `2us` (corrispondente a 2us/100ns = 20 punti). Chiaramente più punti metteremo, più tempo impiegherà il simulatore a terminare la simulazione. (I due parametri per controllare la simulazione possono anche essere assegnati dal menù *Assignments> Setting > Simulator*)

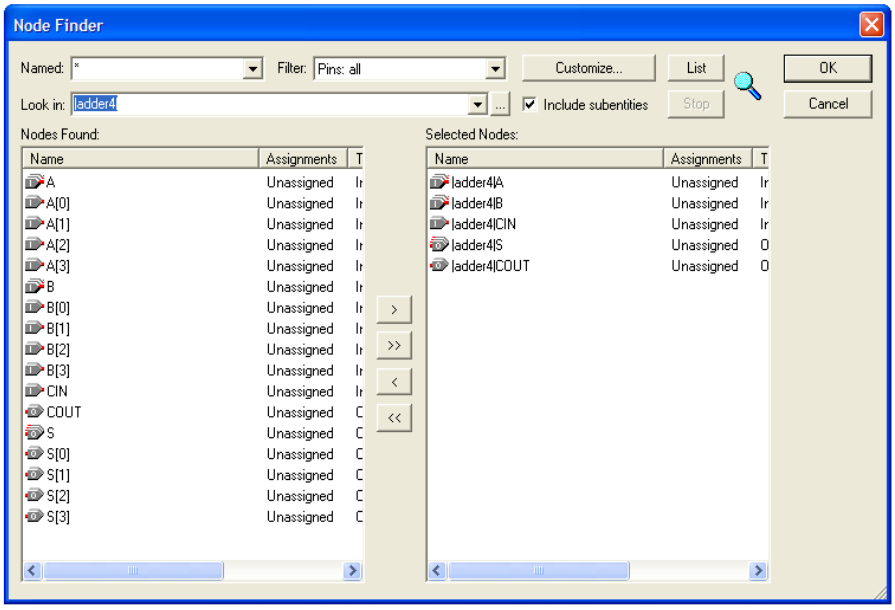
3. Fatto questo andiamo a disegnare le forme d'onda. Aggiungiamo quindi tutti i segnali di ingresso e uscita che vogliamo analizzare. Cliccare col tasto destro in un punto della colonna *Name* e selezioniamo *Insert>Insert Nodes or Bus*



Nella casella "Type" è possibile restringere la ricerca dei segnali a particolari categorie:

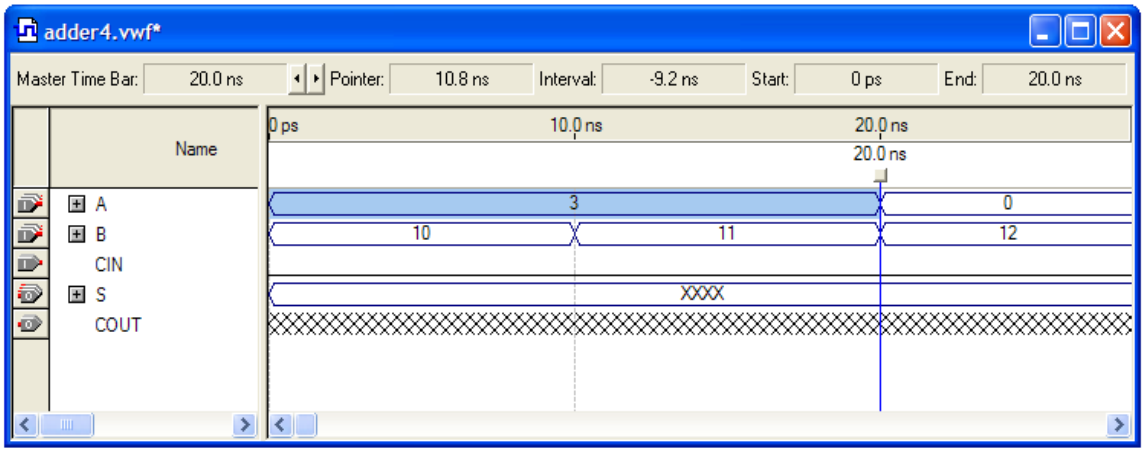
- Inputs: gli ingressi
- Outputs: le uscite
- Reg: uscite di registri
- Combinatorial: uscite di blocchi combinatori

Per agevolare la ricerca è utilizzare il *Node Finder* cliccando il relativo tasto presente nella finestra. La seguente finestra consente di fare una ricerca più dettagliata dei nodi che vogliamo analizzare. Il campo *Filter* permettere di controllare i criteri della ricerca. Selezionare *pins: all e* procedere alla ricerca dei nodi cliccando il tasto *List.* La lista di nodi compatibile con i criteri di ricerca prescelti comparirà alla voce *Node Found.* Si noti che compariranno gli ingressi A, B, S sia come singoli bit, che come "stringa di N bit". Selezioniamo (agendo con il tasto >) la stringa per motivi di compattezza, ed aggiungiamo i riporti in ingresso e uscita CIN, COUT.



IMP: per i circuiti sequenziali è molto utile visualizzare l'uscita dei registri interni: per fare ciò (solo dopo avere eseguito il comando di generazione della netlist) dal node finder selezionare come filtro: **registers pre-synthesis**

Una volta selezionati i nodi di interesse, premere *OK*.



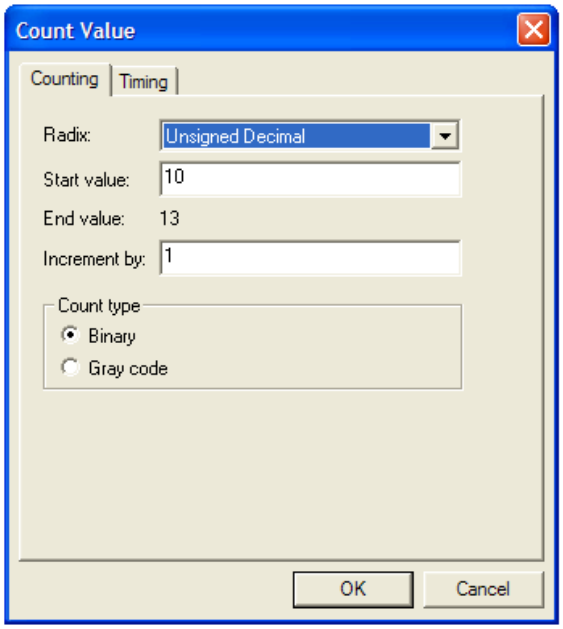
Le uscite COUT e S sono ovviamente indefinite, mentre gli ingressi sono posti a '0' a default. Andiamo a cambiarli! Vogliamo verificare che il sommatore funzioni, quindi andremo a definire dei valori ad A, B e CIN.

Ora selezioniamo una parte della forma d'onda di CIN che per esempio vorremo porre uguale a '1'. La cosa si fa cliccando col tasto sinistro del mouse in un punto della forma d'onda e spostandosi a destra tenendo premuto (come per selezionare il testo in un word-processor).

Per default i segnali sono rappresentati in base binaria. Impostiamo ora A, B, S in decimale. Per esempio selezioniamo la forma d'onda di A tenendo premuto il tasto destro del mouse;

visualizzeremo una finestra da cui selezionare “properties”. Cambiamo il campo *binary* a *Unsigned Decimal* . In modo analogo, si procede con il segnale d’uscita COUT.

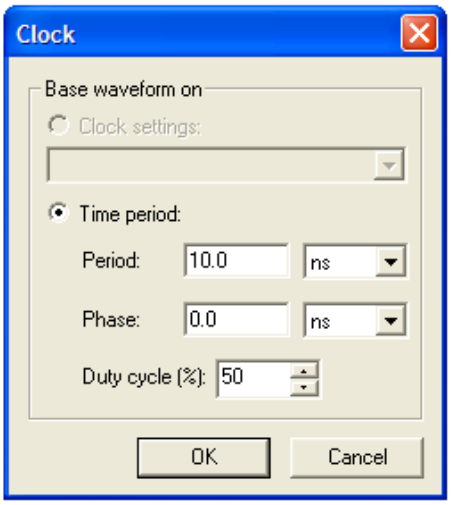
Ora diamo un valore ad B. Selezioniamo il segnale con il tasto destro del mouse. Assegniamo ora un COUNT VALUE ad B, selezionando la voce *Value/Count Value* (un Count Value è un valore che si incrementa con cadenza regolare indicata dall’utente).



I parametri sono:

- Il valore iniziale di conteggio *Start Value*, mettiamolo per esempio a 10.
- Il passo di incremento *Increment By*, mettiamolo a 1.
- L’intervallo dell’incremento che va impostato nel sotto-menù *Timing*. Mettiamolo a 10ns. La rete è infatti combinatoria e l'unico vincolo che occorre rispettare è che le configurazioni degli ingressi si mantengano per un tempo sufficiente ad esaurire i transistori. In un circuito sequenziale sincrono il valore in questo campo dovrà essere multiplo del periodo di clock.

Nota: Qualora dovessimo inizializzare un segnale di CLOCK, si seleziona la voce *Value/Clock* e, nel menù che compare, si specifica il periodo e il duty-cycle.

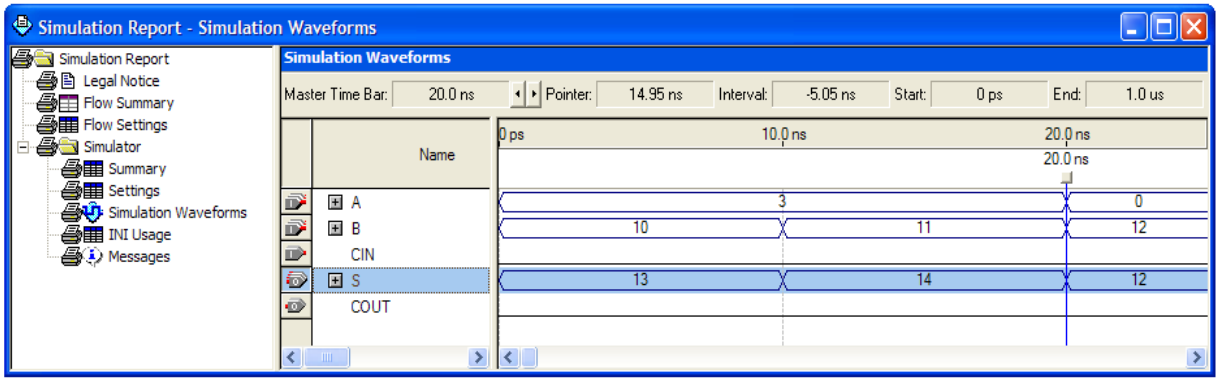


Specificare che si vuole eseguire una simulazione funzionale

Specificare da *Assignments>Settings/SimulatorSettings /**Simulator mode Functional,*** mentre alla voce */Simulator input* indicare il file appena creato *adder4.vwf* .

Simulazione

Fatto ciò lanciamo la simulazione dal menù di QUARTUS ***Processing>Start Simulation***. Le forme d'onda sottostanti rappresentano il risultato della simulazione:



Si noti inoltre, che avendo eseguito una simulazione funzionale, le forme d'onda non presentano ritardi di propagazione.

3. SINTESI LOGICA SU DISPOSTIVI DELLA FAMIGLIA ALTERA

Per effettuare una sintesi completa, che tenga quindi conto anche degli aspetti di timing selezioniamo nel menù la seguente voce: ***Processing>Start Compilation***. Il processo di sintesi sarà molto più lento rispetto alla sintesi funzionale in quanto il tool, per tenere in considerazione tutti i ritardi dovuti alle interconnessioni ed ai blocchi logici utilizzati, dovrà "mappare" il progetto nell'FPGA prescelta.

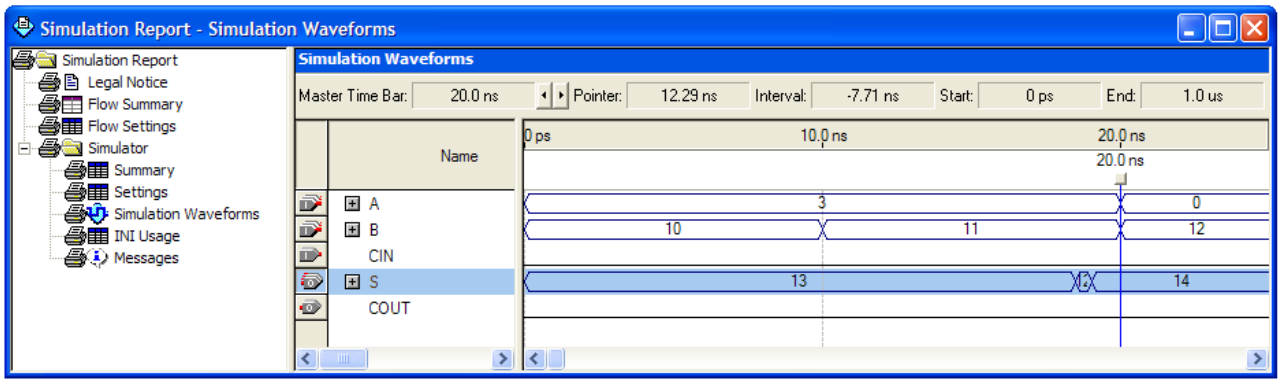
Al termine, nella finestra di **Compilation Report** è possibile analizzare tutte le informazioni relative alla sintesi, ovvero percorsi critici, massima frequenza di funzionamento, tempi di hold e setup, occupazione d'area.

Rispetto alla sintesi funzionale sono presenti ulteriori passi:

- Il Fitter, che alloca i blocchi logici ottenuti dalla "logic synthesis" sugli elementi logici presenti sull'FPGA e ne programma le interconnessioni.
- Il Timing SNF Extractor, che ricava informazioni sui ritardi di propagazine dei segnali attraverso i blocchi logici istanziati nell'FPGA.
- L'Assembler, che crea un file binario, che contiene le configurazioni degli switch e dei blocchi logici e che serve per programmare il dispositivo FPGA, connesso tramite hardware apposito al PC, tramite opportuni tools software.

4. SIMULAZIONE POST-SINTESI

Dopo aver effettuato la sintesi “timing” è possibile effettuare la simulazione del circuito generato con i ritardi effettivi introdotti dai blocchi logici e dalle interconnessioni semplicemente selezionando la voce ***Processing>Start Simulation.*** Si ricordi però di cambiare il tipo di simulazione da *Functional* a *Timing* selezionando ***Assignements/settings/Simulator settings/Simulator mode = Timing***



A differenza della simulazione funzionale, la simulazione tiene ora conto di tutti ritardi in gioco nel circuito. E’ pertanto evidente che rispetto alle forme d’onda viste nella simulazione funzionale, ove l’uscita commutava idealmente istantaneamente al variare degli ingressi, nella simulazione *Timing* l’uscita commuta e si stabilizza con un certo ritardo rispetto all’istante in cui variano i segnali d’ingresso.

Nella simulazione svolta l’ingresso B commuta dopo 10 ns mentre l’uscita inizia a commutare introno ai 18,5 ns, per assumere un valore di transizione non valido e successivamente stabilizzarsi intorno ai 19 ns al corretto valore. Vi è pertanto un ritardo di tra l’uscita e l’ingresso pari a 9 ns.

Analisi delle prestazioni (Timing Analysys T.A)

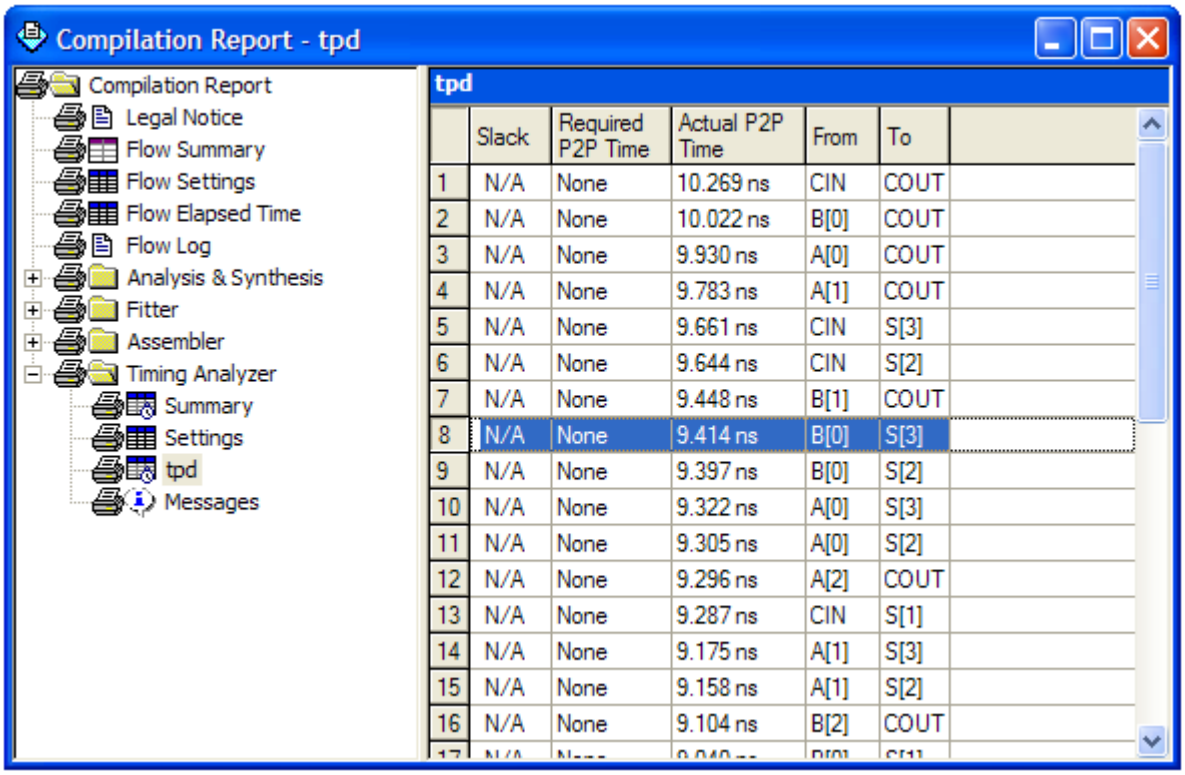
Questa fase consente di conoscere le prestazioni del circuito ottenuto in termini di ritardo di propagazione e di massima frequenza di funzionamento. Essendo il nostro sommatore un blocco puramente combinatorio, non avrà senso definirne una frequenza massima di funzionamento. Analizzeremo quindi i ritardi di propagazione tra nodi di ingresso e uscita.

Per lanciare il Timing Analyzer dal menù di QUARTUS si esegua il comando ***Processing>Start>Start Timing Analyzer***.

Il T.A. può eseguire differenti tipi di analisi, di cui saranno di nostro interesse i seguenti:

- Delay Matrix. Effettua un’analisi limitatamente ai percorsi combinatori che collegano pin di ingresso e pin di uscita. Viene calcolata una matrice in cui le righe corrispondono agli ingressi, le colonne alle uscite, e in cui viene indicato alla posizione (i,j) il ritardo di propagazione dall’ingresso i all’uscita j. Questa matrice non tiene conto dei tempi di setup e hold di eventuali registri. Non tiene conto di eventuali percorsi combinatori “interni”, cioè che iniziano e finiscono internamente e che possono comunque essere più lunghi di quelli indicati! Non è comunque il nostro caso ora, poiché il sommatore è puramente combinatorio, e non avendo registri non può avere percorsi combinatori che si originano internamente.
- Registered Performance. Ha senso solo per reti sincrone, in cui vi sia almeno un registro. Indica la frequenza massima di clock che può essere utilizzata con il circuito

e il dispositivo utilizzati senza incorrere in problemi di funzionamento. Tiene conto anche del tempo di setup dei registri.



tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	10.269 ns	CIN	COUT
2	N/A	None	10.022 ns	B[0]	COUT
3	N/A	None	9.930 ns	A[0]	COUT
4	N/A	None	9.783 ns	A[1]	COUT
5	N/A	None	9.661 ns	CIN	S[3]
6	N/A	None	9.644 ns	CIN	S[2]
7	N/A	None	9.448 ns	B[1]	COUT
8	N/A	None	9.414 ns	B[0]	S[3]
9	N/A	None	9.397 ns	B[0]	S[2]
10	N/A	None	9.322 ns	A[0]	S[3]
11	N/A	None	9.305 ns	A[0]	S[2]
12	N/A	None	9.296 ns	A[2]	COUT
13	N/A	None	9.287 ns	CIN	S[1]
14	N/A	None	9.175 ns	A[1]	S[3]
15	N/A	None	9.158 ns	A[1]	S[2]
16	N/A	None	9.104 ns	B[2]	COUT

Nella precedente tabella troviamo conferma di quanto visto nella simulazione Timing. Si vede che al variare il bit B[0] varia il bit S[3] con 9.414 ns di ritardo.

Si noti che come ci si aspettava il percorso più lungo è quello da CIN a COUT. Tuttavia anche se tale percorso è sicuramente il più lungo in termini di celle attraversate e quindi di ritardi accumulati nell'attraversare le stesse, può in alcuni casi succedere che altri percorsi risultino più critici in termini di timing. Infatti nelle FPGA la configurazione delle interconnessioni gioca un ruolo importante e spesso le interconnessioni hanno ritardi di propagazione maggior dei blocchi combinatori nelle attuali tecnologie.

Floorplan View

Dopo la sintesi Timing è possibile vedere come i blocchi logici dell'FPGA siano stati utilizzati e connessi durante la sintesi logica e il place and route.

Si utilizza un tool detto Floorplan Editor (*Timing Closure FloorPlan*).

I blocchettini bianchi sono quelli rimasti inutilizzati, mentre quelli colorati sono quelli effettivamente utilizzati.

Occupazione delle risorse

E' possibile, dopo la sintesi logica, sapere quante risorse dell'FPGA in termini di blocchi logici ed interconnessioni sono stati utilizzati. Tutte queste informazioni sono visibili nella finestra del ***Compilation Report*** sotto la voce Analysis&Synthesis.

Insieme a moltissime informazioni, viene riportato un riassunto dell'utilizzo di risorse (pin, celle logiche, registri, etc.).

Appendice A: *Codice VHDL del sommatore a 4-bit con riporto*

```
--------  descrizione strutturale: devono essere inseriti nel codice tutti i ---
--------- moduli della gerarchia

library IEEE;
use IEEE.std_logic_1164.all;

---- Half Adder
entity HA is
port ( I1,I2 : in std_logic;
       SUM, CO : out std_logic);
end HA;

architecture BEHAVIOR of HA is
begin
   SUM <= (I1 xor I2);
   CO <= (I1 and I2);
end BEHAVIOR;

-----------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

----- Full Adder
entity FA is
port ( A,B,CIN : in std_logic;
       S, COUT : out std_logic);
end FA;

-- architecture BEHAVIOR of FA is
-- begin
--    S <= (A xor B) xor CIN;
--    COUT <= (A and B) or (B and CIN) or (A and CIN);
-- end BEHAVIOR;

architecture STRUCTURAL of FA is
   component HA
   port ( I1,I2   : in std_logic;
          SUM, CO : out std_logic);
   end component;
signal S1, C1, C2 : std_logic;
begin
   ha1 : HA port map( I1 => B, I2 => CIN, SUM => S1, CO => C1);
   ha2 : HA port map( I1 => A, I2 => S1, SUM => S, CO => C2);

   COUT <= C2 xor C1;
end STRUCTURAL;


-----------------------------------------------------
```

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ADDER4 is
port ( A, B : in  std_logic_vector(3 downto 0);
       CIN  : in  std_logic;
           COUT : out std_logic;
       S    : out std_logic_vector(3 downto 0) );
end ADDER4;


architecture STRUCTURAL of ADDER4 is
   component FA
   port ( A,B,CIN : in std_logic;
          S, COUT : out std_logic);
   end component;
signal K : std_logic_vector(4 downto 0);
begin

    adder_loop : for I in 0 to 3 generate
       fa_I : FA port map ( A => A(I), B => B(I), CIN => K(I), COUT => K(I+1), S
=> S(I) );
    end generate;

    K(0) <= CIN;
    COUT <= K(4);

end STRUCTURAL;
```

```
--- descrizione comportamentale del sommatore  a  4 bit

--  library IEEE;
--  use IEEE.std_logic_1164.all;
--  use IEEE.std_logic_arith.all;
--- use IEEE.std_logic_unsigned.all;


--   entity ADDER4 is
--   port ( A, B : in  unsigned (3 downto 0);
--       CIN  : in  std_logic;
--       COUT : out std_logic;
--       S    : out unsigned(3 downto 0) );
-- end ADDER4;

-- architecture BEHAVIORAL of ADDER4 is

-- signal K : unsigned (4 downto 0);
-- signal Aint, Bint, Cint : unsigned(4 downto 0);
-- begin
--    Aint <=conv_unsigned(A,5);
--    Bint <=conv_unsigned(B,5);
--    Cint <= conv_unsigned(CIN,5);
--    K <= Aint+Bint+Cint;
--
--    S <= K(3 downto 0);
--    COUT <= K(4);
-- end BEHAVIORAL;
```