PROVA FINALE – PROGETTO DI RETI LOGICHE

(aggiornato al 4-02-2021)

Nome: Daniele Casciani

Componente: Divisore intero con resto

1 Obiettivo e specifiche

1.1 Descrizione generale

Si vuole implementare un componente hardware descritto in VHDL che, una volta forniti un dividendo e un divisore, sia in grado di calcolare quoziente e resto della divisone in 32 colpi di clock, emulando l'algoritmo di "lunga divisone", e rendendoli disponibili sulle uscite del dispositivo.

L'algoritmo si basa su eseguire operazioni di shift e sottrazione su dati parziali. Quoziente e resto sono ricavati in seguito alla possibilità o meno di sottrare al minuendo parziale il nostro divisore.

I valori in ingresso e uscita sono vettori a 32 bit, dove quello più significativo si trova a sinistra.

La macchina è controllata dai segnali di input RESET e START(oltre al segnale di CLOCK) i quali, rispettivamente, inizializzano e avviano la computazione.

Infine, la validità del output corrente è garantita dagli ulteriori due segnali, osservabili sull'uscita, ERROR e FINISH.

1.2 Interfaccia del componente

La macchina deve avere la seguente interfaccia:

```
entity Fsa is
    port( N:        in std_logic_vector(31 downto 0);
          D:        in std_logic_vector(31 downto 0);
          CLK:      in std_logic;
          RESET:    in std_logic;
          START:    in std_logic;
          FINISH:   out std_logic;
          ERROR:    out std_logic;
          Q:        out std_logic_vector(31 downto 0);
          R:        out std_logic_vector(31 downto 0)
          );
end Fsa;
```

In particolare:

- CLK: segnale di clock in ingresso.
- RESET: segnale di reset in ingresso che inizializza la macchina.
- START: se 1 salvo gli input, se 0 avvio la computazione
- N: segnale in ingresso che indica il dividendo.
- D: segnale in ingresso che individua il divisore.
- Q: segnale d'uscita che riporta il quoziente della divisione.
- R: segnale d'uscita che riporta il resto della divisione.
- FINISH: segnale che indica quando la macchina ha terminato la computazione

- ERROR: segnale che indica quando è stato inserito un divisore non valido

1.3 Corretto utilizzo del dispositivo

Per inserire correttamente gli input il segnale di START deve essere portato ad 1, l'elaborazione inizierà solo quando lo stesso segnale verrà riportato a 0.

Il segnale FINISH rimarrà a 1 per tutto l'intervallo di tempo per cui il segnale in uscita è considerato un risultato valido.

Quando il segnale di ERROR è alto gli altri valori delle uscite non sono considerati risultati validi.

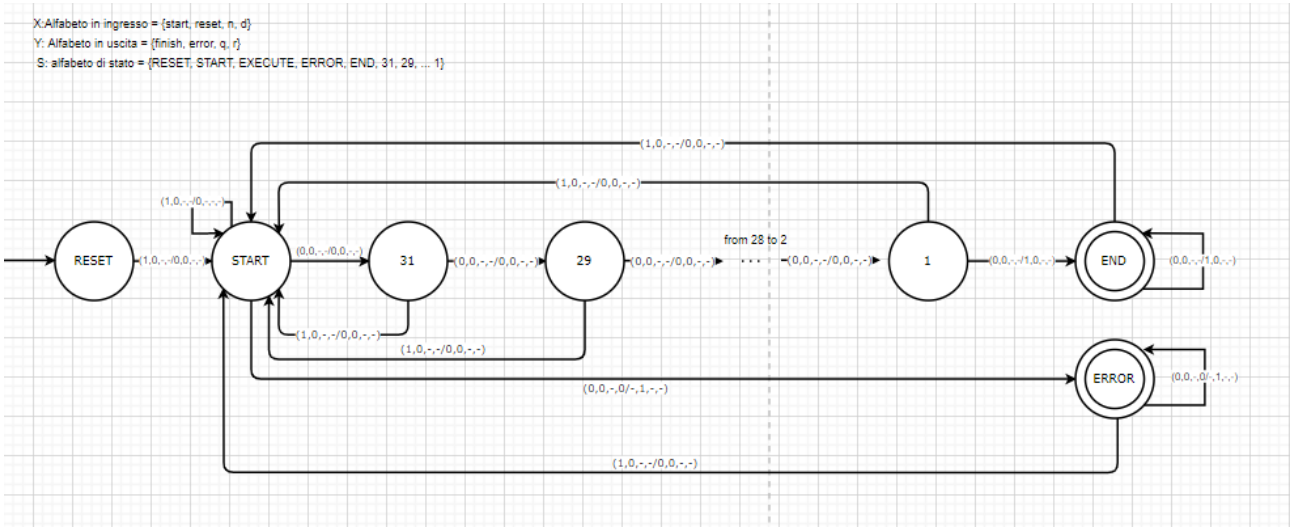
1.4 Strumenti sintesi utilizzati

Xilinx ISE Design Suite

- Target FPGA (xc3s200-5pq208)

2 Implementazione

2.1 Diagramma degli stati



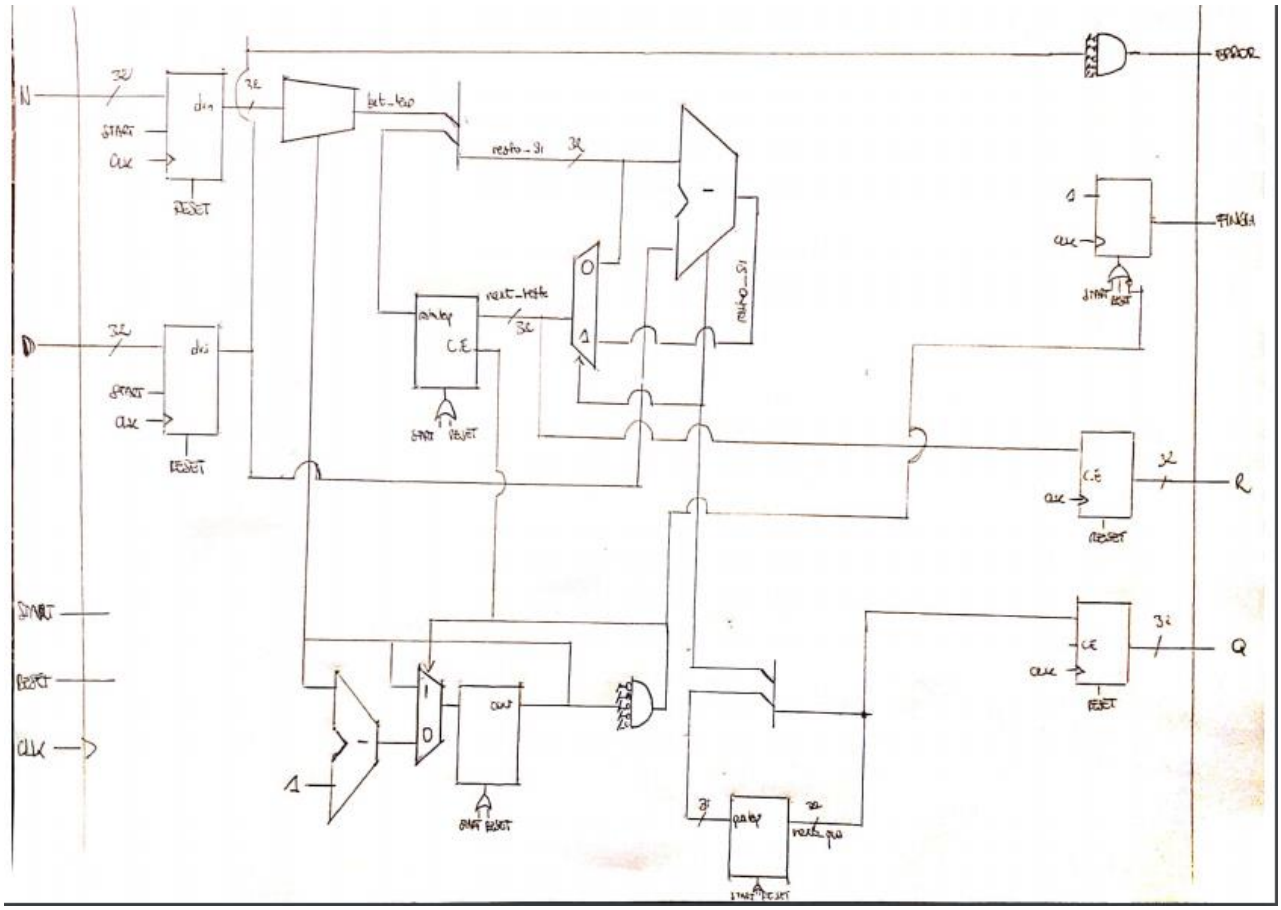
N.B: sono stati omessi gli archi uscenti da ogni stato e diretti verso lo stato di RESET quando il segnale corrispondente viene posto ad 1 indipendentemente dallo stato attuale o dagli altri ingressi della macchina. Inoltre, è stato omesso il segnale di ingresso, in quanto ogni transizione di stato avviene quando clk è 1.

<u>Descrizione stati:</u>

- RESET: stato di reset necessario per il corretto funzionamento del dispositivo. Si rimane nello stato fino a che RESET rimane a 1 e non viene attivato lo START.
- START: quando START = 1 vengono salvati gli input, dunque viene fatto un check sulla validità dei dati. Si rimane nello stato fino a che start non torna a 0.
- 31, 29…1: comprendono la fase di calcolo vera e propria (operazione di confronto, shift e sottrazione), operazioni iterate sui valori parziali che mutano ad ogni stato.

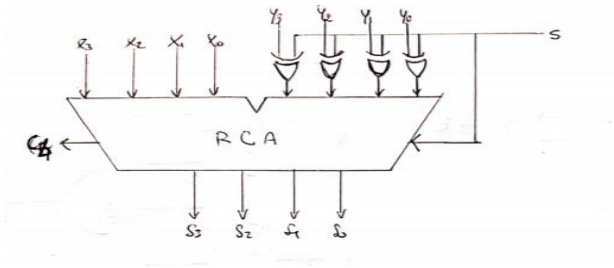
- END: quoziente e resto sono resi disponibili sulle uscite. Si rimane in questo stato fino a quando non viene riattivato il segnale di start.
- ERROR: se viene letto un divisore = 0 e START passa a 0 arriviamo nello stato di errore dove il segnale corrispondente vale 1 e i risultati sulle altre uscite non sono considerati validi

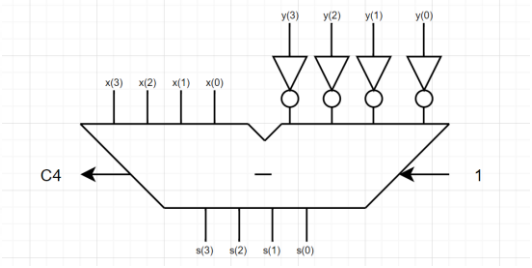
2.2 Struttura del componente



2.3 Componenti fondamentali e funzioni importanti

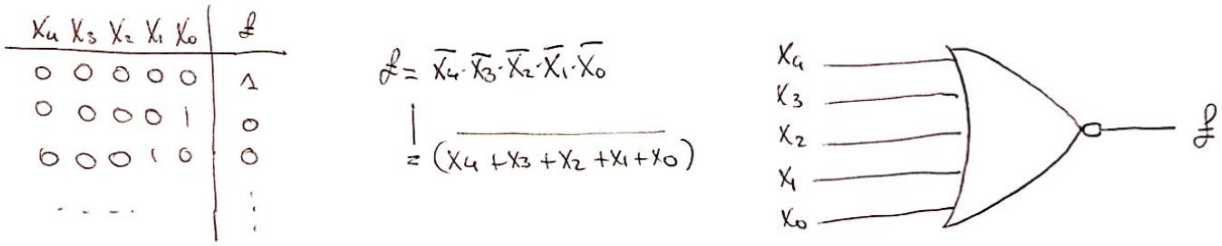
Sommatore/sottrattore





**Progettato come sommatore/sottrattore anche se nel progetto svolge solo la funzione di sottrattore.*

Comparatore



Funzione '>='

Nello stato di EXECUTE c'è bisogno di effettuare un confronto tra due valori. Non avendo ha disposizione un componente per tale funzione si procede per sintesi strutturale.

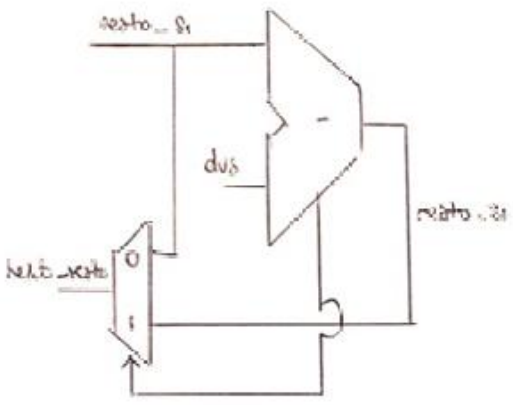
Possiamo verificare la maggioranza, del minuendo rispetto al sottraendo, usando la codifica in complemento a due e valutando il bit di riporto che deriva dalla somma dei due termini.

Il carry varrà 1 se il confronto è positivo 0 altrimenti.

La scelta di usare il complemento a due ci permette inoltre di calcolare la sottrazione parallelamente al confronto, facendo la somma del minuendo e del complemento a due del sottraendo.

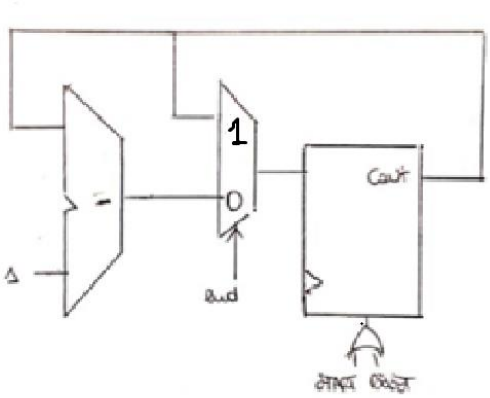
```
if carry_out == 1 then
   R = R + D //D in complemnto a 2
   Q[i]=1
else
   Q[i]=0
end if;
```

**in alto uno pseudo codice che descrive l'implementazione della funzione; la sintesi strutturale in basso.*



Contatore

Si necessita dell'ausilio di un contatore binario che scandisca le iterazioni della computazione e i clock rimanenti al termine.



**sintesi strutturale contatore binario naturale con C.E.*

2.4 Segnali interni

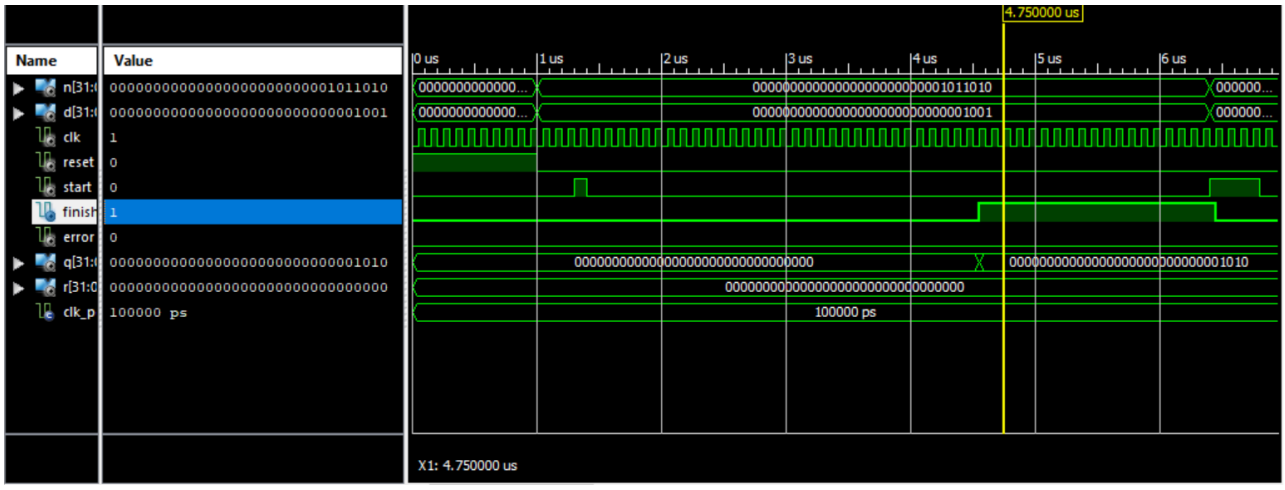
Il componente usa diversi segnali interni durante la computazione. I più importanti dal punto di vista concettuale sono:

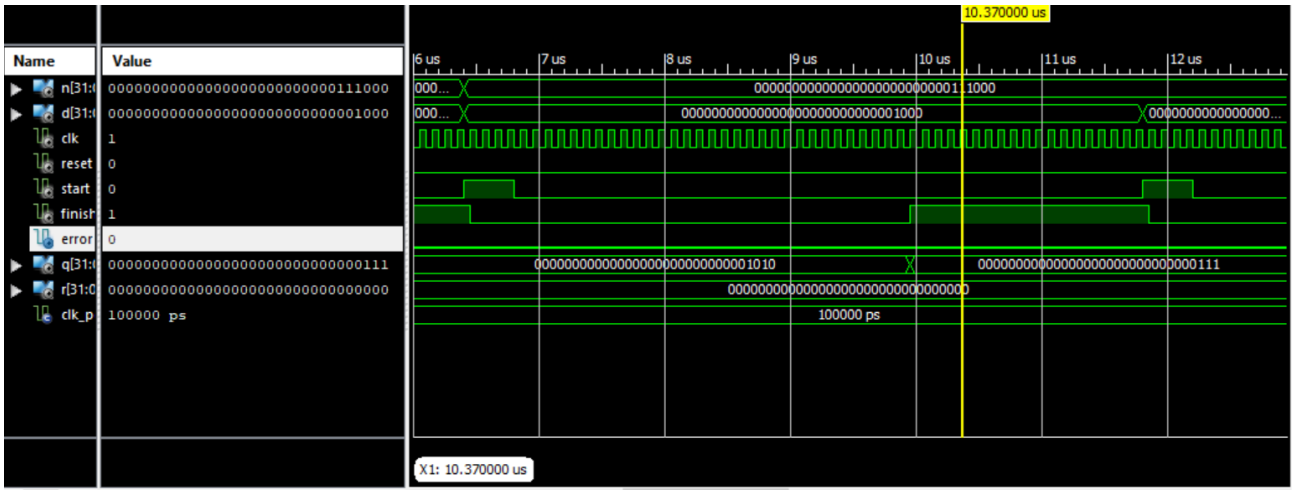
- resto_temp e quo_temp: sono i "dati parziali" con i quali gli stati 31, 29, … 1 operano ad ogni clock per trovare il Q[i] (insieme a count descrivono lo stato corrente).
- next_resto e next_quo: sono gli stati prossimi dei precedenti.
- Count: tiene conto dei clock trascorsi.

3 Test

Il dispositivo e ciascuno dei suoi componenti hanno superato tutti i test ai quali è stato sottoposto. Ogni test è stato eseguito in Behavioral Simulation e Post-Route SImulation.

Di seguito vengono riportati alcuni esempi.




Name
Value
n[31:
00000000000000000000000000111000
d[31:
00000000000000000000000000001000
clk
1
reset
0
start
0
finish
1
error
0
q[31:
00000000000000000000000000000111
r[31:0
00000000000000000000000000000000
clk_p
100000 ps
10.370000 us
6 us
7 us
8 us
9 us
10 us
11 us
12 us
X1: 10.370000 us