**Davide Carini (Matricola:890064 – CodicePersona: 10568649)**

**PROVA FINALE**

**(Progetto n° 11 di Reti Logiche)**

**Corso di Reti Logiche**

**Anno Accademico 2019/2020**

**Professore: Carlo Brandolese**

**Indice**

1. Introduzione
   1. Scopo del Progetto
   2. Esempio a 8 bit
2. Design architettura
   1. Interfaccia del componente ( top level )
   2. Entity principali
3. Test
4. Riferimenti

**INTRODUZIONE**

* Scopo del Progetto

Lo scopo è progettare un divisore intero su 32 bit basato sul metodo di “divisione lunga”. Siano N il dividendo, D il divisore, Q il quoziente , R il resto e i la dimensione delle parole. L’algoritmo è descritto dal seguente pseudocodice:

if( D == 0 ) {

error();

}

Q = 0;

R = 0;

for(i = n-1; i >= 0; i-- ) {

R = R << 1 ;

R[0] = N[i] ;

if( R ≥ D ) {

R = R – D ;

Q[i] = 1 ;

} else {

Q[i] = 0 ;

}

}

Deve essere realizzata una rete sequenziale che implementi il divisore basato su tale algoritmo. Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

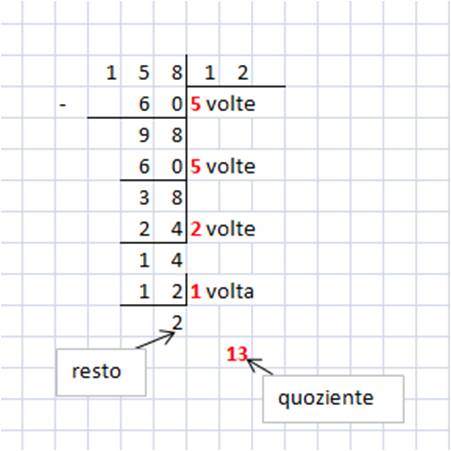
L’algoritmo di “divisione lunga” è l’algoritmo standard più utilizzato per la divisione di numeri espressi in notazione decimale spesso chiamato come divisione in colonna.

Figura 1. Esempio di "Lunga Divisione"

Come primo approccio ho pensato a capire lo scopo del progetto e comprendere il problema da un punto di vista generale più ad ampia veduta rispetto ad una descrizione subito dettagliata dei sottoproblemi presenti. Per fare ciò, ho tradotto il ciclo for in una serie di processi in ordine temporale descritto tramite diagrammma di flusso:

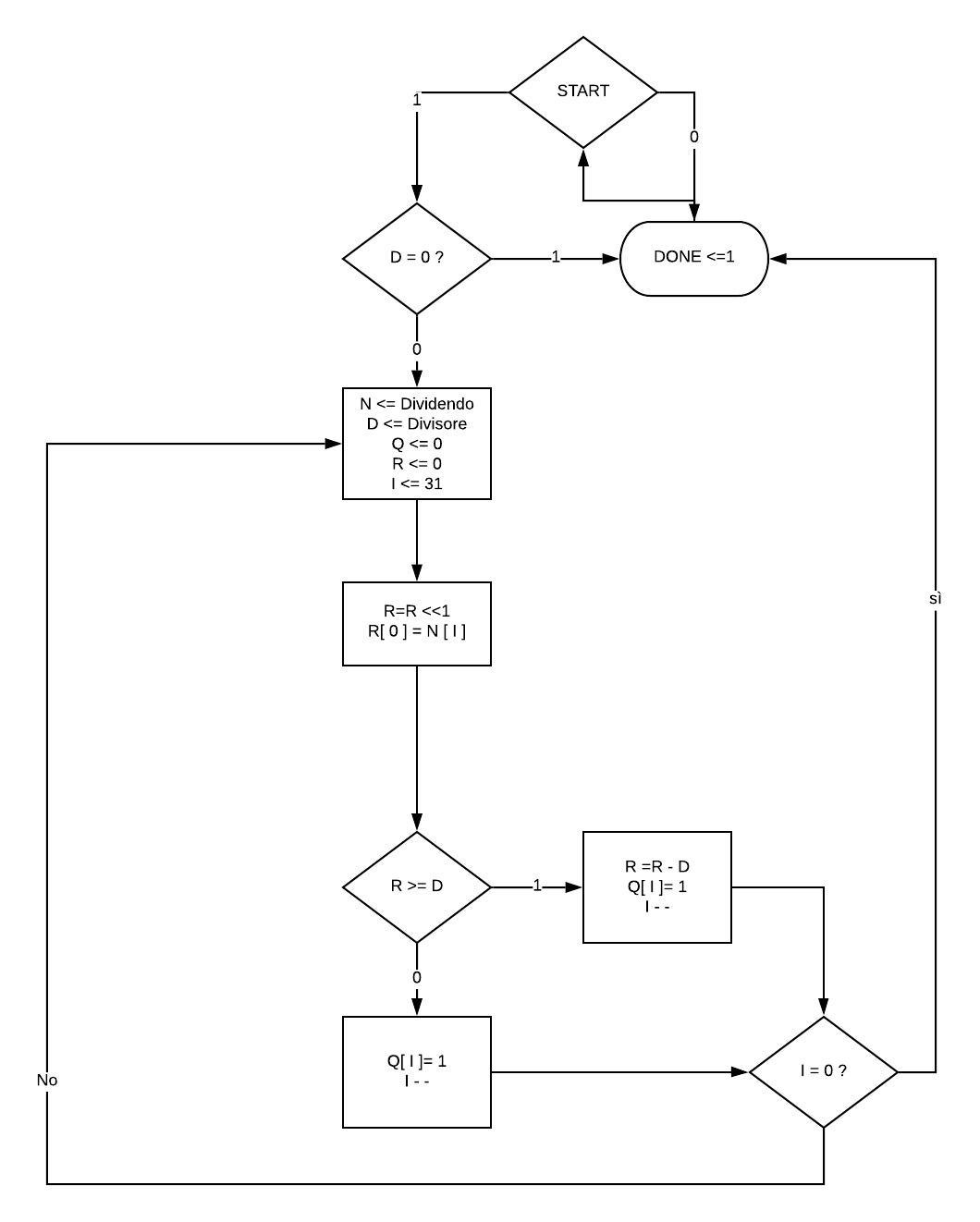


Figura 2. Diagramma di flusso

Essendo un ciclo le istruzioni devono essere eseguite in modo sequenziale.Inizialmente ho pensato ad un possibile utilizzo di una pipeline che potesse eseguire le diverse istruzioni parallelamente. Il problema è che a differenza di una cpu che ad ogni ciclo di pipeline carica una nuova istruzione , in questa caso il risultato dell’istruzione corrente verrà utilizzato in quella successiva. Questo mi ha portato all’esclusione dell’utilizzo della pipeline che in questo problema perderebbe di significato.

Ho pensato quindi di risolvere l’algoritmo utilizzando un divisore che descriverò dopo l’esempio di implementazione del ciclo for.

* Esempio ( con n=8 bit )

Dimostro tramite un esempio il corretto funzionamento dell’algoritmo.

Pongo N=100011112 (14310) e D=000010012(910). Verifico inizialmente che il divisore sia diverso da 0 (condizione rispettata). Dopo inizializzo quoziente e resto a 0.

Ciclo FOR

i=n-1=7 i=3

R= 00 R=010000

R=01 R=010001

Condizione if false Condizione if true

Q=0 R=1000

i=6 Q=00001

R=010 i=2

R=010 R=10000

Condizione if false R=10001

Q=00 Condizione if true

i=5 R=1000

R=0100 Q=000011

R=0100 i=1

Condizione if false R=10000

Q=000 R=10001

i=4 Condizione if true

R=01000 R=1000

R=01000 Q=0000111

Condizione if false i=0

Q=0000 R=10000

R=10001

Condizione if true

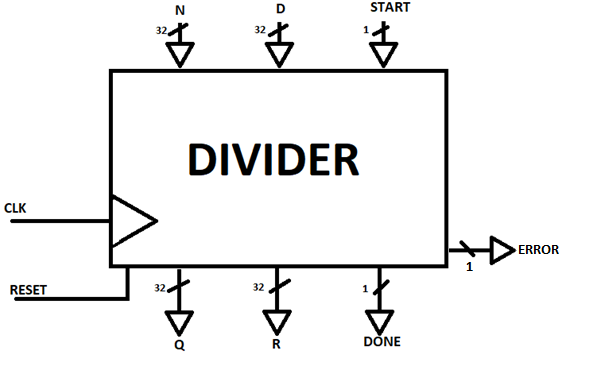
R=10002 -> 810

Q=000011112 -> 1510

La divisione è stata fatta con parole di 8 bit in modo tale da semplificare il ciclo iterativo. Con parole di 32 bit, il ciclo for avrà 32 iterazioni differenti.

**DESIGN ARCHITETTURA**

* Interfaccia del componente (top level)

****

Il divisore è composto da 5 segnali di ingresso e 4 segnali di uscita. Gli ingressi rappresentano:

* N rappresenta il dividendo a 32 bit
* D rappresenta il divisore a 32 bit
* START è il segnale di ingresso che quando vale 1 corrisponde all’inizio dell’ algoritmo
* CLK rappresenta il clock utile a sincronizzare i dispositivi interni al divisore
* RESET è il segnale che consente il ripristino dei valori interni al dispositivo

Le 4 uscite invece rappresentano:

* Q corrisponde al quoziente a 32 bit della divisione
* R rappresenta il resto a 32 bit della divisione
* ERROR è il segnale che esprime la condizione di errore ovvero quando il divisore è nullo
* DONE è il segnale che viene settato ad 1 quando il risultato della divisione ovvero il quoziente ed il resto sono pronti

Il divisore (componente top level) mi servirà solo per mappare i sottocomponenti e quindi permettere la comucazione tra di loro.

* Entity principali

I principali sotto componenti che ho pensato di utilizzare sono :

* Left shifter
* Full subtractor
* Registri
* Contatore
* Comparatore

Vediamo le seguenti entity nel dettaglio:

**LEFT SHIFTER**

Lo shifter è utilizzato per effettuare il left shift del resto nella prima istruzione del ciclo. Al resto viene troncato il MSB e viene aggiunto uno 0 come LSB. Questo risultato sarà quindi il nuovo valore del resto che dovrà essere utilizzato nell’istruzione successiva. In vhdl per implementare lo hift dovrà fare riferimento al costrutto di slicing dove uno slice rappresenta una porzione di vettore e l’ operatore di concatenamento.

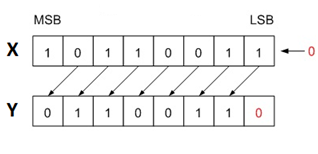
****

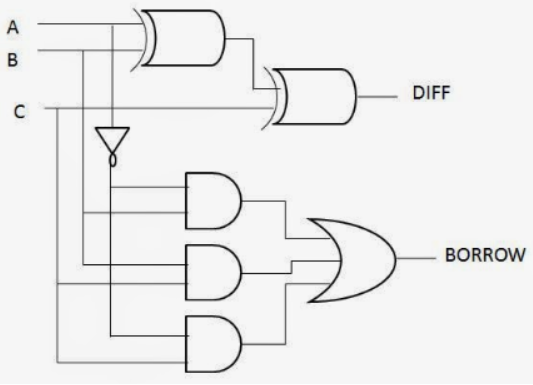
Figura 3. Eempio di left shift

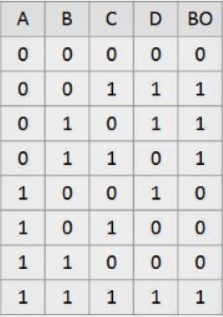


**FULL SUBTRACTOR a 32 bit**



Per descrivere il sottrattore ho preso spunto dal full adder. Il full subtractor a 32 bit conterrà 32 subtractor basici. La sottrazione avrà sempre come risultato un numero positivo in quanto la sottrazione verrà fatta solo se è rispettata la condizione: R >=D.

Il sottrattore prende come ingressi R e D e restituirà il nuovo valore di R.



**Difference = A XOR B XOR C**

**Borrow = (NOT (A) \* B) + B \* C + (NOT (A) \* C)**

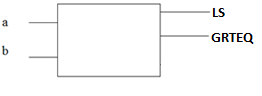
Figura 4.Circuito logico sottrattore

Figura 5. Tavola della verità



**COMPARATORE a 32 bit**

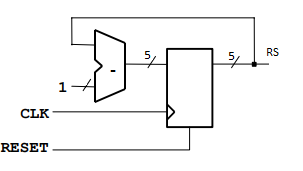
Il comparatore è un circuito combinatorio che prende come ingressi il resto e il divisore e setta LS a 1 se R < D oppure setta GRTEQ a 1 se R >= D.

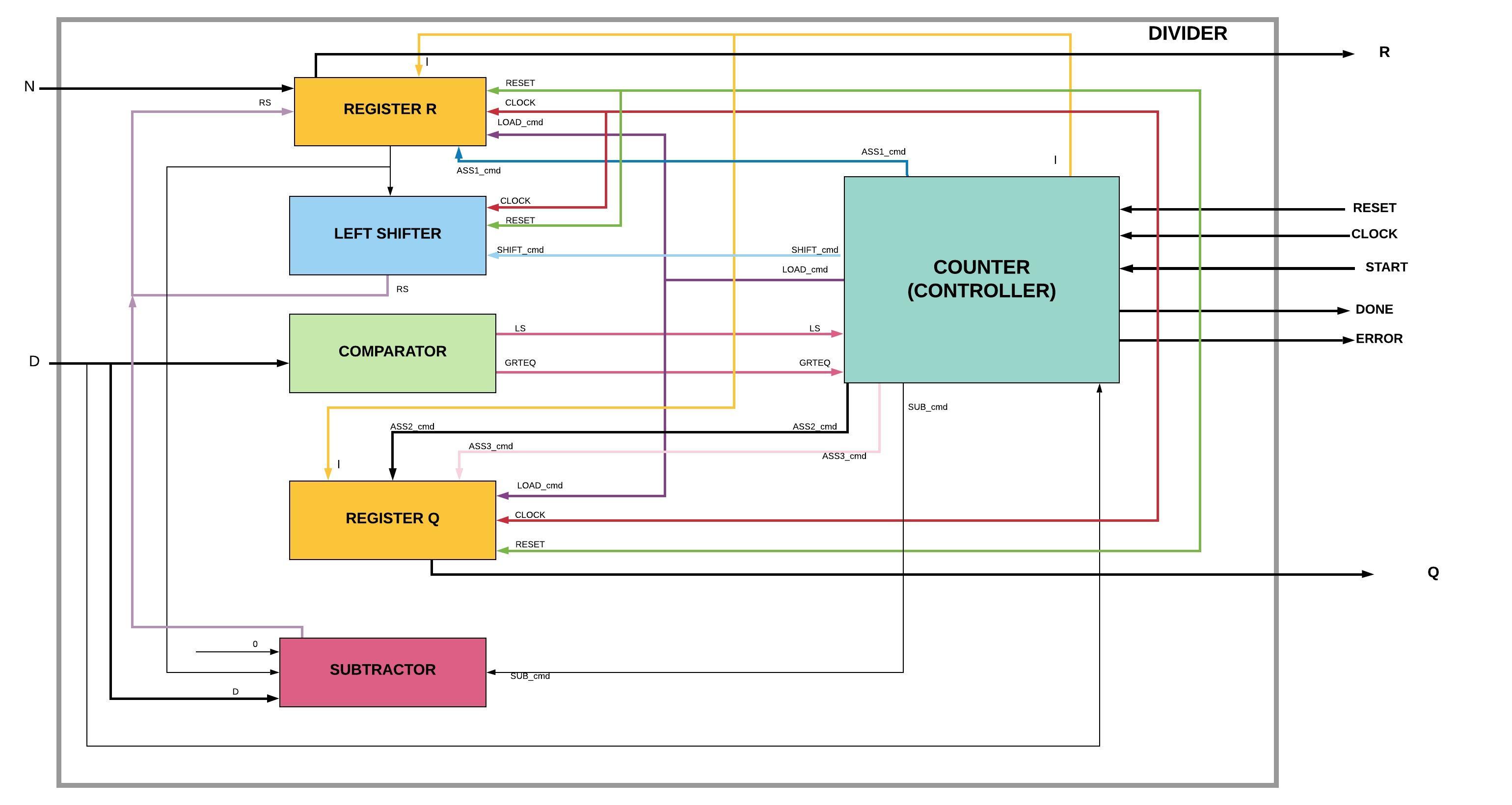
****

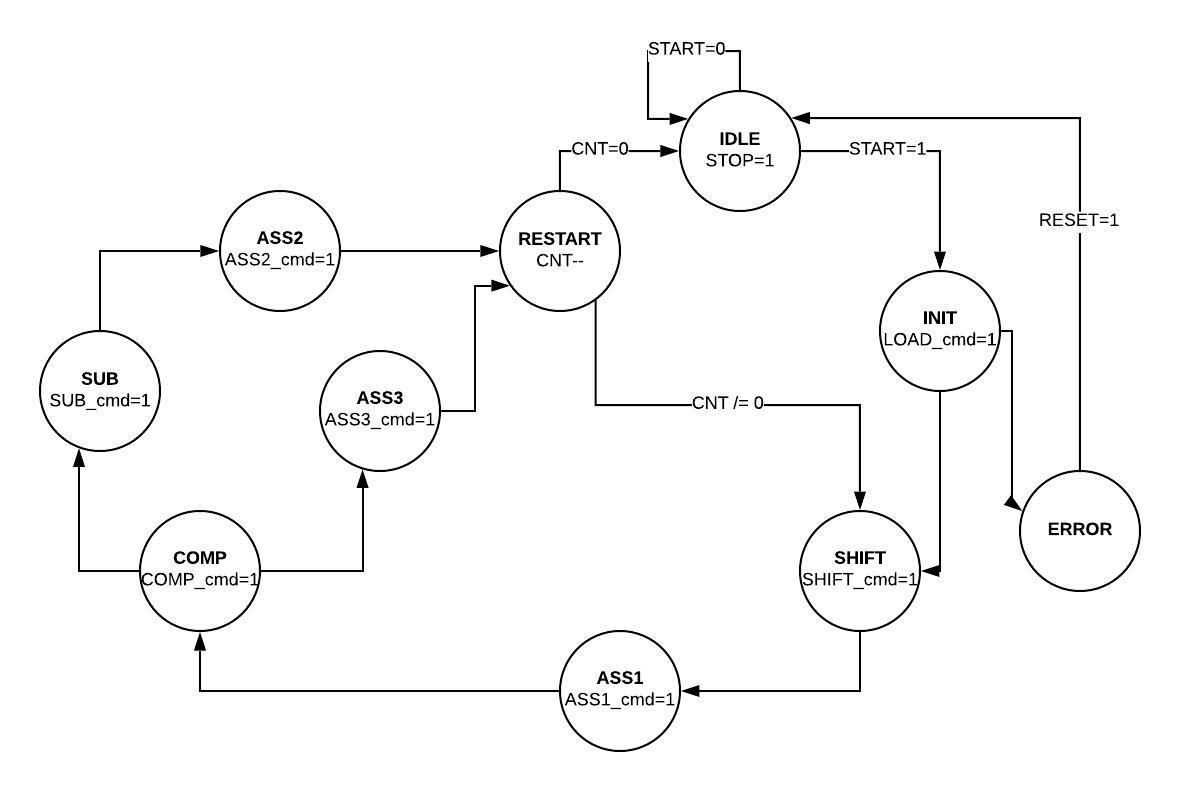
**REGISTER 32 bit**

**CONTATORE**



Il contatore viene utilizzato per effettuare le 32 iterazioni del ciclo for. Il contatore sarà un contatore down ovvero opera in decremento. Inizialmente il counter sarà a “11111” e dopo verrà decrementato fino a diventare “00000”. Può essere utilizzato un contatore modulo 2n che lavora al contrario. Poiché dobbiamo rappresentare 32 iterazioni differenti, saranno necessari 5 bit.





RIFERIMENTI

<https://www.allaboutcircuits.com/technical-articles/how-to-vhdl-description-of-a-simple-algorithm-the-control-path/>

<https://staff.emu.edu.tr/muhammedsalamah/Documents/CMPE224/labs/Exp7.pdf>

<https://www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine-in-vhdl/>

<http://unina.stidue.net/Architettura%20dei%20Sistemi%20di%20Elaborazione/Materiale/Addizionatori%20v1.01.pdf>

Giovanni d’aliesio

**REMEMBER:**

Macchina a Stati

Il documento deve descrivere l’architettura del circuito e non la sua implementazione. Questo significa che la sua macchina a stati non la deve sintetizzare lei e neppure descrivere in termini di diagrammi degli stati.

Spiegare la funzione dell’architettura.

La macchina a stati è un contatore

Esempi

Descrivere architettura e le entity principali che lo compongono.

Entity principali: full subtractor, shifter, comparator, mux , counter

Dovrei avere 2 registri per salvare Q e R ad ogni iterazione del ciclo.

If si implementa con mux pilotato da R >= D

Le architecture è preferibile chiamarle RTL oppure struct.

EMAIL BRANDOLESE  
La cosa migliore e che lei mi presenti una specifica, ovvero un documento che descrive  
il problema ed il modo in cui lo si intende affrontare. Per intenderci, ci deve essere almeno  
la definizione dei segnali di interfaccia del componente e dei sottocomponenti principali,

una breve descrizioni delle funzioni svolte dai moduli principali e una ipotesi su come verificare  
in simulazione la correttezza del progetto mediante test bench.

In questo modo mi sarà possibile capire se sta procedendo nella giusta direzione.