Davide Carini (Matricola: 890064 - Codice Persona: 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

Macchina a Stati

Testin

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.

**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 n 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.

* Esempio ( con n=8 bit )

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

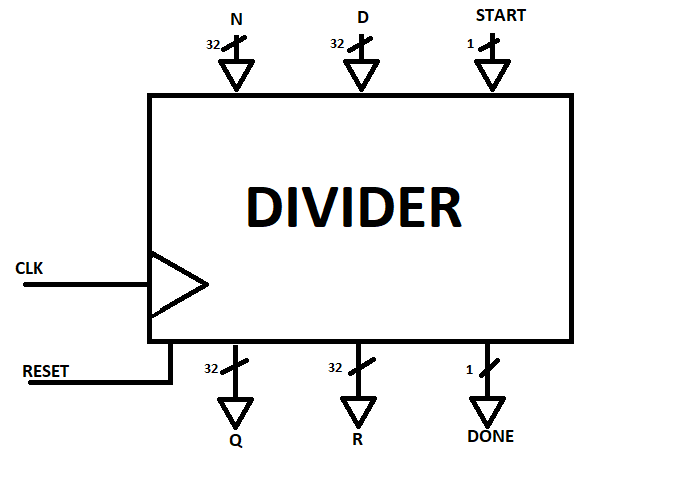
R=10002 -> 810

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 a livello di top level



library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity DIVIDER is

generic(N:integer :=32);

port(

CLK : IN std\_logic;

RST: IN std\_logic;

START : IN std\_logic;

D: IN std\_logic\_vector (N-1 downto 0);

N: IN std\_logic\_vector (N-1 downto 0);

Q: OUT std\_logic\_vector(N-1 downto 0);

R: OUT std\_logic\_vector(N-1 downto 0);

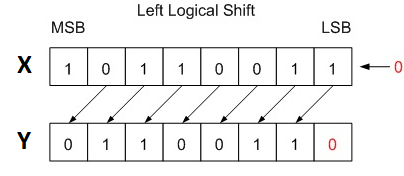
DONE: OUT std\_logic

);

end DIVIDER;

* Entity principali

**LEFT SHIFTER (con N=32 bit)**

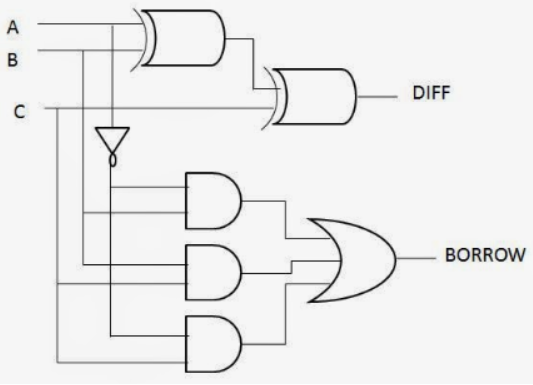
**

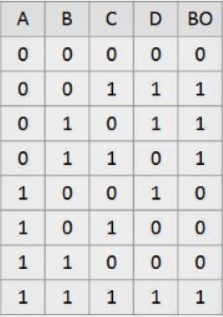
Utilizzo il costrutto di slicing (slice=porzione di un vettore) e l’operatore di concatenamento.



**FULL SUBTRACTOR ( con n=32 bit)**

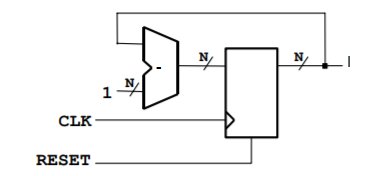
<http://unina.stidue.net/Architettura%20dei%20Sistemi%20di%20Elaborazione/Materiale/Addizionatori%20v1.01.pdf>





Difference = A XOR B XOR C Borrow = A' \* B + B \* C + A' \* C

**Contatore DOWN con N=5 bit**

****

***IMPLEMENTATION counter***

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity left\_shifter is

generic( N: integer :=32);

port(

R: IN std\_logic\_vector(N-1 downto 0);

RS: OUT std\_logic\_vector(N-1 downto 0)

);

end left\_shifter;

architecture rtl of left\_shifter is

begin

RS <= R(N-2 downto 0) & '0';

end rtl;

***TEST BENCH counter***

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY TB\_COUNTER\_DOWN IS

generic( N: integer :=5);

END TB\_COUNTER\_DOWN;

ARCHITECTURE behavior OF TB\_COUNTER\_DOWN IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT COUNTER\_DOWN

PORT(

CLK : IN std\_logic;

RESET : IN std\_logic;

I : OUT std\_logic\_vector(0 to N-1) );

END COMPONENT;

--Inputs

signal CLK : std\_logic ;

signal RESET : std\_logic ;

--Outputs

signal I : std\_logic\_vector(0 to N-1);

-- Clock period definitions

constant CLK\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: COUNTER\_DOWN PORT MAP (

CLK => CLK,

RESET => RESET,

I => I

);

-- Clock process definitions

CLK\_process :process

begin

clk <= '0';

wait for 10 ns;

clk <= '1';

wait for 10 ns;

end process;

process

begin

reset <= '1';

wait for 100 ns;

reset <= '0';

wait;

end process;

END;

**COMPARATOR**

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

-- n-bit Comparator (ESD book figure 2.5)

-- by Weijun Zhang, 04/2001

--

-- this simple comparator has two n-bit inputs &

-- three 1-bit outputs

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

library ieee;

use ieee.std\_logic\_1164.all;

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

entity Comparator is

--generic(n: natural :=2);

--port( A: in std\_logic\_vector(n-1 downto 0);

-- B: in std\_logic\_vector(n-1 downto 0);

port( A: in std\_logic\_vector(1 downto 0);

B: in std\_logic\_vector(1 downto 0);

less: out std\_logic;

equal: out std\_logic;

greater: out std\_logic

);

end Comparator;

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

architecture behv of Comparator is

begin

process(A,B)

begin

if (A<B) then

less <= '1';

equal <= '0';

greater <= '0';

elsif (A=B) then

less <= '0';

equal <= '1';

greater <= '0';

else

less <= '0';

equal <= '0';

greater <= '1';

end if;

end process;

end behv;

**MUX**

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

-- VHDL code for 4:1 multiplexor

-- (ESD book figure 2.5)

-- by Weijun Zhang, 04/2001

--

-- Multiplexor is a device to select different

-- inputs to outputs. we use 3 bits vector to

-- describe its I/O ports

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

library ieee;

use ieee.std\_logic\_1164.all;

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

entity Mux is

port( I3: in std\_logic\_vector(2 downto 0);

I2: in std\_logic\_vector(2 downto 0);

I1: in std\_logic\_vector(2 downto 0);

I0: in std\_logic\_vector(2 downto 0);

S: in std\_logic\_vector(1 downto 0);

O: out std\_logic\_vector(2 downto 0)

);

end Mux;

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

architecture behv1 of Mux is

begin

process(I3,I2,I1,I0,S)

begin

-- use case statement

case S is

when "00" => O <= I0;

when "01" => O <= I1;

when "10" => O <= I2;

when "11" => O <= I3;

when others => O <= "ZZZ";

end case;

end process;

end behv1;

architecture behv2 of Mux is

begin

-- use when.. else statement

O <= I0 when S="00" else

I1 when S="01" else

I2 when S="10" else

I3 when S="11" else

"ZZZ";

end behv2;

Pensare ad architettura pipeline (non so se la strada è quella corretta…)

architeture pipe of SAMPLE is

signal T1, T2: std\_logic;

begin

regs: process( CLK )

begin

if( CLK’event and CLK = ’1’ ) then

T1 <= h( X );

T2 <= g( T1 );

Y <= f( T2 );

end if;

end process;

end pipe;

<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>