Relazione

Progetto per la sessione d'esame estiva
2023 / 2024

Calcolo della superficie di un solido composto (cubo sormontato da piramide)

AUTORI

Sestri Daniele
matricola: 320713

Piovaticci Luca
matricola: 328235

Università degli Studi di Urbino Carlo Bo
Insegnamento di Reti Logiche

Contents

1 Specifica **4**
- 1.1 Scopo del progetto . . . 4
- 1.2 Specifica funzionale . . . 4
  - 1.2.1 Relazioni tra i dati di ingresso e il dato in uscita . . . 4
  - 1.2.2 Dominio e codominio . . . 5
- 1.3 Specifica parametrica . . . 5
- 1.4 Ulteriore scelta di progetto . . . 5

2 Impostazione del progetto a livello RT **6**
- 2.1 Data Flow Graph . . . 6
- 2.2 Risorse . . . 6

3 Progettazione delle risorse a livello gate **7**
- 3.1 Porte logiche elementari . . . 7
  - 3.1.1 NOT . . . 7
  - 3.1.2 NAND . . . 7
  - 3.1.3 NOR . . . 8
- 3.2 Porte logiche composte . . . 9
  - 3.2.1 AND . . . 9
  - 3.2.2 EXOR . . . 9
- 3.3 Registri . . . 10
  - 3.3.1 Flip Flop D Edge Triggered . . . 11
  - 3.3.2 Flip Flop D Level Sensitive . . . 11
  - 3.3.3 Flip Flop SR . . . 12
  - 3.3.4 Latch SR . . . 12
- 3.4 Multiplexer . . . 14
- 3.5 Demultiplexer . . . 16
- 3.6 Macro funzionali . . . 17
  - 3.6.1 Half adder . . . 17
  - 3.6.2 Full Adder . . . 17
  - 3.6.3 Addizionatore . . . 18
  - 3.6.4 Moltiplicatore . . . 18

4 Data Path **21**

5 Control Unit **22**
- 5.1 Diagramma degli stati . . . 22
- 5.2 Codifica degli stati e segnali di controllo . . . 23
- 5.3 Tabelle della verità della funzione d'uscita e di stato futuro . . . 24
- 5.4 Implementazione della Control Unit . . . 24

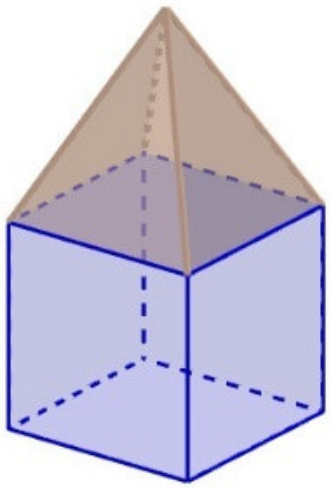
6 Simulazione e analisi del progetto **25**
- 6.1 Verifica funzionale . . . 25
  - 6.1.1 Avvio della simulazione . . . 25
  - 6.1.2 Avvio primo clock con inizializzazione . . . 26
  - 6.1.3 Avvio secondo ciclo di clock con abilitata la transizione degli stati . . . 27
  - 6.1.4 Avvio terzo ciclo di clock con abilitata la transizione degli stati . . . 28
  - 6.1.5 Evidenza Overflow Addizionatore . . . 29
  - 6.1.6 Evidenza Overflow Moltiplicatore . . . 30

6.2 Simulazioni della CU e delle Macro funzionali 31
6.2.1 Control Unit 31
6.2.2 Addizionatore 31
6.2.3 Moltiplicatore 32
6.3 Stima di complessità circuitale e prestazioni 33
6.3.1 Stima dell'area, del tempo di contaminazione e di propagazione 33
6.3.2 Complessità e prestazioni 35

1 Specifica

1.1 Scopo del progetto

Lo scopo del progetto è l'implementazione di un circuito per calcolare la superficie totale visibile di un solido composto da un cubo sormontato da una piramide a base quadrata avente gli spigoli di base coincidenti con gli spigoli di una faccia del cubo.



La progettazione sarà supportata da uno strumento CAD (Computer Aided Design[1]) per l'elettronica. Lo strumento di progettazione e simulazione utilizzato per la realizzazione del circuito sarà TKGate versione 2.1 per sistema operativo Linux Mint 21.1 Cinnamon.

1.2 Specifica funzionale

I dati richiesti in ingresso saranno:

L = lato di una faccia del cubo

A = apotema della piramide

Il dato in uscita sarà la superficie totale visibile, che corrisponderà alla superficie delle 5 facce esposte del cubo più la superficie delle 4 facce laterali esposte della piramide.

1.2.1 Relazioni tra i dati di ingresso e il dato in uscita

La relazione tra i dati in ingresso e di uscita è evidenziata dalla seguente espressione aritmetica

$$S_{\text{tot}} = 5L^2 + \frac{4LA}{2}$$

implementeremo sul circuito la sua versione corrispondente semplificata:

$$S_{\text{tot}} = L(5L + 2A)$$

[1] Progettazione assistita da computer

1.2.2 Dominio e codominio

Il dominio e il codominio della funzione in questione sono definiti sui numeri naturali.
Nel contesto del progetto, gli operandi e i relativi risultati saranno rappresentati a 8 bit. La formula di rappresentazione dei dati è $2n - 1$ con n corrispondente al numero di bit utilizzati. Di conseguenza, $2^8 - 1 = 255$, quindi si potranno rappresentare numeri interi decimali da 0 a 255.

Verranno implementati meccanismi di segnalazione per l'utente in caso di overflow, sia nelle operazioni di addizione che di moltiplicazione.

1.3 Specifica parametrica

L'obiettivo principale del progetto è ottimizzare l'utilizzo delle risorse attraverso l'implementazione del Resource Sharing.

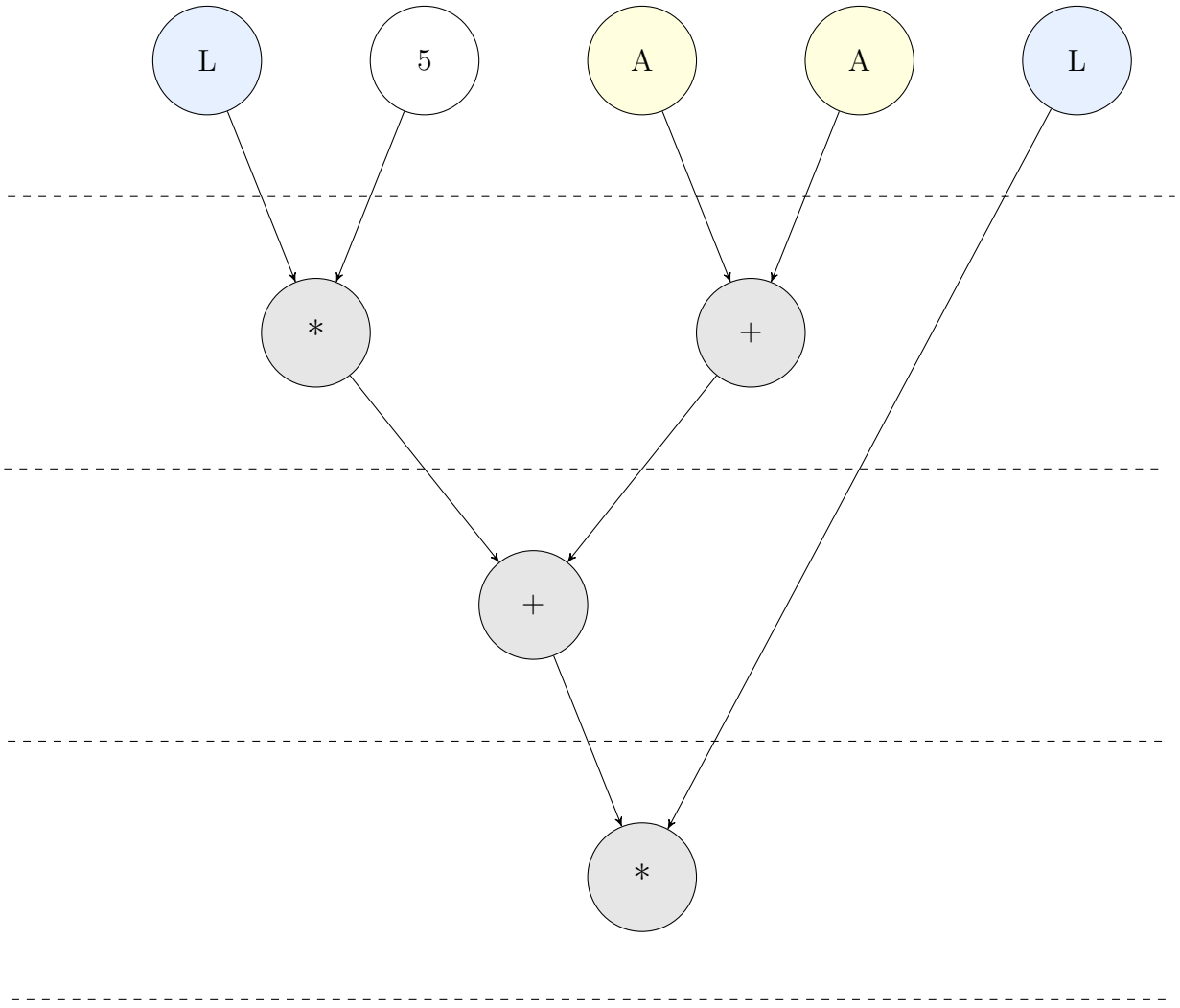
Il circuito è stato progettato per eseguire le operazioni sequenzialmente in 3 cicli di clock distinti per consentire l'utilizzo di un unico addizionatore e un unico moltiplicatore in Resource Sharing e di bilanciare al meglio ogni ciclo di clock.

1.4 Ulteriore scelta di progetto

Come ulteriore scelta di progetto, si è deciso che il risultato completo sarà reso disponibile esclusivamente al termine del ciclo di transizioni, evitando così la presentazione di risultati intermedi. L'implementazione di questa strategia garantisce l'integrità e la coerenza dei dati all'uscita del circuito, assicurando che solo informazioni complete siano presentate come output finale.

2 Impostazione del progetto a livello RT

2.1 Data Flow Graph



2.2 Risorse

Le risorse sono implementate in tecnologia FC-MOS [2]. I transistors utilizzati sono pMOS per il circuito di "Pull Up" ed nMOS per il circuito di "Pull Down".

Le risorse principali impiegate nel progetto sono:

- Cinque registri a 8 bit;
- Tre multiplexer a 2 vie a 8 bit;
- Due de-multiplexer a 2 vie a 8 bit;
- Un addizionatore Ripple Carry Adder a 8 bit;
- Un moltiplicatore a 8 bit;
- Una control unit.

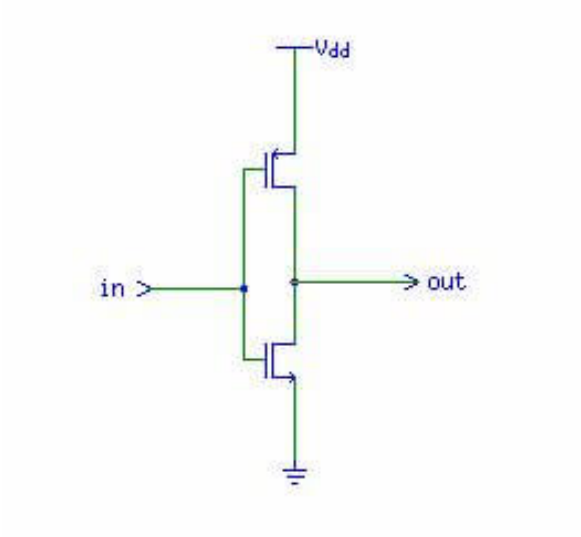
[2] Fully Complementary Metal Oxide Semiconductor

3 Progettazione delle risorse a livello gate

3.1 Porte logiche elementari

3.1.1 NOT

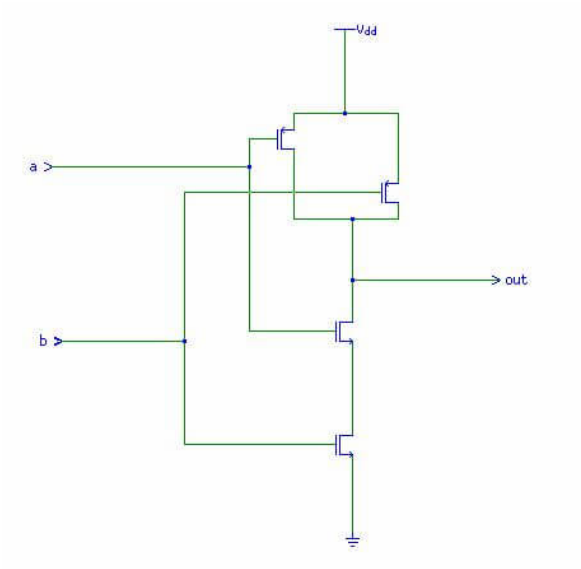
La porta logica elementare NOT, detta anche INVERTER, riceve in ingresso un segnale ad un bit e restituisce il valore di questo segnale in forma negata.



in	out
0	1
1	0

3.1.2 NAND

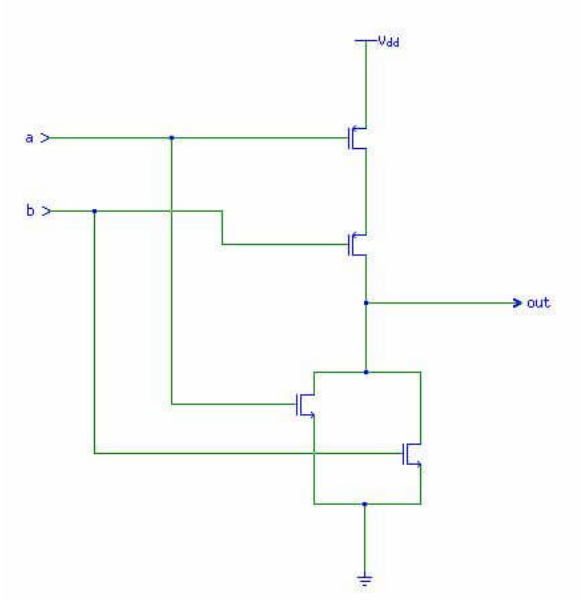
La porta logica elementare NAND riceve in ingresso due segnali a un bit e restituisce in uscita un segnale ad un bit. Il valore restituito sarà falso solo quando tutti gli ingressi sono veri; in tutti gli altri casi, l'output sarà vero.



a	b	out
0	0	1
0	1	1
1	0	1
1	1	0

3.1.3 NOR

La porta logica elementare NOR riceve in ingresso due segnali a un bit e restituisce in uscita un segnale ad un bit. Il valore restituito sarà vero solo quando tutti gli ingressi sono falsi; in tutti gli altri casi, l'output sarà falso.

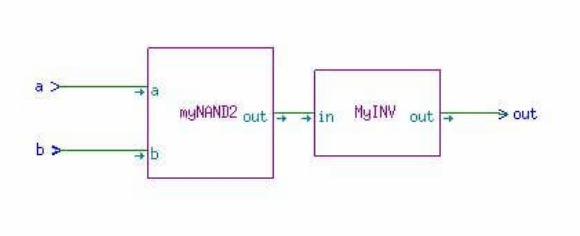


a	b	out
0	0	1
0	1	0
1	0	0
1	1	0

3.2 Porte logiche composte

3.2.1 AND

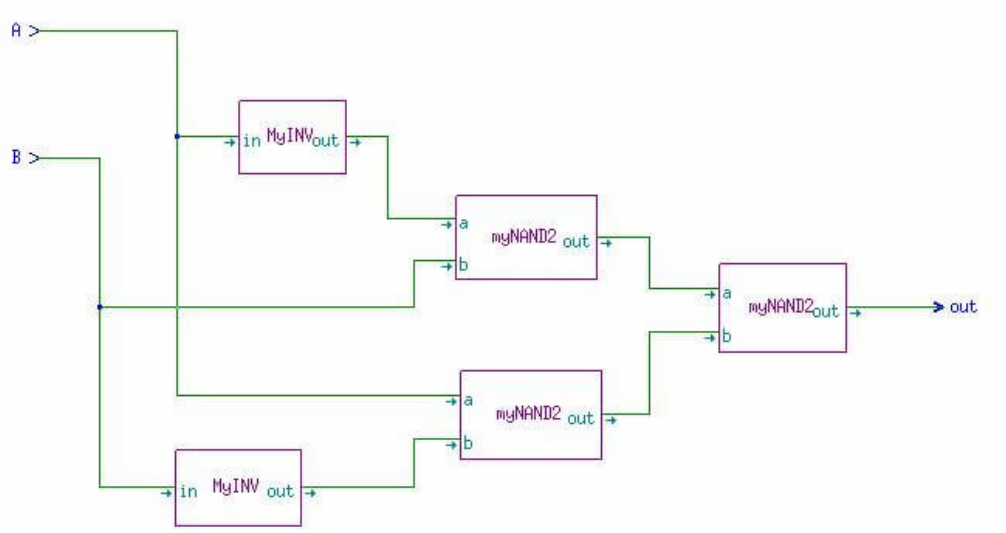
La porta logica AND riceve in ingresso due segnali a un bit e restituisce in uscita un segnale ad un bit. Il valore restituito sarà vero solo quando tutti gli ingressi sono veri; in tutti gli altri casi, l'output sarà falso.



a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

3.2.2 EXOR

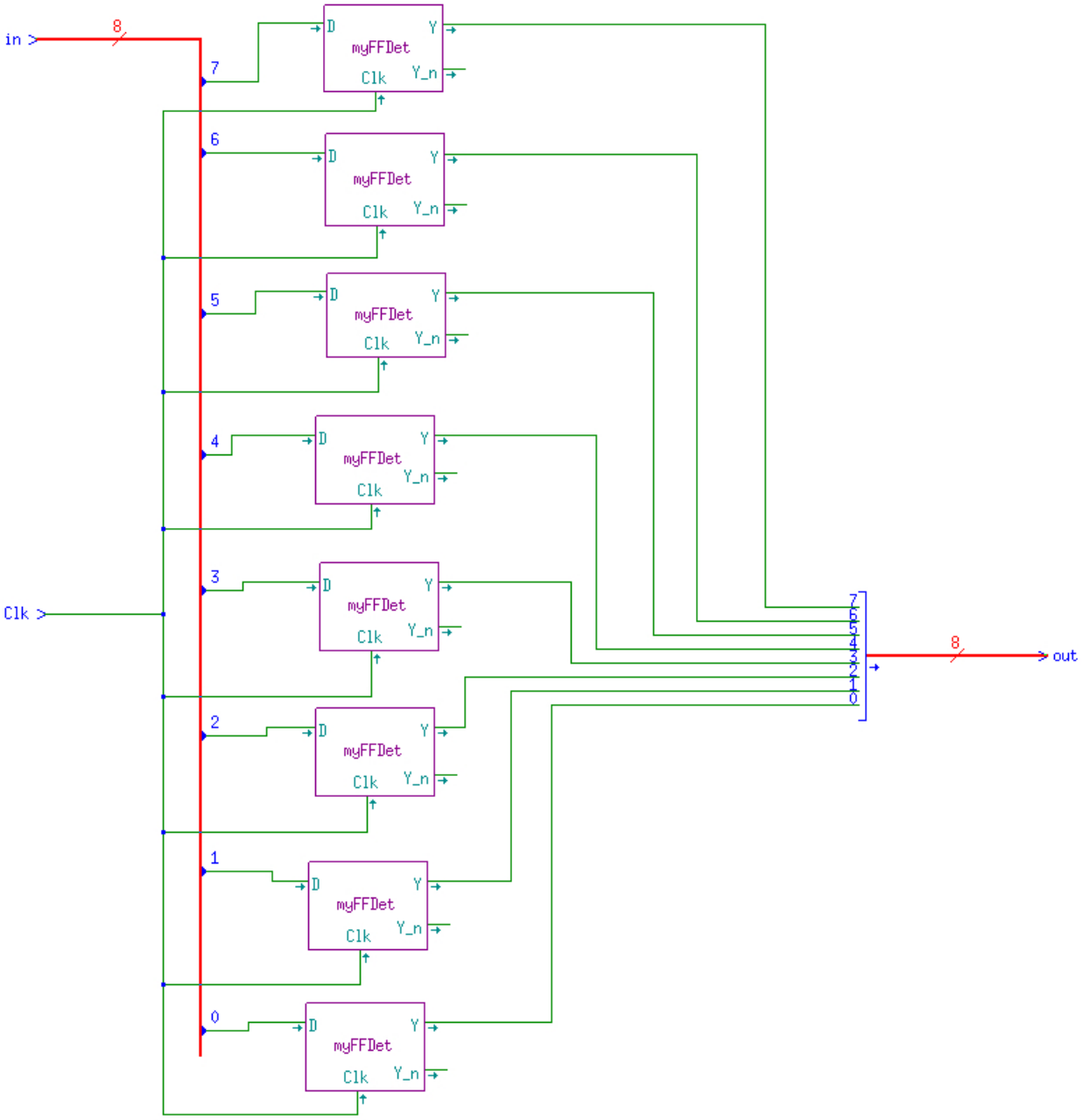
La porta logica EXOR, detta anche "OR esclusivo", riceve in ingresso due segnali a un bit e restituisce in uscita un segnale ad un bit. Il valore restituito sarà vero solo quando i due ingressi sono diversi tra loro; in tutti gli altri casi, l'output sarà falso.



A	B	out
0	0	0
0	1	1
1	0	1
1	1	0

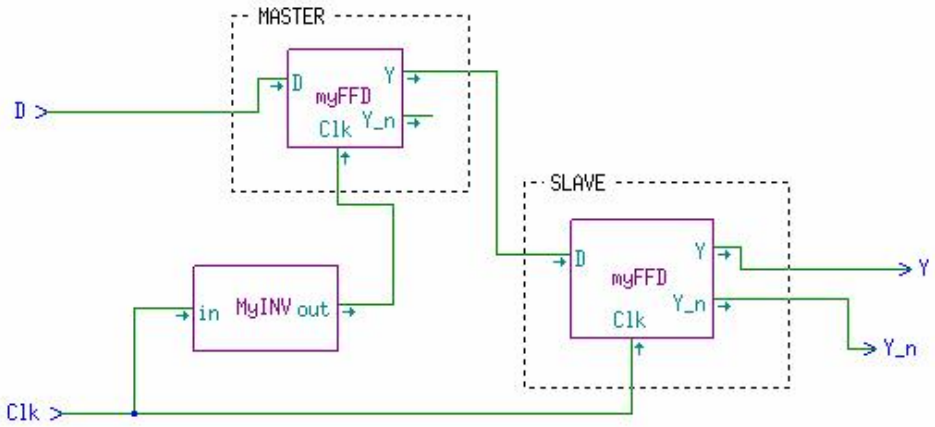
3.3 Registri

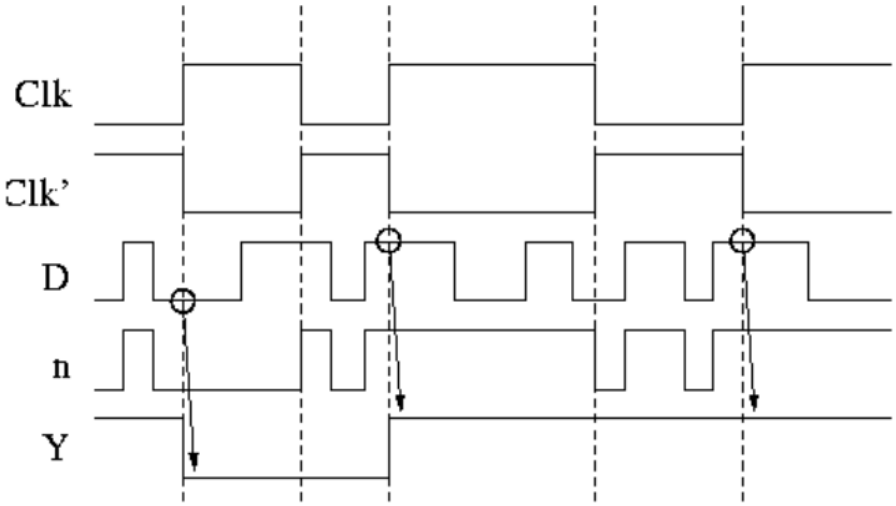
I registri permettono di campionare i dati in ingresso e di memorizzarli temporaneamente per ogni ciclo di clock. Nel circuito descritto, i registri impiegati sono ad 8 bit. Il registro realizzato è una batteria di Flip Flop Edge Triggered. Esso garantisce il campionamento di una parola di n bit e l'aggiornamento dei dati all'arrivo di ogni fronte di salita del clock.



3.3.1 Flip Flop D Edge Triggered

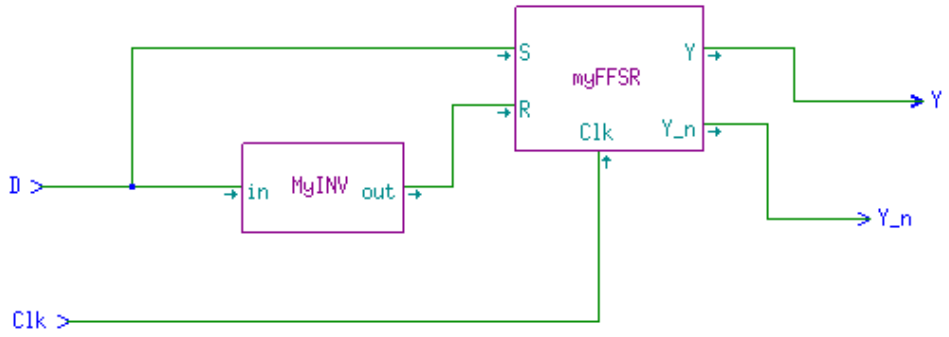
I flip-flop D edge-triggered sono circuiti che catturano il dato in ingresso in corrispondenza di un fronte specifico del segnale di clock, nel nostro progetto sul fronte di salita. Viene costruito utilizzando i Flip Flop D Level Sensitive.





3.3.2 Flip Flop D Level Sensitive

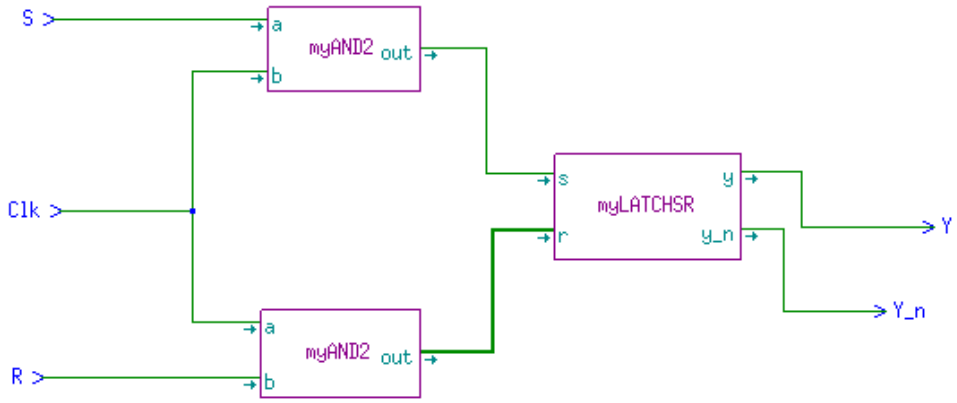
I flip-flop D level-sensitive sono circuiti che modificano il loro stato in risposta al livello continuo del segnale di clock, anziché ai suoi fronti. Questa caratteristica permette al flip-flop di reagire mentre il clock si mantiene in uno stato alto (o basso, a seconda della configurazione). Viene costruito utilizzando il Flip Flop SR.



Clk	D	S	R	s	r	actions
0	0	0	1	0	0	HOLD
0	1	1	0	0	0	HOLD
1	0	0	1	0	1	RESET
1	1	1	0	1	0	SET

3.3.3 Flip Flop SR

I flip-flop SR (Set-Reset) sono circuiti che consentono di stabilizzare l'uscita a un livello alto o basso in risposta a specifici segnali di controllo. In assenza di segnali di attivazione, questi dispositivi mantengono il loro stato corrente. Viene costruito utilizzando il Latch SR.

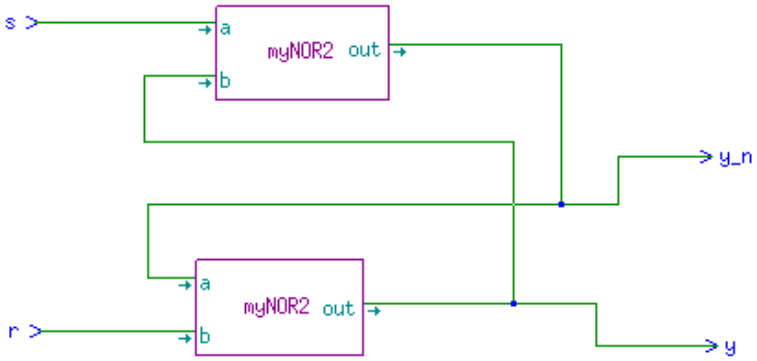


Clk	S	R	s	r	actions
0	0	0	0	0	HOLD
0	0	1	0	0	HOLD
0	1	0	0	0	HOLD
0	1	1	0	0	HOLD
1	0	0	0	0	HOLD
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	1	1	NOT ALLOWED

3.3.4 Latch SR

Il Latch SR (Set-Reset) è un circuito che stabilizza un'uscita in uno dei due stati possibili in risposta ai segnali di controllo. In condizioni di assenza di input attivi, il latch mantiene il suo

stato corrente.



y_n	s	r	y	y_n	actions
0	0	0	1	0	HOLD
0	0	1	0	1	RESET
0	1	0	1	0	SET
0	1	1	0	0	NOT ALLOWED
1	0	0	0	1	HOLD
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	0	0	NOT ALLOWED

3.4 Multiplexer

Il multiplexer è un dispositivo che seleziona uno tra diversi segnali di input per trasmetterlo all'uscita, basandosi su segnali di controllo. Nell'implementazione del nostro circuito, abbiamo usato dei multiplixer a due vie ad 8 bit.

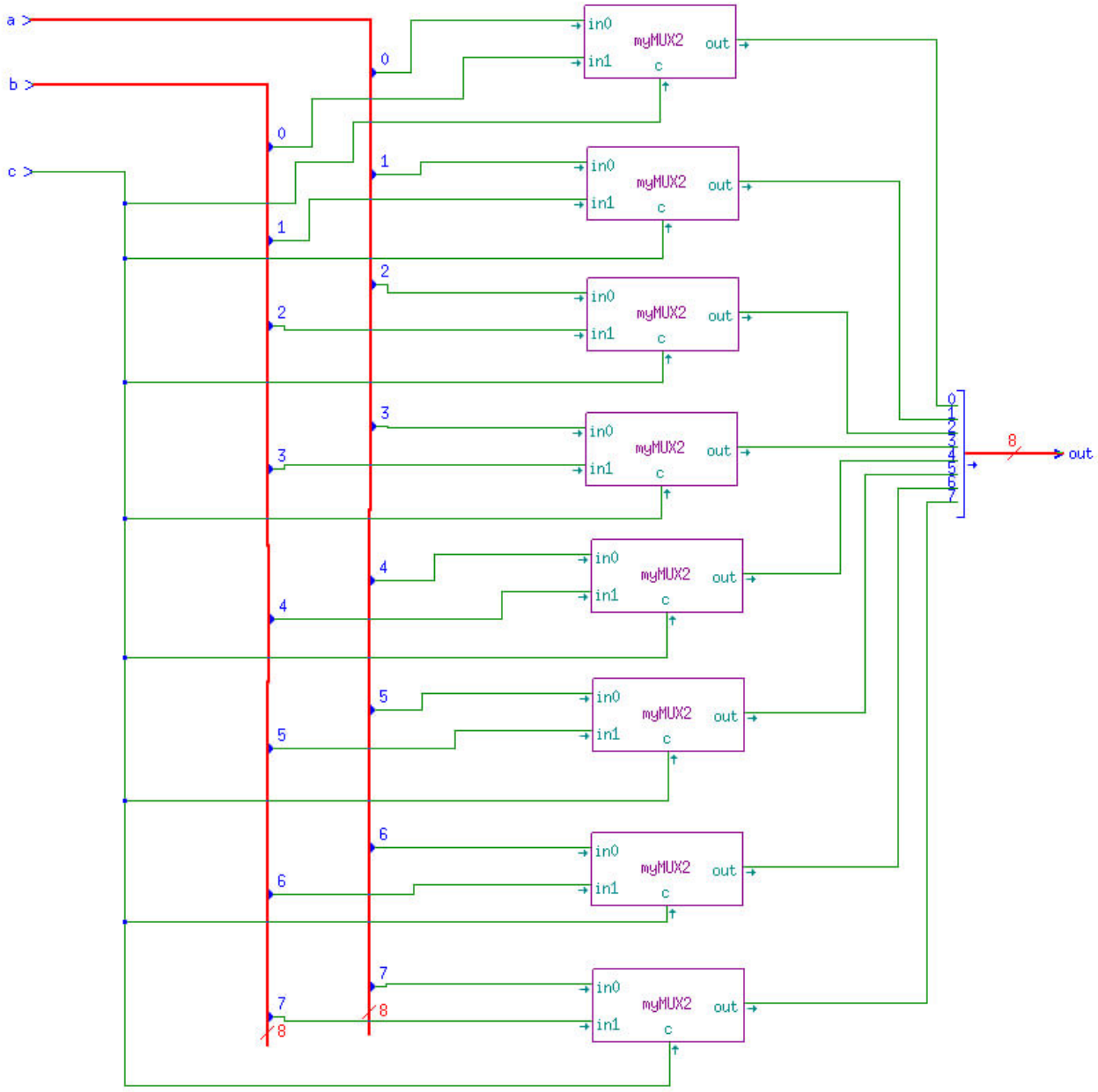


Figure 1: Multiplexer a 2 vie ad 8 bit

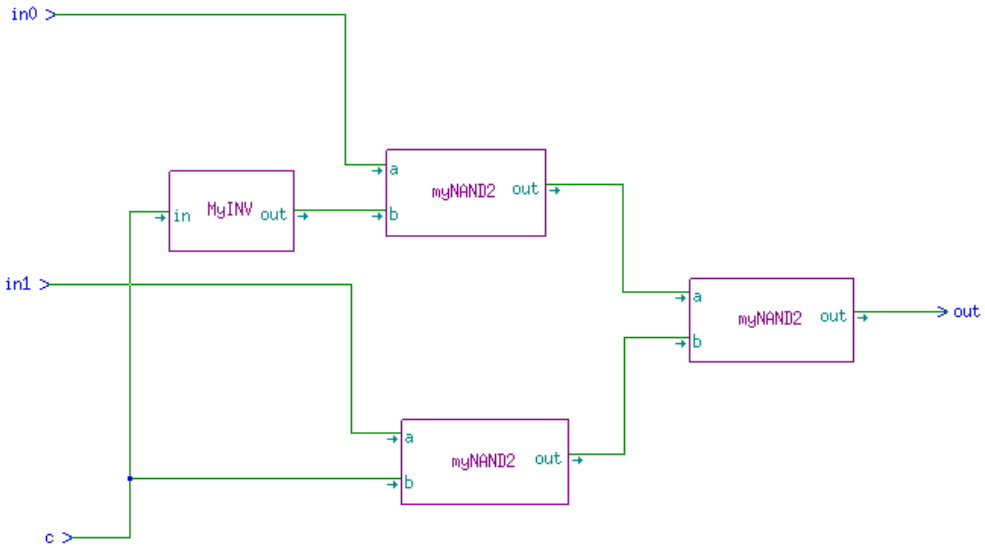


Figure 2: Multiplexer a 2 vie ad 1 bit

3.5 Demultiplexer

Il demultiplexer (DEMUX) è un dispositivo utilizzato per inviare un segnale di ingresso verso una tra diverse uscite possibili, selezionata dai segnali di controllo. Nell'implementazione del nostro circuito, abbiamo usato dei multiplixer a due vie ad 8 bit.

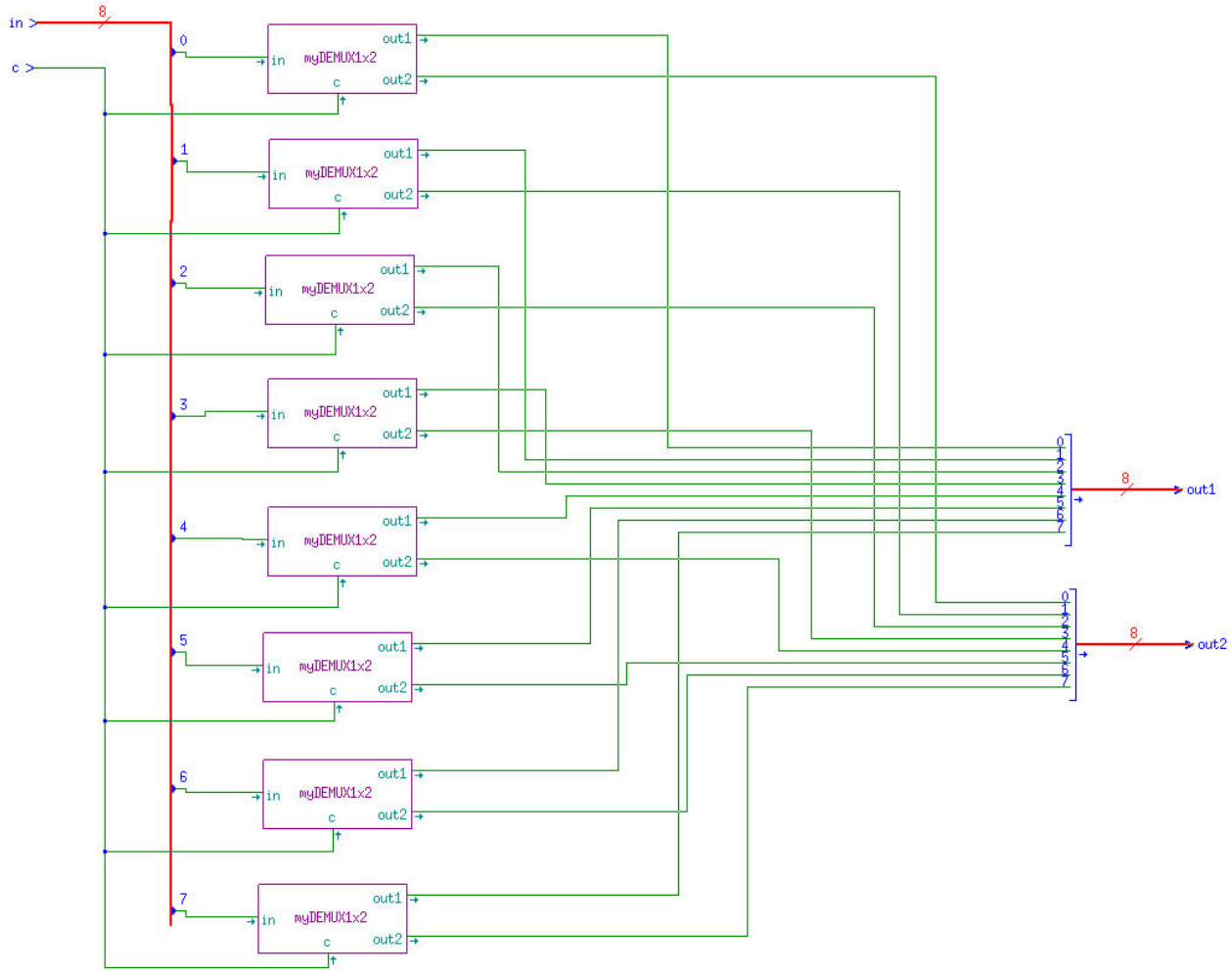


Figure 3: Demultiplexer a 2 vie ad 8 bit

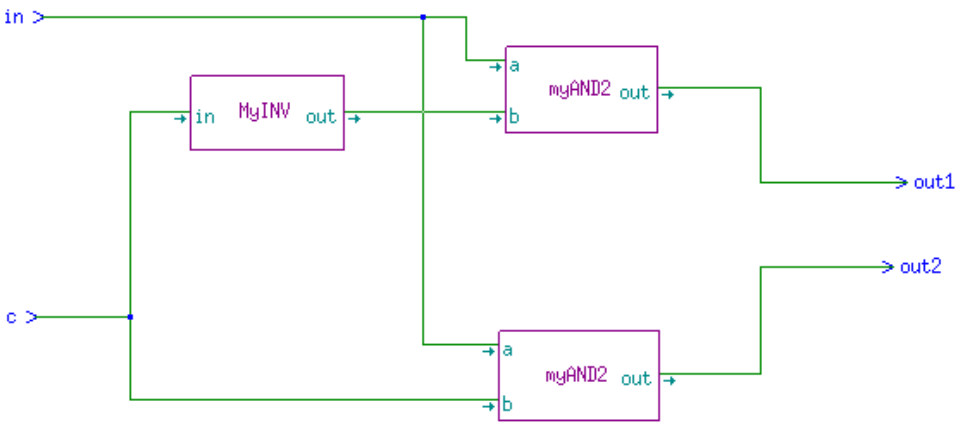


Figure 4: Demultiplexer a 2 vie ad 1 bit

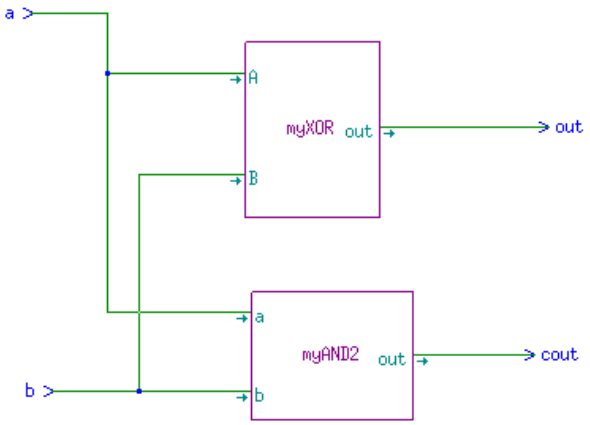
3.6 Macro funzionali

In relazione alla specifica funzionale, le macro necessarie sono:

- Un addizionatore a 8 bit in modalità Ripple Carry Adder
- Un moltiplicatore a 8 bit

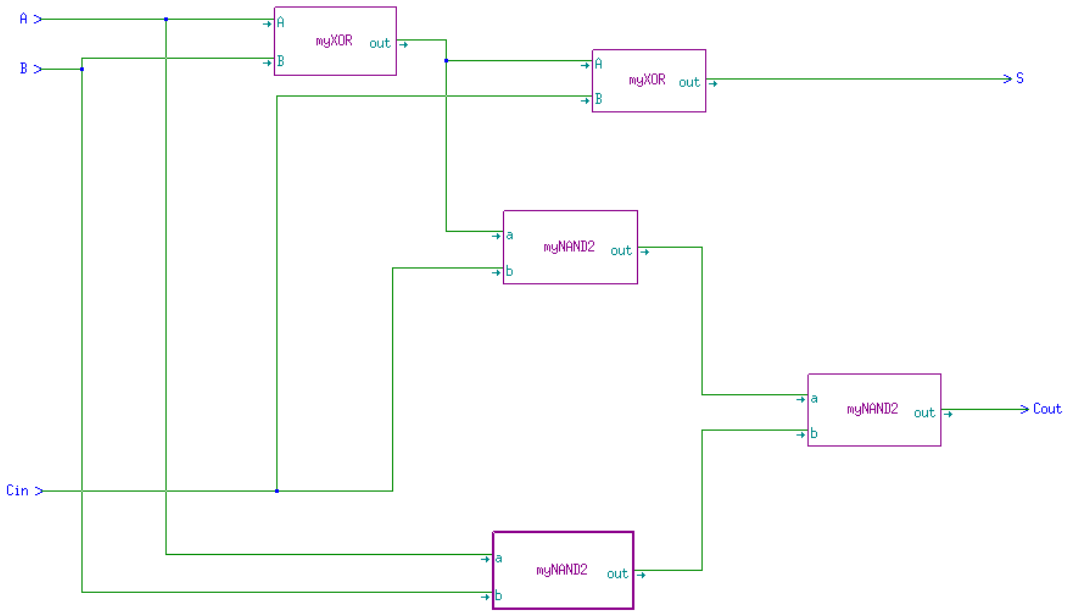
3.6.1 Half adder

L'Half Adder è un circuito che esegue la somma di due bit singoli, producendo due output: la somma e il riporto.



3.6.2 Full Adder

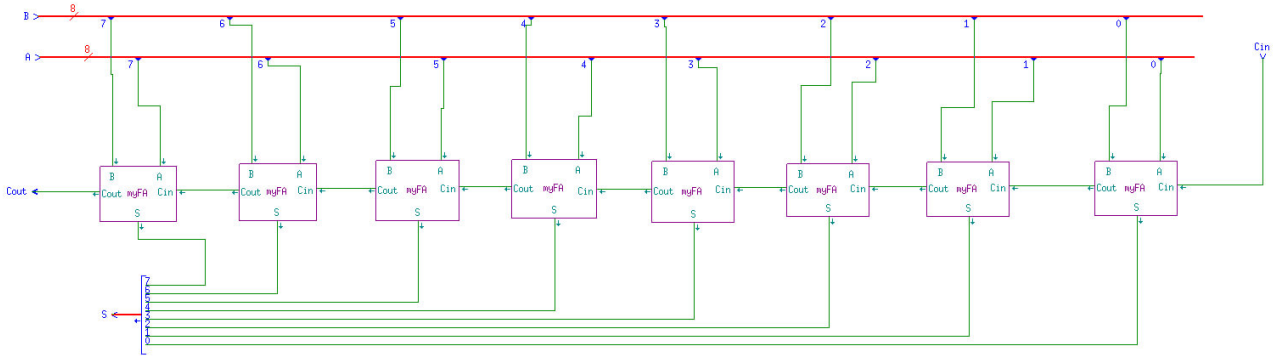
Il Full Adder è un circuito che somma tre bit di ingresso, due cifre binarie più un riporto in ingresso, producendo un risultato di somma e un riporto in uscita.



3.6.3 Addizionatore

L'addizionatore ripple carry è una configurazione nell'aritmetica dei circuiti digitali, utilizzato per sommare due numeri binari. Composto da una serie di full adders collegati in cascata, ogni adder elabora un bit dei numeri sommati e trasferisce un eventuale riporto al bit successivo.

Questo metodo presenta limitazioni nella velocità di elaborazione a causa della propagazione del riporto attraverso ogni stadio del circuito.
Nonostante tali limiti, rimane una soluzione pratica poiché i requisiti di velocità non sono stringenti. Inoltre, la sua struttura modulare facilita la scalabilità e l'integrazione.



3.6.4 Moltiplicatore

Nell'implementazione del moltiplicatore si vuole replicare il procedimento della moltiplicazione in colonna:

1. i due numeri binari vengono disposti uno sopra l'altro, allineando accuratamente le loro cifre meno significative.
2. Ogni cifra del moltiplicatore è moltiplicata sequenzialmente contro il moltiplicando, generando una serie di prodotti parziali.
3. Ciascun prodotto parziale è posizionato progressivamente una colonna a sinistra rispetto al prodotto immediatamente precedente, corrispondente al suo spostamento a sinistra nel moltiplicatore.
4. Infine, l'addizione di questi prodotti parziali fornisce il risultato finale della moltiplicazione.

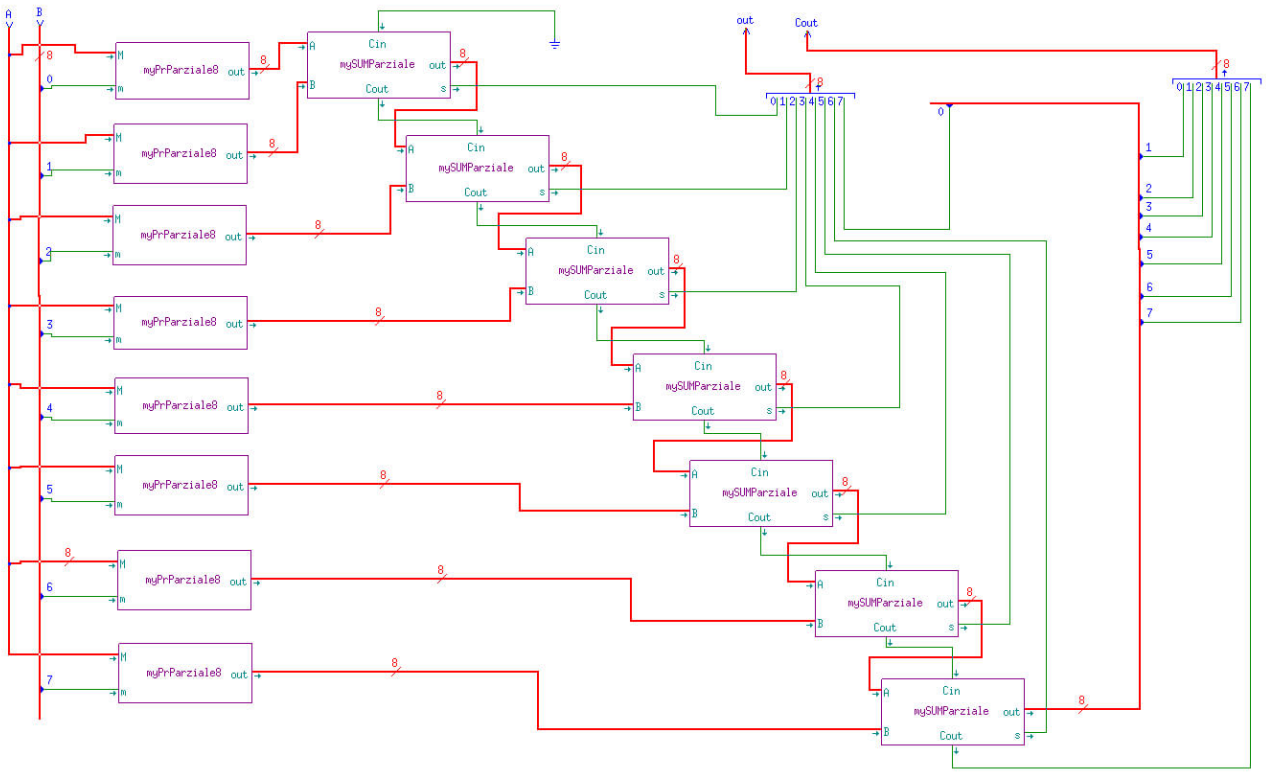


Figure 5: Un moltiplicatore a 8 bit

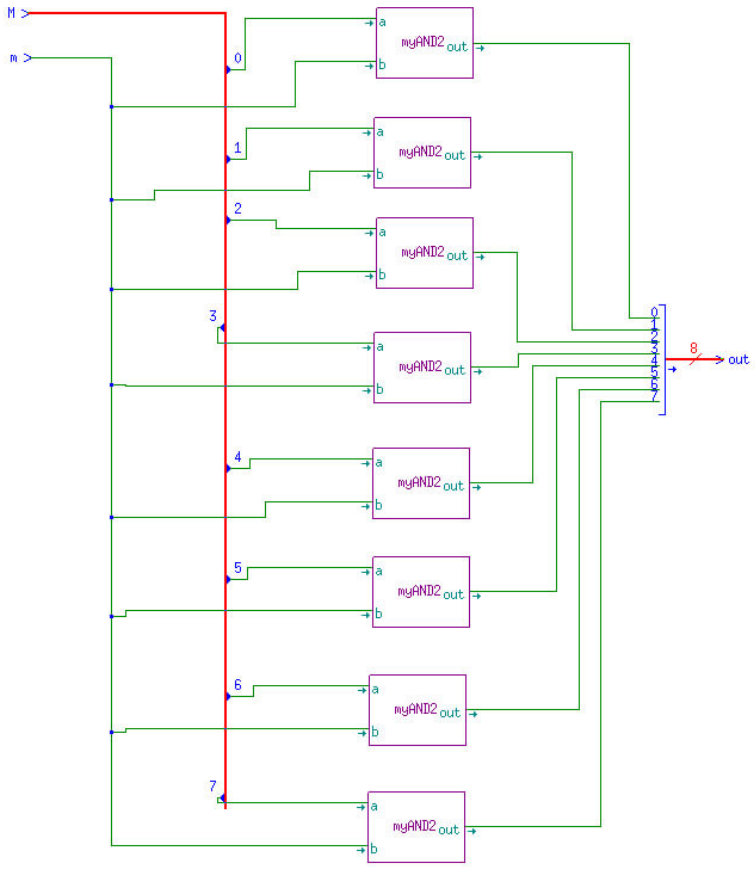


Figure 6: Componente per il calcolo dei prodotti parziali

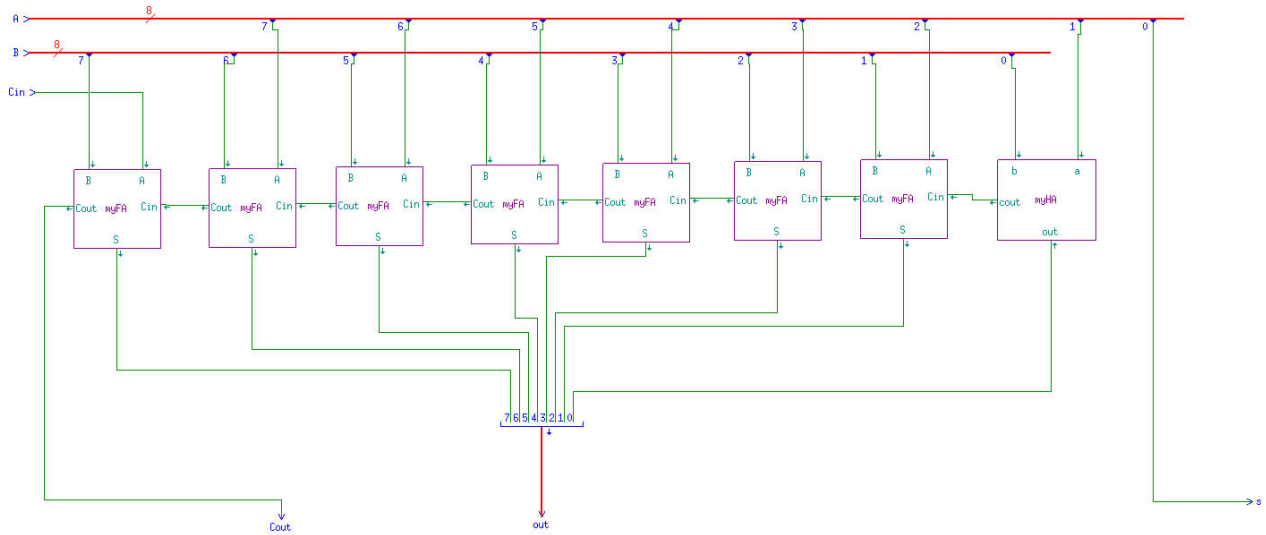
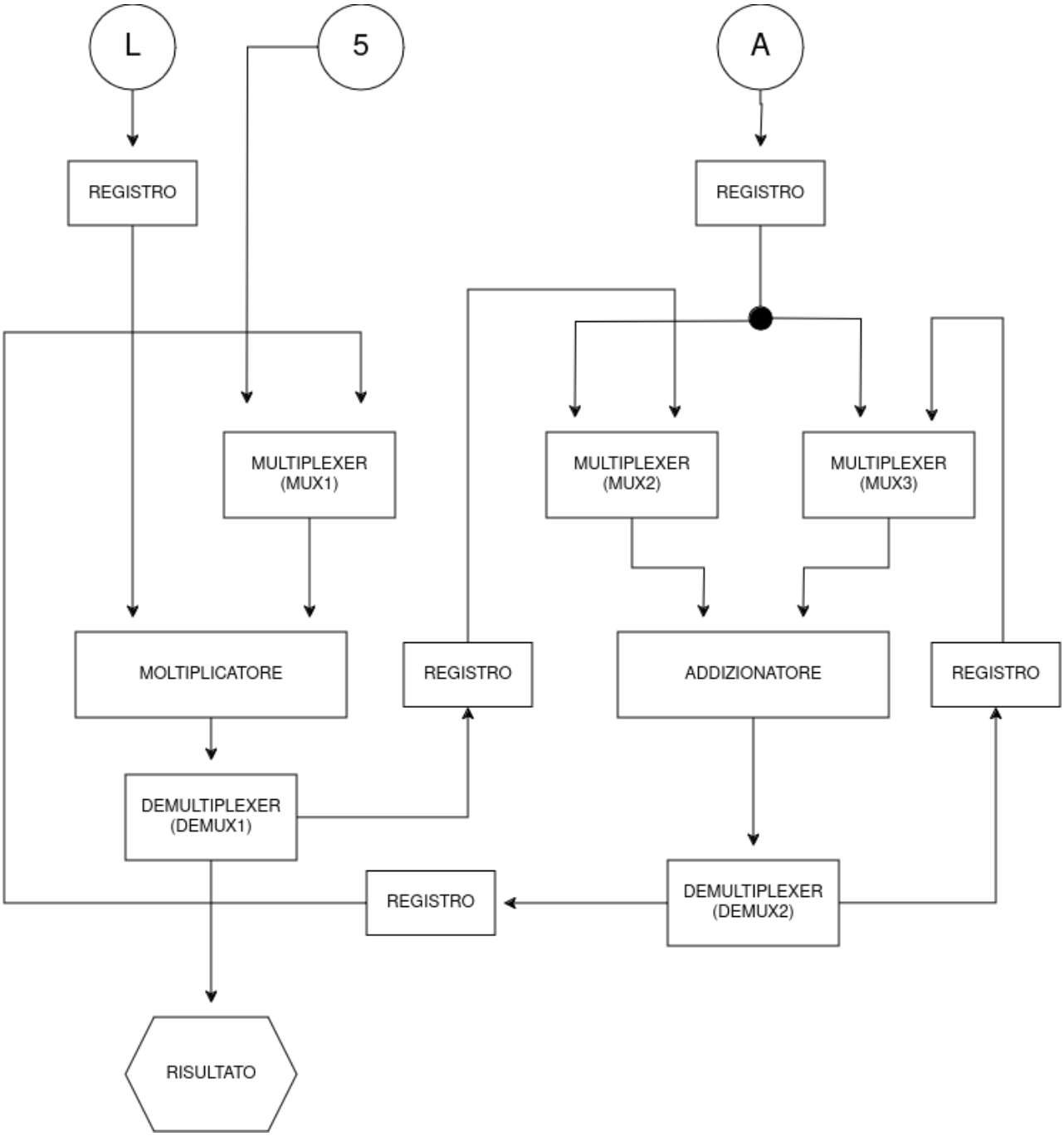


Figure 7: Componente per la somma dei prodotti parziali

4 Data Path


L
5
A
REGISTRO
REGISTRO
MULTIPLEXER (MUX1)
MULTIPLEXER (MUX2)
MULTIPLEXER (MUX3)
MOLTIPLICATORE
REGISTRO
ADDIZIONATORE
REGISTRO
DEMULTIPLEXER (DEMUX1)
REGISTRO
DEMULTIPLEXER (DEMUX2)
RISULTATO

5 Control Unit

La Control Unit è un componente per la gestione dei flussi di dati e le operazioni del sistema:

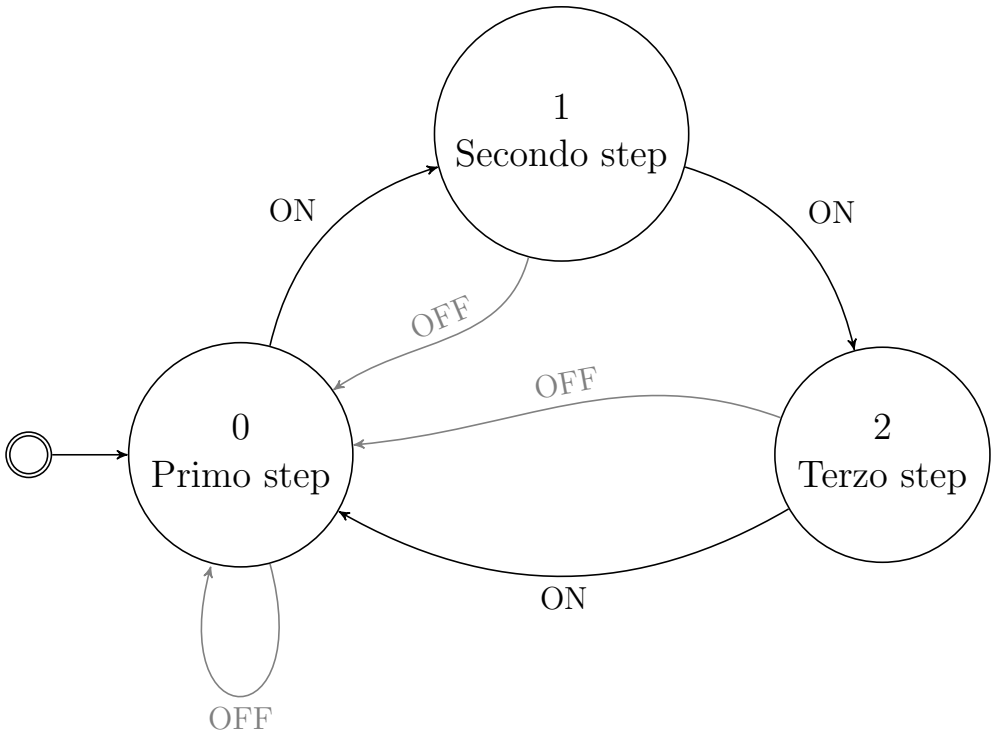
- Comanderà l'azione dei multiplexer e dei demultiplexer;
- Aiuterà a dare un riscontro visuale per informare l'utente in quale step del ciclo di transizione si trova;
- Riceverà il segnale di clock in ingresso. In questo contesto viene fornito manualmente dall'utente attraverso l'utilizzo di uno switch;
- Riceverà il segnale di abilitazione in ingresso. In questo contesto viene fornito manualmente dall'utente attraverso l'utilizzo di uno switch.

5.1 Diagramma degli stati

In questa sezione, presentiamo una macchina a stati finiti (FSM) sincronizzata con il ciclo di clock che modella un sistema sequenziale. L'FSM è costituita da tre stati distinti, rappresentati da 0, 1 e 2, i quali corrispondono rispettivamente allo scorrere del ciclo di clock. Come rappresentato nella sezione 2.1.

La logica di transizione è governata dal seguente comportamento:

- 0 (Iniziale/Reset): Il sistema si inizializza o si resetta allo stato 0, che è considerato lo stato di partenza o di reset. In assenza di input o in presenza dal segnale di enable ad 'OFF', la FSM ritorna o permane in questo stato, indipendentemente dallo stato corrente.
- Se il segnale di enable è 'ON', la FSM segue una sequenza ciclica di transizioni attraverso gli stati. Partendo da 0, il sistema transita allo stato 1 al successivo fronte di salita del clock, quindi a 2 al ciclo di clock seguente, e infine ritorna a 0, completando un ciclo di transizione.



Legenda:
Cerchio con due linee di testo:

Freccia indica transizione:

Stato	Enable	Uscita	Stato futuro
0	OFF	Primo step	0
0	ON	Primo step	1
1	OFF	Secondo step	0
1	ON	Secondo step	2
2	OFF	Terzo step	0
2	ON	Terzo step	0

5.2 Codifica degli stati e segnali di controllo

In questa sezione, progettiamo i segnali di controllo per i multiplexer e i demultiplexer.
Utilizziamo il valore 0 per indicare la selezione dell'ingresso di sinistra per i multiplexer e per indicare l'uscita di sinistra per i demultiplexer. Di conseguenza, utilizziamo il valore 1 per indicare la selezione dell'ingresso di destra per i multiplexer e per indicare l'uscita di destra per i demultiplexer.

Prendendo come riferimento lo schema del Data Path nella sezione 4:

- il segnale A governerà il multiplexer MUX1;
- il segnale B governerà i multiplexer MUX2 e MUX3;
- il segnale C governerà il demultiplexer DEMUX1;
- il segnale D governerà il demultiplexer DEMUX2.

Si è optato per regolare il funzionamento degli instradatori affinché il risultato completo sia reso disponibile esclusivamente al termine del ciclo di transizioni, evitando così la presentazione di risultati intermedi.
L'implementazione di questa strategia garantisce l'integrità e la coerenza dei dati all'uscita del circuito, assicurando che solo informazioni complete siano presentate come output finale.

Stato presente	Stato Futuro	A (mux1)	B (mux2, mux3)	C (demux1)	D (demux2)
0	1	0	0	1	1
1	2	0	1	1	0
2	0	1	1	0	0

Procediamo con la codifica degli stati. Aggiungiamo una ulteriore configurazione per poter sviluppare correttamente la sintesi logica di questi blocchi. Consideriamo i risultati di questa configurazione come condizioni di indifferenza e le impostiamo in modo tale da ottimizzare il circuito.

Stato presente	Stato Futuro	A (mux1)	B (mux2, mux3)	C (demux1)	D (demux2)
00	01	0	0	1	1
01	10	0	1	1	0
10	00	1	1	0	0
11	10	1	1	0	0

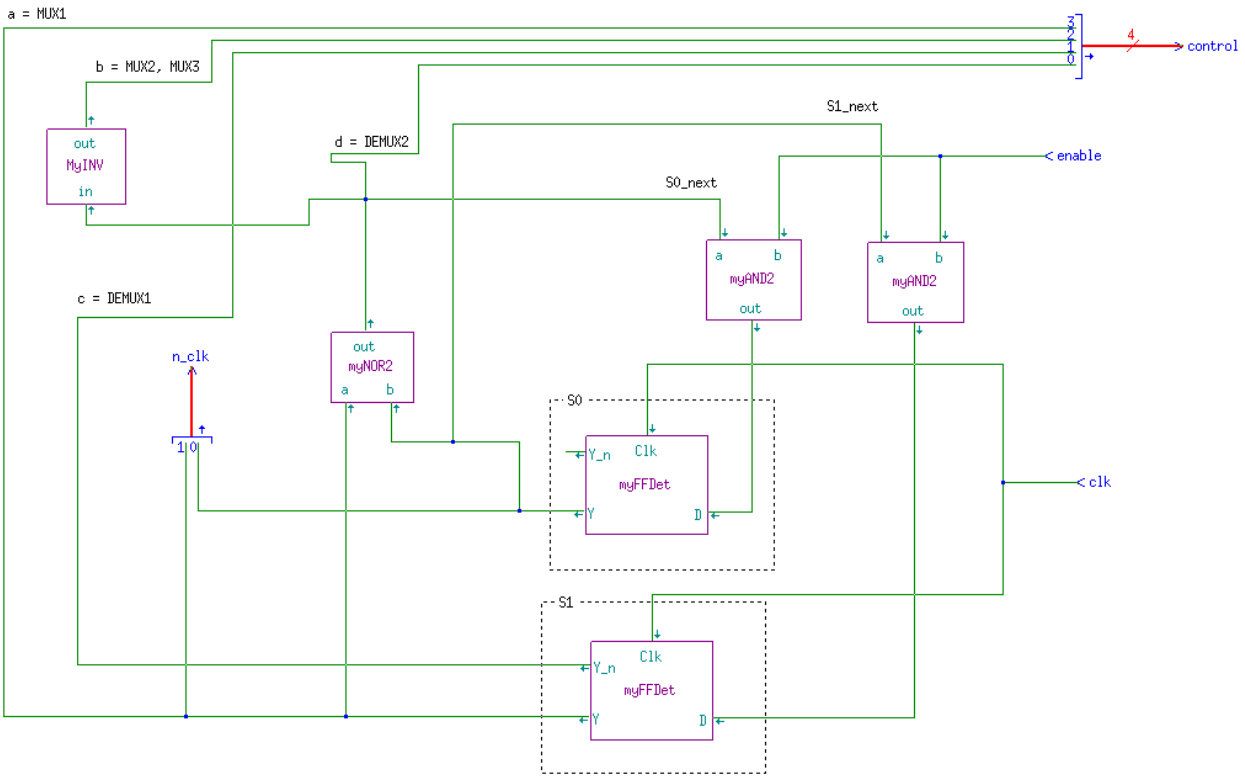
5.3 Tabelle della verità della funzione d'uscita e di stato futuro

Stato Presente		Stato Futuro		Control			
S1	S0	$S1_{\text{next}}$	$S0_{\text{next}}$	A	B	C	D
0	0	0	1	0	0	1	1
0	1	1	0	0	1	1	0
1	0	0	0	1	1	0	0
1	1	1	0	1	1	0	0

$S1_{\text{next}} = S0$
$S0_{\text{next}} = (S1 + S0)'$
$A = S1$
$B = S1 + S0$
$C = S1'$
$D = (S1 + S0)'$

si noti che:
$B' = D = S0_{\text{next}}$
$C' = A$

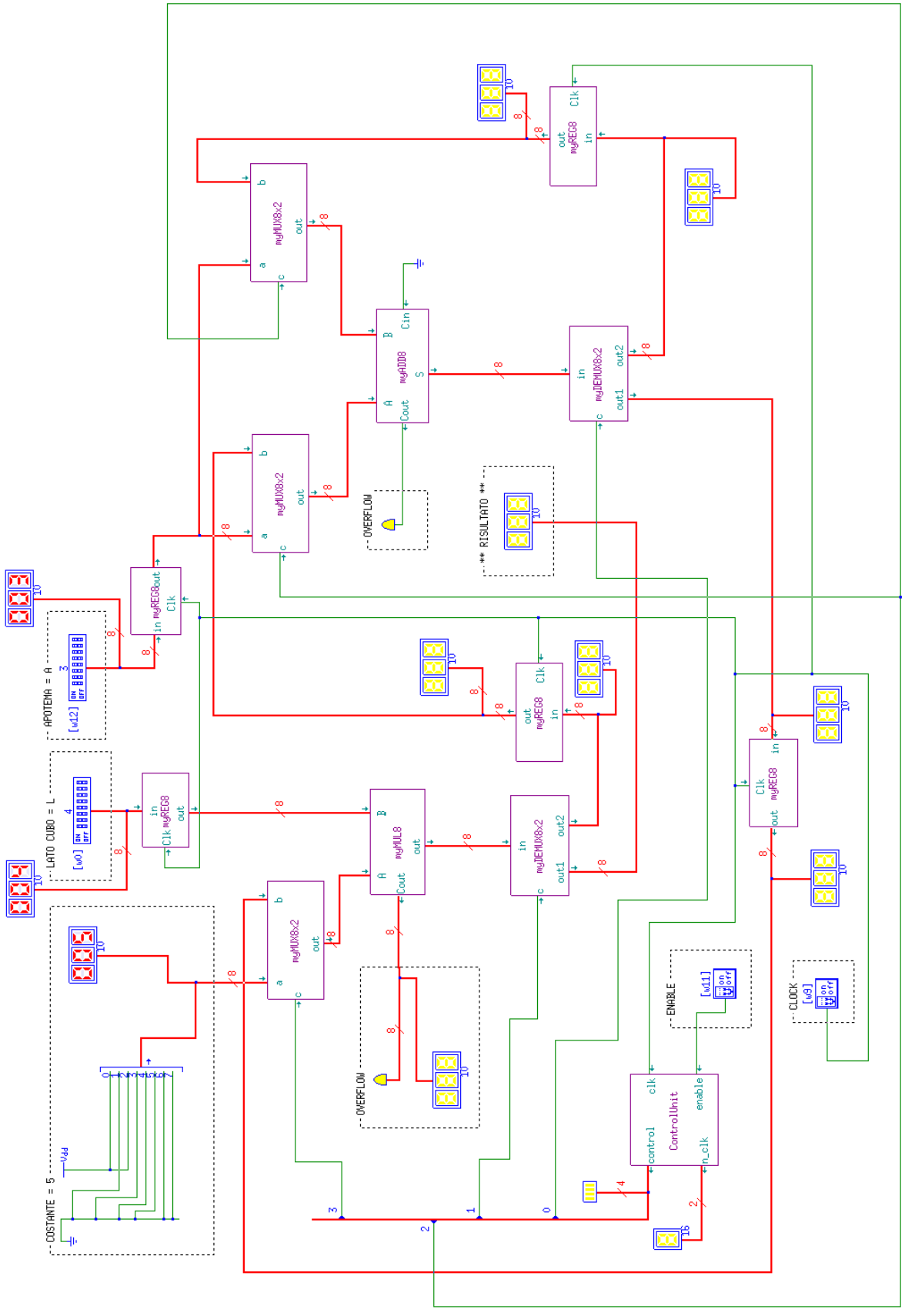
5.4 Implementazione della Control Unit



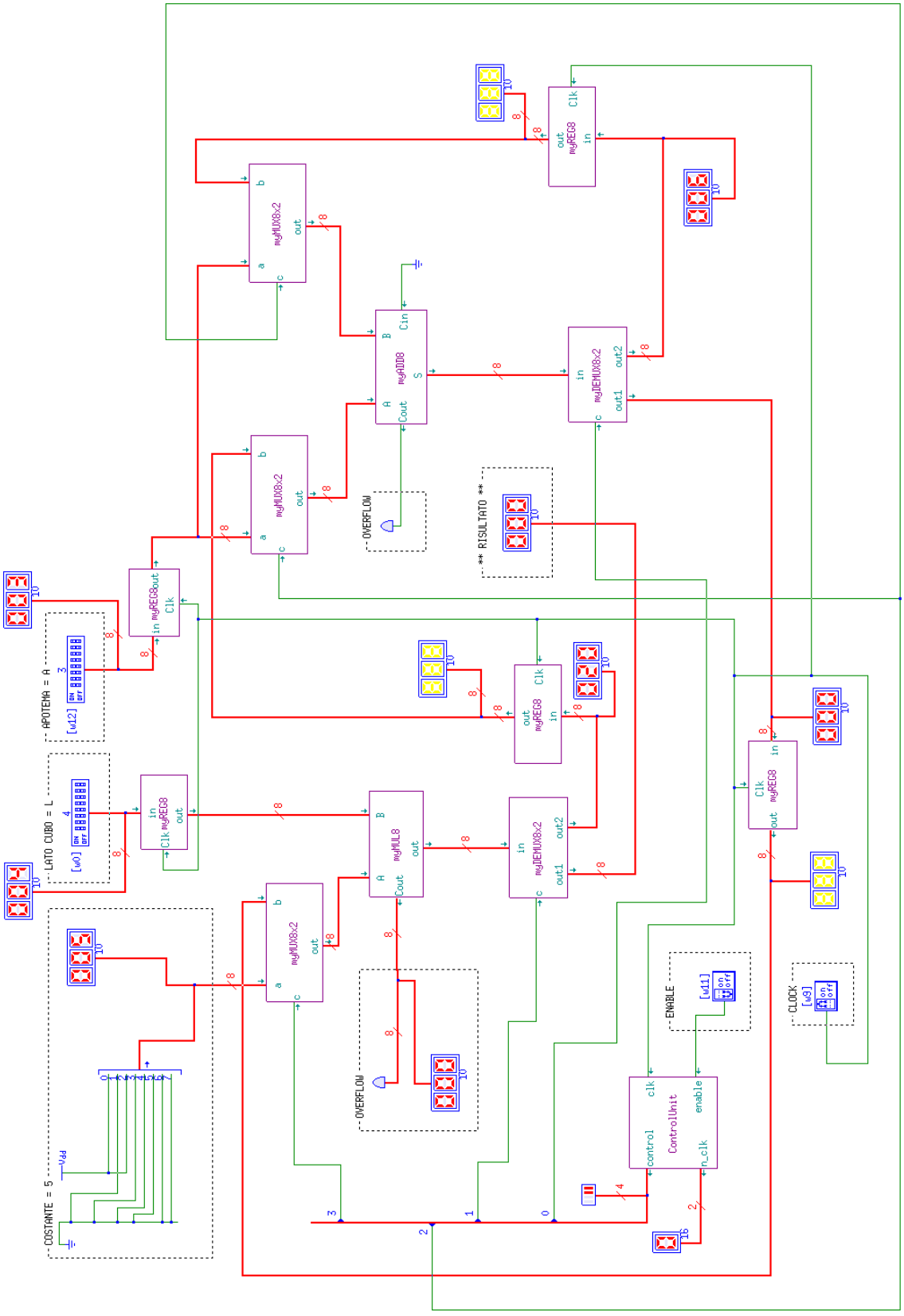
6 Simulazione e analisi del progetto

6.1 Verifica funzionale

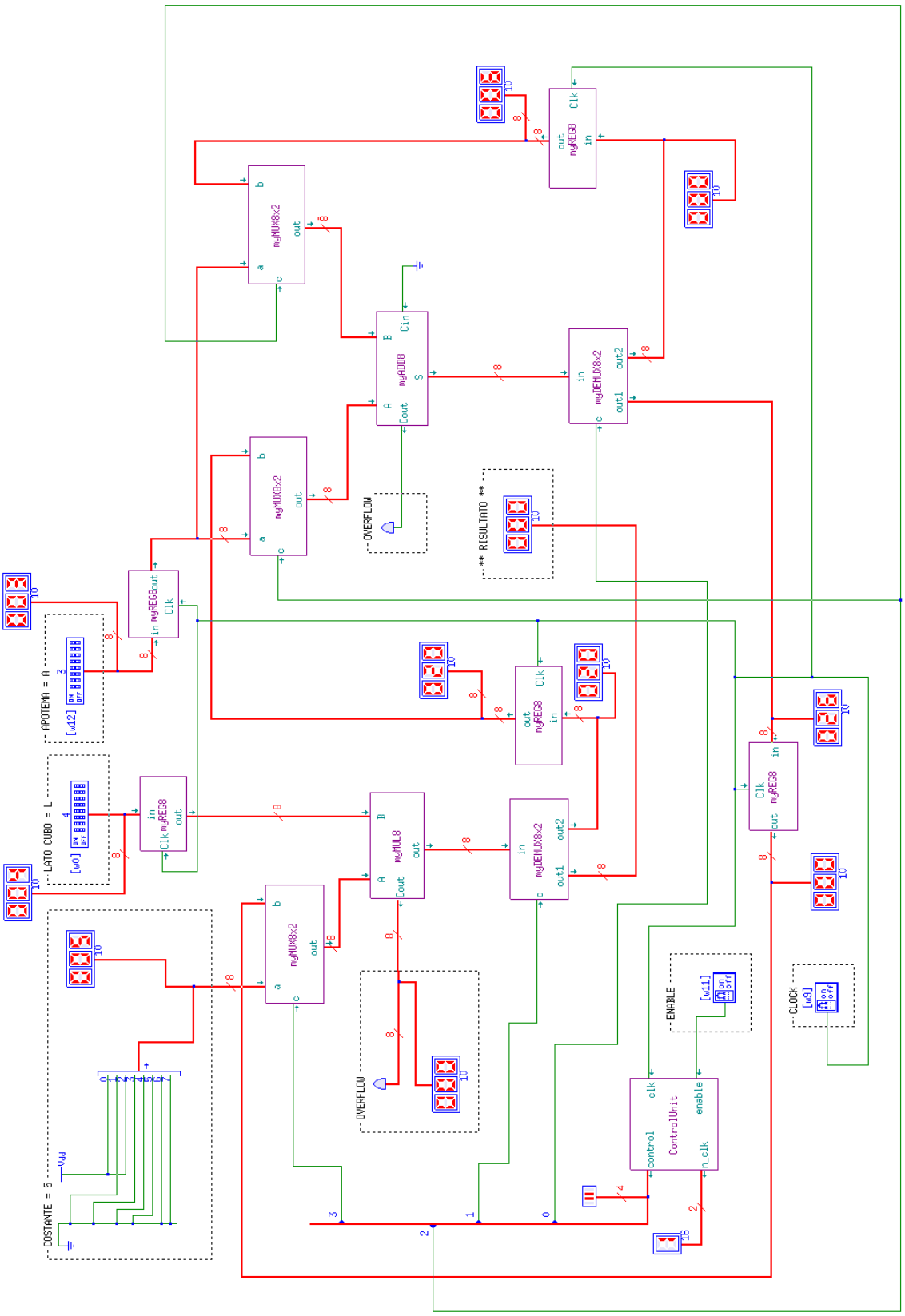
6.1.1 Avvio della simulazione



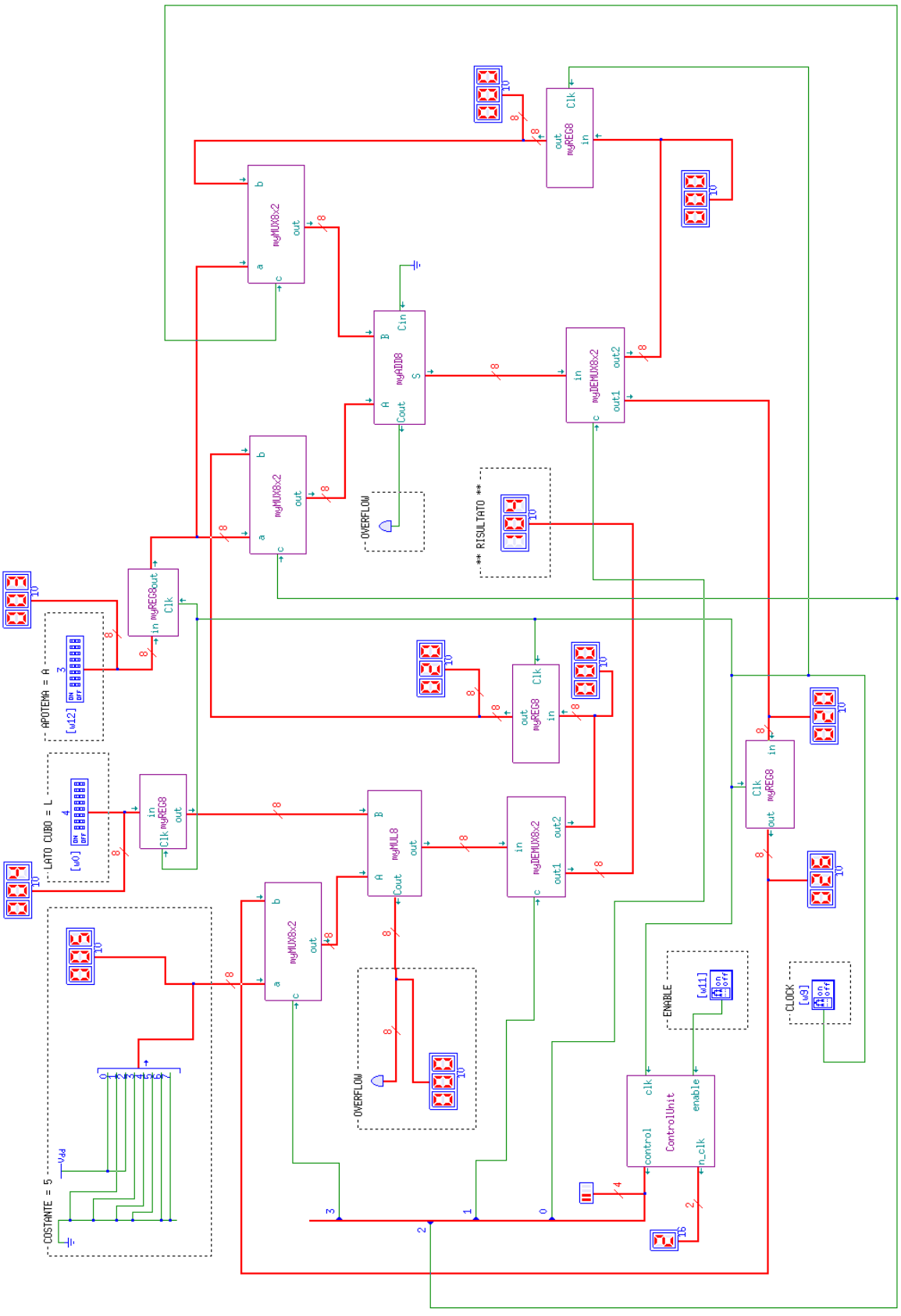
6.1.2 Avvio primo clock con inizializzazione



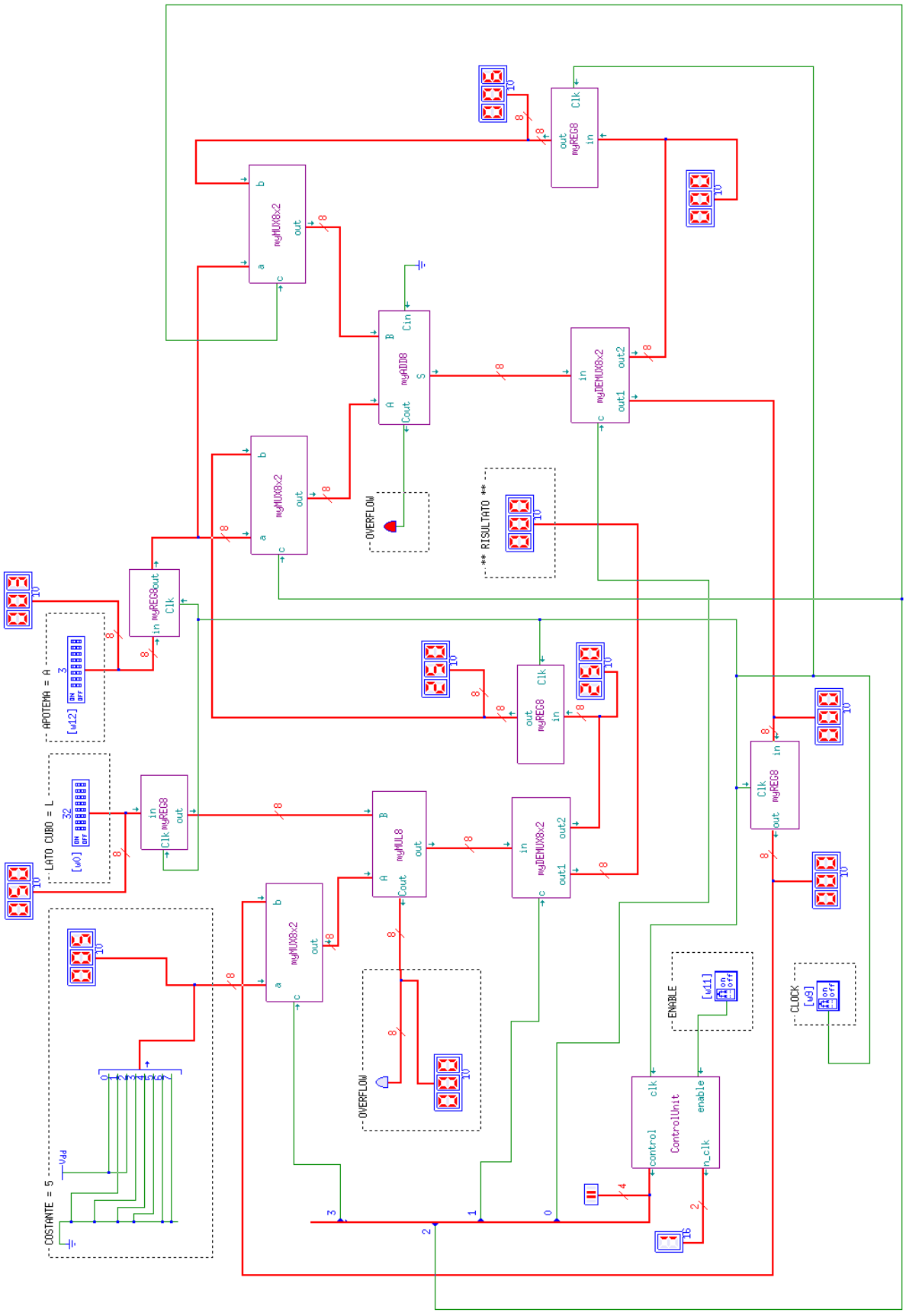
6.1.3 Avvio secondo ciclo di clock con abilitata la transizione degli stati



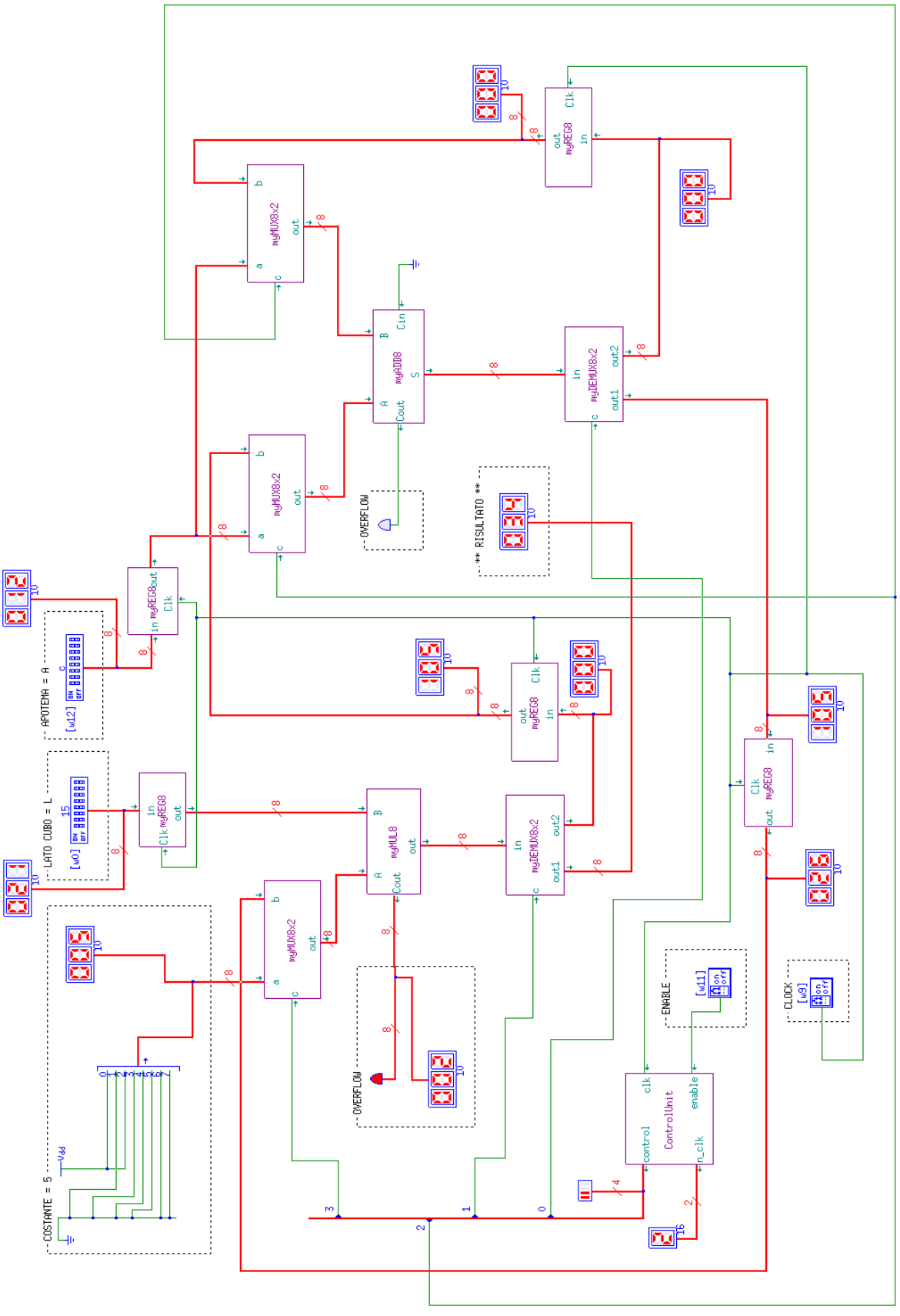
6.1.4 Avvio terzo ciclo di clock con abilitata la transizione degli stati



6.1.5 Evidenza Overflow Addizionatore

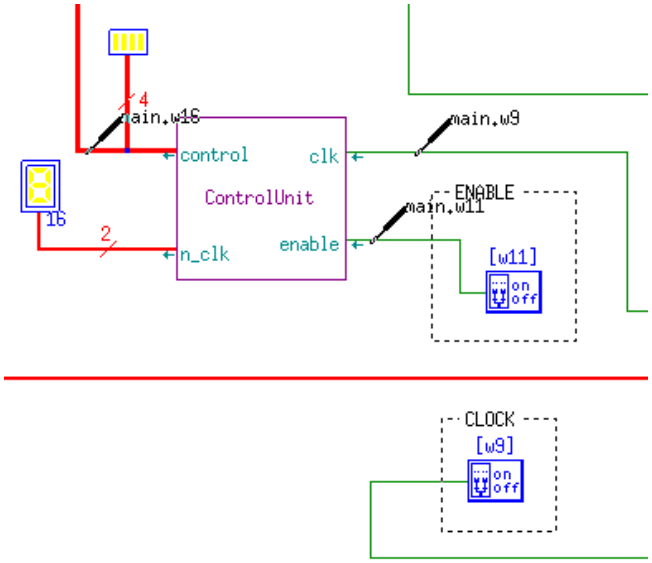


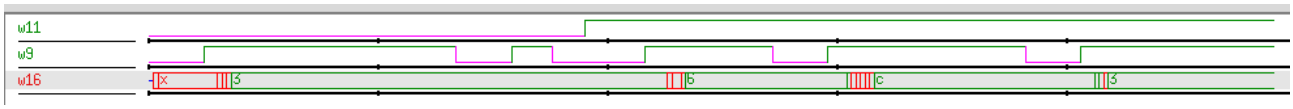
6.1.6 Evidenza Overflow Moltiplicatore



6.2 Simulazioni della CU e delle Macro funzionali

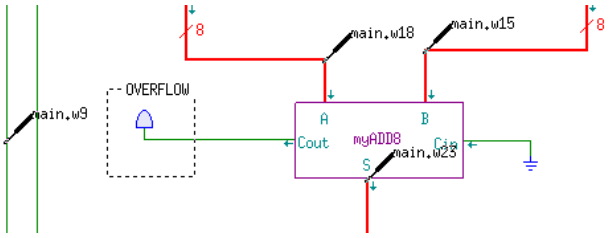
6.2.1 Control Unit

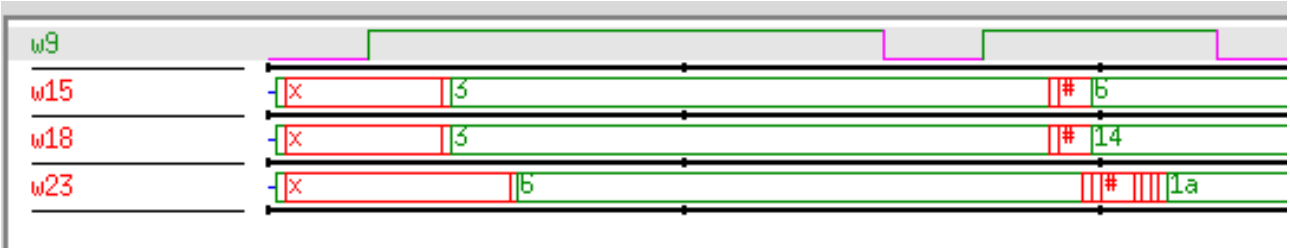




- w11 è il probe presente sull'enable;
- w9 è il probe presente sul clock;
- w16 è il probe presente sull'uscita della Control Unit;
- X indica indeterminato;
- i numeri in w16 sono rappresentati in formato esadecimale:
  - 3 indica 0011;
  - 6 indica 0110;
  - C indica 1100.

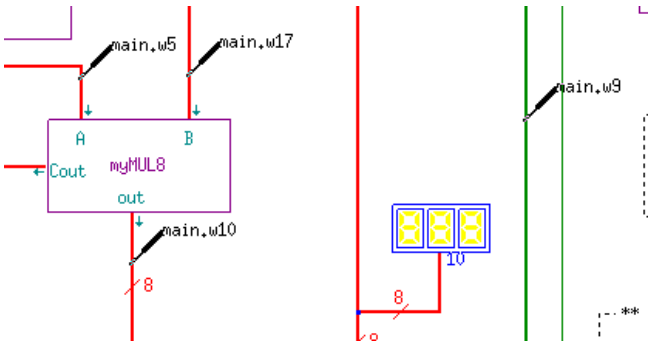
6.2.2 Addizionatore

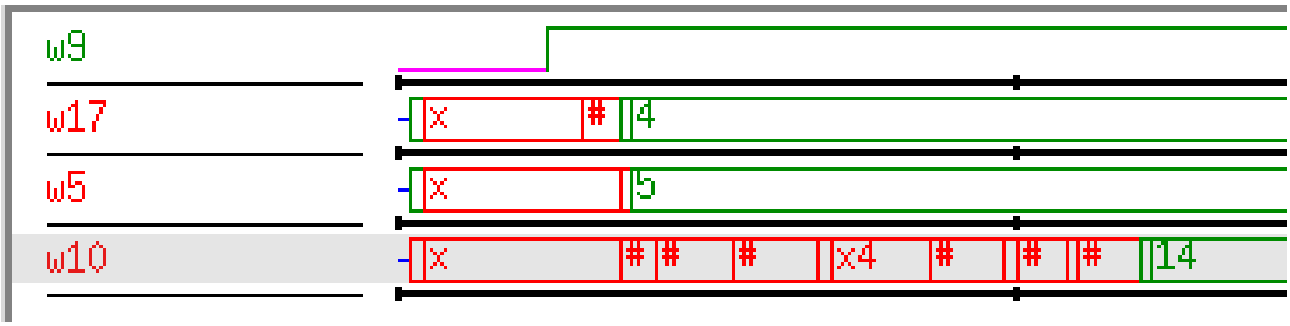




- w9 è il probe presente sul clock;
- w15 e w18 sono gli ingressi dell'addizionatore;
- w23 è il probe presente sull'uscita dell'addizionatore;
- X indica indeterminato;
- i numeri in w23 sono rappresentati in formato esadecimale:
  - 3 indica 00000011;
  - 6 indica 00000110;
  - 14 indica 00010100 quindi 20;
  - 1A indica 00011010 quindi 26;

6.2.3 Moltiplicatore





- w9 è il probe presente sul clock;
- w17 e w5 sono gli ingressi del moltiplicatore;
- w10 è il probe presente sull'uscita del moltiplicatore;
- X indica indeterminato;

- i numeri in w10 sono rappresentati in formato esadecimale:
  - 4 indica 00000100;
  - 5 indica 00000101;
  - 14 indica 00010100 quindi 20;

6.3 Stima di complessità circuitale e prestazioni

6.3.1 Stima dell'area, del tempo di contaminazione e di propagazione

- **Area (A)** : La valutazione dell'area si baserà sul conteggio degli ingressi delle porte logiche. Contare gli ingressi fornisce un livello di dettaglio adeguato senza la necessità di scendere a un livello di granularità che coinvolga i singoli transistor;
- **Tempo di contaminazione (TC)** : Il tempo di contaminazione esprime l'intervallo di tempo unitario necessario affinché, a fronte di una nuova configurazione di dati in ingresso, le uscite vengano influenzate. Dal punto di vista circuitale, si valuterà il percorso più corto del segnale contando il numero delle porte logiche da esso attraversate. Verrà calcolato con il modello di ritardo unitario[3];
- **Tempo di propagazione (TP)** : Il tempo di propagazione, esprime l'intervallo di tempo necessario affinché, a fronte di una nuova configurazione di dati in ingresso, i corrispondenti dati in uscita siano stabili. Dal punto di vista circuitale, si valuterà il percorso più lungo del segnale contando il numero delle porte logiche da esso attraversate. Verrà calcolato con il modello di ritardo unitario.

Porta	A	TP	TC
NOT	1	1	1
NAND	2	1	1
NOR	2	1	1
AND	3	2	2
EXOR	8	3	2

[3]Il modello a ritardo unitario considera come unità la singola porta attraversata dal segnale.

Porta	A	TP	TC
FDls [4]	11 (2 × NOR) + (2 × AND) + (1 × NOT)	4	3
FFDet [5]	23 (2 × FFDls) + (1 × NOT)	9	7
REG 8bit	184 8 × FFDet	9	7
MUX 1x2	7 (3 × NAND) + (1 × NOT)	3	2
MUX 8x2	56 8 × (MUX 1x2)	3	2
DEMUX 1x2	7 (2 × AND) + (1 × NOT)	3	2
DEMUX 8x2	56 8 × (DEMUX 1x2)	3	2
HA	11 (1 × EXOR) + (1 × AND)	3	2
FA	22 (2 × EXOR) + (3 × NAND)	6 2 × EXOR	2 2 × NAND
Addizionatore	176 8 × FA => $O(n)$	48 8 × FA => $O(n)$	2 1 × FA => $O(1)$
Prodotto Parziale	24 8 × AND	2	2
Somma Parziali	165 (7 × FA) + (1 × HA)	45 (7 × FA) + (1 × HA)	2 1 × HA [considerando l'uscita out]
MUL 8 bit	1347 [8 × (Prodotto Parziale)]+ [7 × (Somma Parziali)] ⇒ $O(n^2)$	317 [1 × (Prodotto Parziale)]+ [7 × (Somma Parziali)] ⇒ $O(n)$	2 1 × AND ⇒ $O(1)$ [considerando l'uscita out]
CU	55 (2 × FFDet) + (2 × AND)+ (1 × NOR) + (1 × NOT)	13 (1 × AND) + (1 × FFDet)+ (1 × NOR) + (1 × NOT)	7 1 × FFDet

[5]Flip Flop D Level Sensitive
[6]Flip Flop D Edge Triggered

6.3.2 Complessità e prestazioni

Tempo di clock	$T_{clk} > T_p(\text{MUX}) + T_p(\text{MUL}) + T_p(\text{DEMUX})$
Area	$A_{MUL} + A_{ADDER} + (5 \times A_{REG}) +$ $(3 \times A_{MUX}) + (2 \times A_{DEMUX}) + A_{CU}$
Tempo di propagazione	$3 \times T_p(\text{REG}) + 2 \times T_p(\text{MUL}) + 3 \times T_p(\text{DEMUX}) +$ $2 \times T_p(\text{MUX}) + 1 \times T_p(\text{ADD})$ REG → MUL → DEMUX → REG → MUX → ADD → DEMUX → REG → MUX → MUL → DEMUX
Tempo di contaminazione	$T_c(\text{REG}) + T_c(\text{MUL}) + T_c(\text{DEMUX})$ REG → MUL → DEMUX