Quadrato di un prodotto

Progetto d’esame di Reti Logiche A.A. 2021/2022

Stefano Zizzi , 312793

# Specifica

### Scopo del progetto

Il progetto prevede la realizzazione di un circuito su TKGate 2.1, capace di elaborare l’elevamento al quadrato di un prodotto di due fattori, codificati da un massimo di 4 bit.

### Specifica funzionale

Sia 𝐴 che 𝐵 potranno ottenere in input solamente valori la cui codifica sarà espressa tramite un massimo di 4 bit ciascuno. Il risultato di questa espressione aritmetica, tuttavia, sarà rappresentato con l’utilizzo di 16 bit. Ciò è dovuto al fatto che il valore massimo che identifica 𝐴 e 𝐵 è 65536.

Assumendo che 𝐴 e 𝐵 abbiano entrambi valore 16, il valore massimo che si potrà ottenere è . Per vedere quanti bit servono per rappresentare in binario il numero calcoliamo che restituisce 16.

### Specifica parametrica

Tenendo conto che i valori d’ingresso possono essere rappresentati da un massimo di 4 bit, il circuito prenderà in considerazione due valori di input in codifica esadecimale, che saranno espressi con un massimo di 4 bit, escludendo i valori che supereranno questa discretizzazione.

Per ottenere il risultato finale sono necessari 3 cicli di clock, il necessario per elaborare tutte le moltiplicazioni.

# Impostazione del progetto a livello RT

### Data Flow Graph

Per realizzare questa specifica è stato scelto di implementare la condivisione di risorse come stile di progetto. Infatti, per elaborare le 2 moltiplicazioni, il circuito implementa solamente un moltiplicatore. La scelta di usare due moltiplicatori al posto di uno singolo è dovuta al fatto che per la realizzazione dei vari componenti è stato utilizzato l’incapsulamento, sfruttando componenti che verranno implementati in altre parti del circuito, tra cui il prodotto dei due numeri iniziali.

I cicli di clock necessari per elaborare l’intera espressione correttamente sono tre. La condivisione di risorse comporta un aumento dei cicli di clock, ma con il guadagno di utilizzare meno risorse e quindi di diminuire la complessità circuitale del nostro circuito.

Dato che il circuito utilizza la condivisione delle risorse, non è possibile sfruttare il pipelining.

### Risorse

Sono state implementate le seguenti risorse:

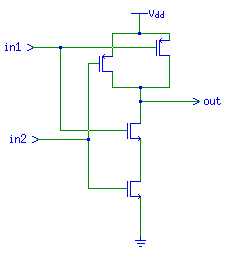
* **Registri**. L’implementazione di questo circuito necessita di tre registri:
  + Il primo ed il secondo registro, entrambi utilizzati per stabilizzare gli input forniti al primo moltiplicatore, sono a 4 bit come gli operandi.
  + Il terzo registro è da 16 bit ed il suo compito è prima quello di memorizzare il risultato di e poi di stabilizzare il risultato che rappresenterà il quadrato del prodotto.
* **Multiplexer**. L’implementazione di questo circuito necessita di un MUX con entrambi gli ingressi da 8 bit.
* **Macro aritmetiche**. Le macro aritmetiche implementate sono:
  + Un moltiplicatore 8x8 bit per calcolare sia e .
* **Full Adder (FA)**. Con implementazione da 2, 4, 8 bit.
* **Control Unit**. La Control Unit è stata implementata necessariamente per poter gestire i MUX in maniera automatizzata, in quanto ad ogni fronte di salita del clock i due multiplexer necessitano di un ingresso di controllo differente dal precedente.

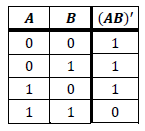
# Progetto delle risorse a livello gate

### Porte logiche elementari

Le porte logiche elementari della tecnologia FCMOS utilizzate per la realizzazione del progetto sono:

* NAND
* NOR
* NOT (Inverter)

**NAND**



### **NOR**

### 

### 

### 

### 

### 

### 

### 

### **NOT(Inverter)**

### 

### 

### 

### 

### 

### Inoltre, per la realizzazione del progetto, sono state utilizzate altre porte logiche derivanti dalla composizione di più porte logiche elementari della tecnologia FCMOS. Le porte in questione sono:

### AND

### EXOR

### 

### **AND**

### 

### 

### 

### 

### Come si può vedere in figura, l’AND è stato costruito con un NAND a due ingressi e con l’uscita negata, ottenuta collegando un Inverter. L’ AND è stato utilizzato per l’implementazione del moltiplicatore e dell’Half Adder.

**EXOR**

### 

### 

### 

### 

L’EXOR è stato costruito con due Inverter e tre NAND a due ingressi, come mostrato in figura. Questa porta logica è stata utilizzata per realizzare sia il Full Adder che l’Half Adder.

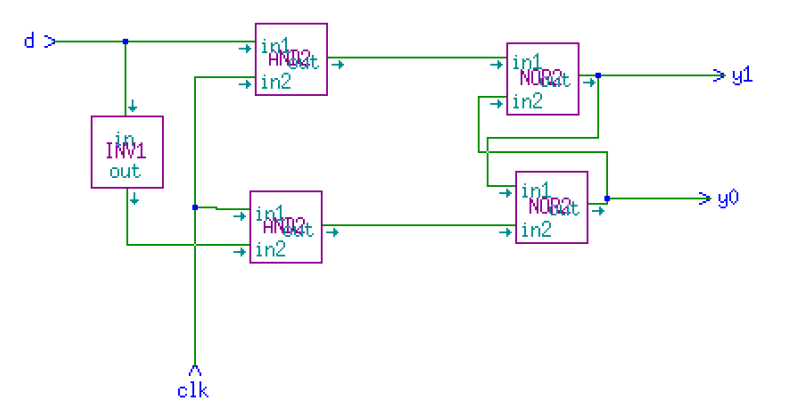
Di seguito sono elencati i vari tempi di contaminazione, di propagazione e l’area delle porte logiche elementari e composte sopra citate. Per la stima della complessità circuitale di una porta logica è stato scelto di valutare il numero di input di porte logiche, in quanto un input equivale a due transistor.

|  |  |  |  |
| --- | --- | --- | --- |
| **NAND** | 1 | 1 | 2 |
| **NOR** | 1 | 1 | 2 |
| **NOT(Inverter)** | 1 | 1 | 1 |
| **AND** | 2 | 2 | 3 |
| **EXOR** | 2 | 3 | 8 |

### Registri

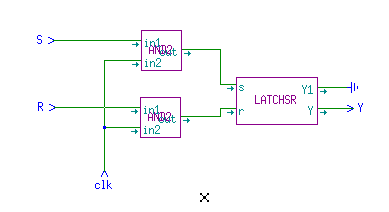
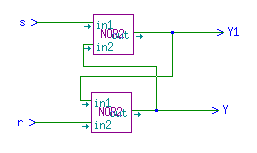
Il circuito del quadrato per elaborare il risultato correttamente stabilizzato, necessita di quattro registri. I registri sono stati utilizzati per fornire dei sincronismi, quindi per suddividere l’elaborazione in vari cicli di clock. Sono stati utilizzati diversi registri a seconda del numero di bit che dovevano ricevere in ingresso, per poi essere stabilizzati. Per questo motivo sono stati realizzati registri a 4 e a 16 bit. La struttura interna di questi ultimi componenti è pressoché identica, con la differenza che all’interno ci saranno Flip Flop Edge Triggered per bit da stabilizzare. I Flip Flop Edge Triggered, a loro volta, saranno costituiti da due Flip Flop D (level sensitive).

**FFD**



**Studio delle metriche:**

* Tempo di propagazione = Tp(Not) + Tp(And) + 2Tp(Nor) = 1 + 2 + 2 x 1 = 5;
* Tempo di contaminazione = Tp(And) + 2Tp(Nor) = 2 + 2 x 1 = 4;
* Area = A(Not) + 2A(And) + 2A(Nor) = 1 + 2 x 3 + 2 x 2 = 11.

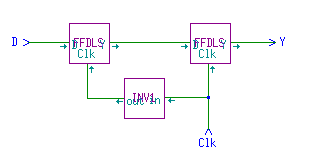
**FLIP FLOP SR LATCH SR**

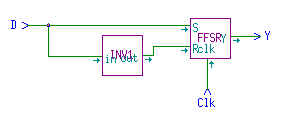
**Studio delle metriche: Studio delle metriche:**

Tp = 2 + 1= 3; Tp = 1;

Tc = 2 + 1 = 3; Tc = 1 ;

A= 3 + 3 + 4 = 4. A= 2 + 2 = 4.

**FLIP FLOP D LS FLIP FLOP D ET**



**Studio delle metriche: Studio delle metriche:**

Tp = 3 + 1= 4; Tp = 7 + 7 = 14;

Tc = 3; Tc = 3;

A= 4 + 1 = 5. A= 5 + 5 + 1 = 11.

**REGISTRO A 4 BIT**

### 

### 

### 

### 

### 

### 

### 

### **Studio delle metriche:**

### Tempo di propagazione = Tp(Flip Flop Edge Triggered) = 10;

### Tempo di contaminazione = Tp(Flip Flop Edge Triggered) = 10;

### Area = 4A(Flip Flop Edge Triggered) = 4 x 23 = 92

### **REGISTRO A 16 BIT**

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Studio delle metriche:**

### Tempo di propagazione = Tp(Flip Flop Edge Triggered) = 10;

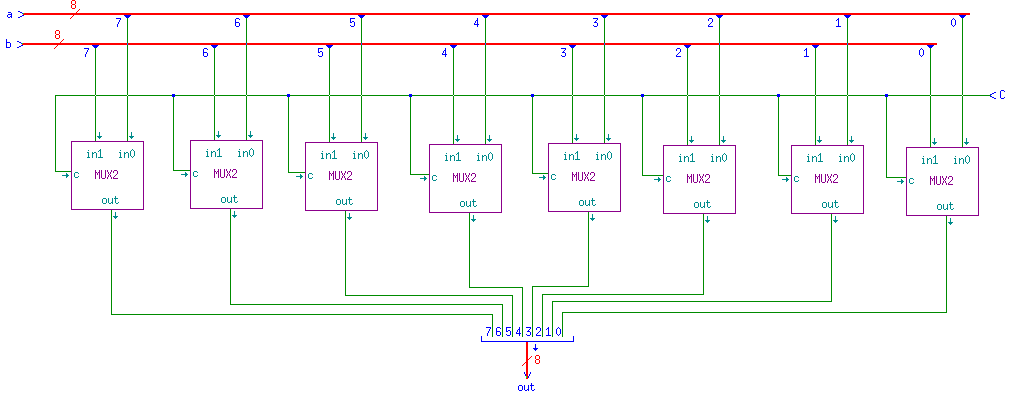
### Tempo di contaminazione = Tp(Flip Flop Edge Triggered) = 10;

### Area = 16A(Flip Flop Edge Triggered) =16 x 23 = 345

### 

### Multiplexer

### I Multiplexer sono stati necessari per potere adottare come stile di progetto la condivisione di risorse. Considerando che gli ingressi dei due multiplexer implementati non erano di un solo bit, è stato progettato un multiplexer da 8 bit. La progettazione del MUX è stata basata riutilizzando il Multiplexer normale, ovvero quello con gli ingressi entrambi da 1 bit. Per non dover ricorrere ad altri MUX è stato trasformato il primo ingresso a 4 bit, in un segnale ad 8 bit, in questo modo è stato utilizzato lo stesso MUX, sia per gli ingressi che per il risultato della prima moltiplicazione.

**MUX 8 BIT** 

**Studio delle metriche:** Tempo di propagazione = Tp(MUX) = 3;  
 Tempo di contaminazione = Tc(MUX) = 2;  
 Area = 8A(MUX) = 8 x 7 = 56.

### 

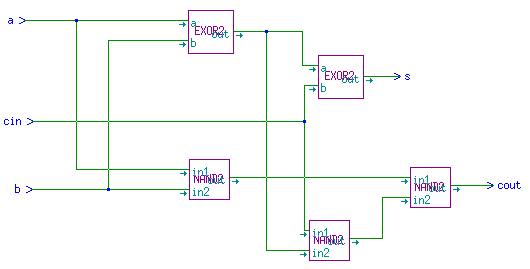
### 

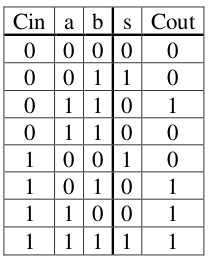
### 

### Macro funzionali

**FULL ADDER**

Questo componente è alla base dei Full Adder a n bit. Il componente riceve in ingresso il riporto di un’altra addizione e i due operandi (tutti a un bit) e restituisce la somma degli operandi e il riporto.





**Studio delle metriche:**

Tempo di propagazione = Tp(Exor) + 2Tp(Nand) = 3 + 2 = 5;

Tempo di contaminazione = 2Tp(Nand) = 2;

Area = 2A(Exor) + 3A(Nand) = 16 + 6 = 22.

Il calcolo della complessità circuitale è stimato contando il numero degli input delle porte logiche, per ottenere una stima molto più accurata.

# **FULL ADDER A 4 BIT**

# Il componente grazie all’uso di 4 Full Adder calcola la somma di due operandi a 4 bit. Usato nel moltiplicatore 4 x 4 bit.

# 

# **Studio delle metriche:**

# Tempo di propagazione = 4Tp(FA) = 20;

# Tempo di contaminazione = Tp(FA) = 5;

# Area = 4A(FA) = 4 x 22 = 88.

# 

# 

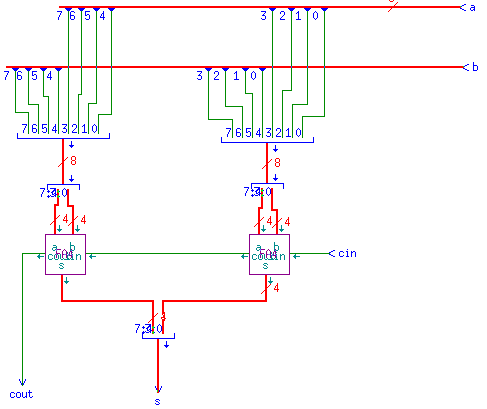
# 

# 

# 

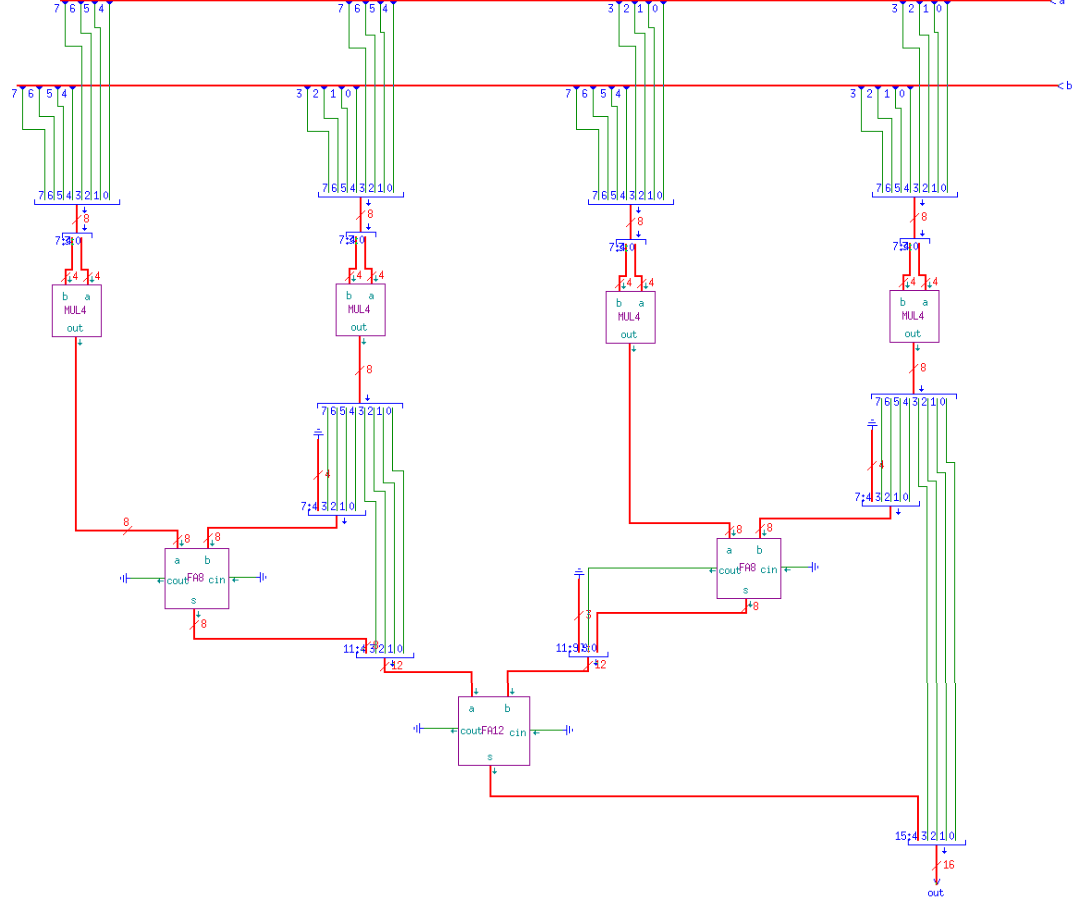
# **FULL ADDER A 8 BI**

Per realizzare questo componente è stato sfruttato l’incapsulamento, utilizzando i FA4 due volte, in modo tale da ottenere una somma tra 8 bit senza dover riutilizzare 8 FA. Questa porta logica è stata utilizzata nel circuito del moltiplicatore 8 bit x 8 bit.



**Studio delle metriche:**

* Tempo di propagazione = 2Tp(FA4) = 40;
* Tempo di contaminazione = Tp(FA4) = 20;
* Area = 2A(FA4) = 176.

**FULL ADDER A 12 BIT**

**Studio delle metriche:**

* Tempo di propagazione = Tp(FA4) + Tp(FA8) = 20 + 40 = 60
* Tempo di contaminazione = Tp(FA4) = 20
* Area = A(FA8) + A(FA4) = 176 + 88 = 264

Tutti i Full Adder, in generale, sono stati impiegati per poter sommare i prodotti parziali

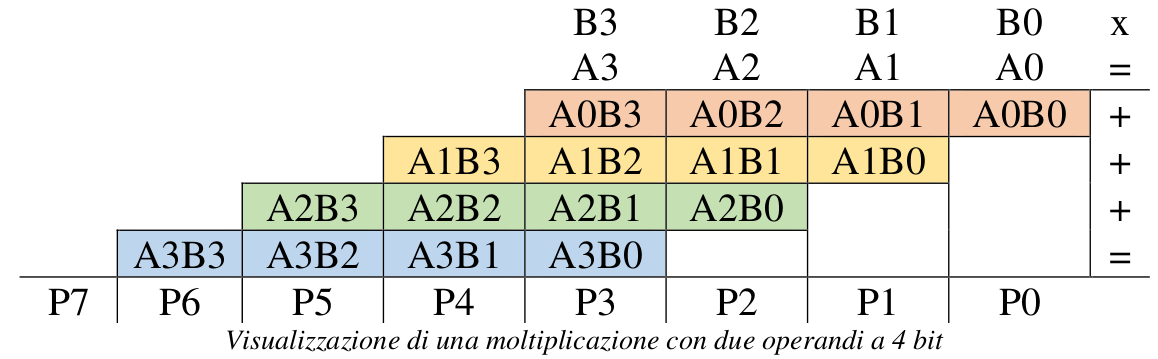
che si sono ottenuti durante l’operazione di moltiplicazione.

Moltiplicatori

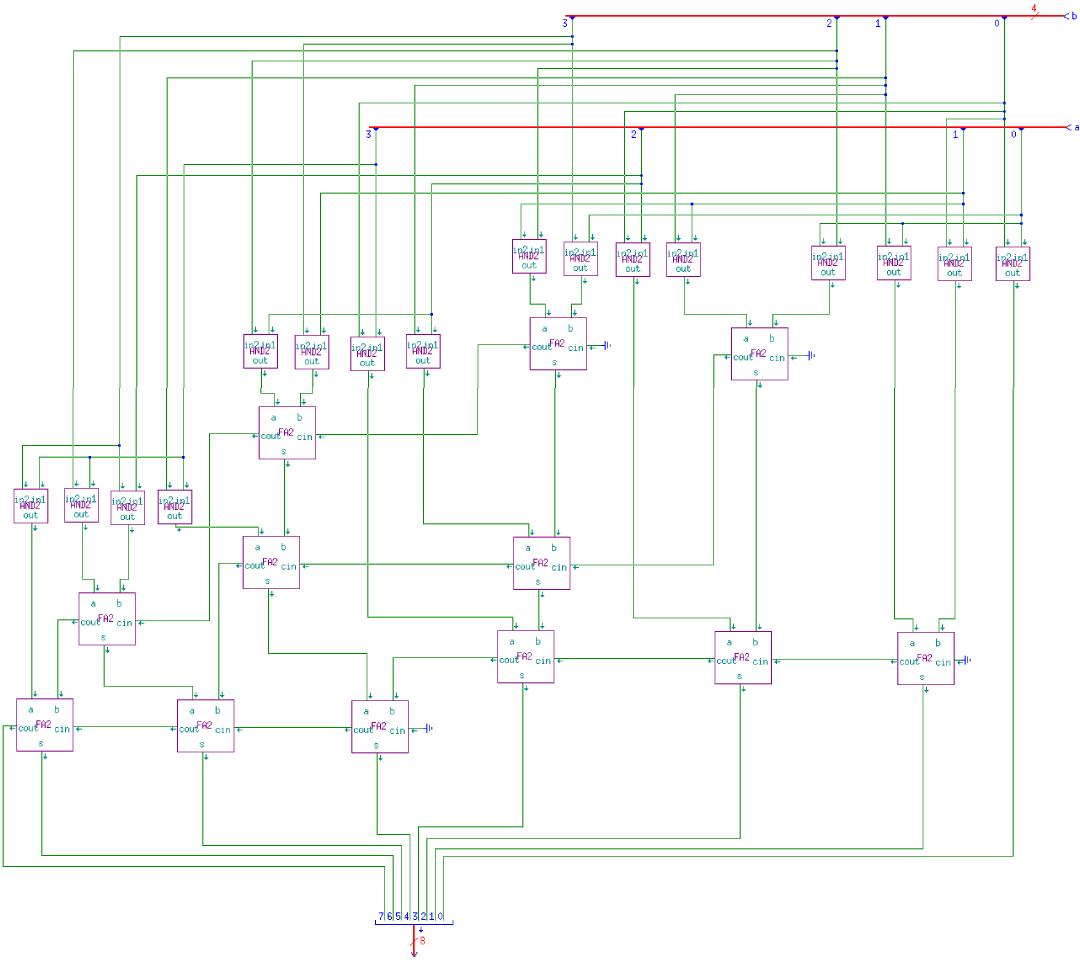
L’intera struttura di un quadrato si basa sulle moltiplicazioni, quindi è stato necessario sviluppare diversi moltiplicatori. Uno da 4 x 4 bit per calcolare il prodotto iniziale, uno da 8 x 8 bit per calcolare il prodotto finale . Di seguito verranno spiegati in dettaglio ognuno di essi.

**MOLTIPLICATORE 4X4 BIT**

Questo componente è utilizzato per calcolare il prodotto tra i due input ed è alla base per l’implementazione del moltiplicatore 8 bit × 8 bit.



Il suo funzionamento è semplice. Seguendo le regole per la moltiplicazione sappiamo che si creano 4 prodotti parziali da 4 bit ognuno. Per crearli usiamo 4 operatori AND per prodotto parziale (es. A0B0 è il risultato di A0 AND B0 e così via). Nel primo prodotto parziale il bit meno significativo viene restituito direttamente come primo bit del prodotto finale e così via verranno sommati tramite dei FA2 tutti i bit dai meno significativi ai più significativi.



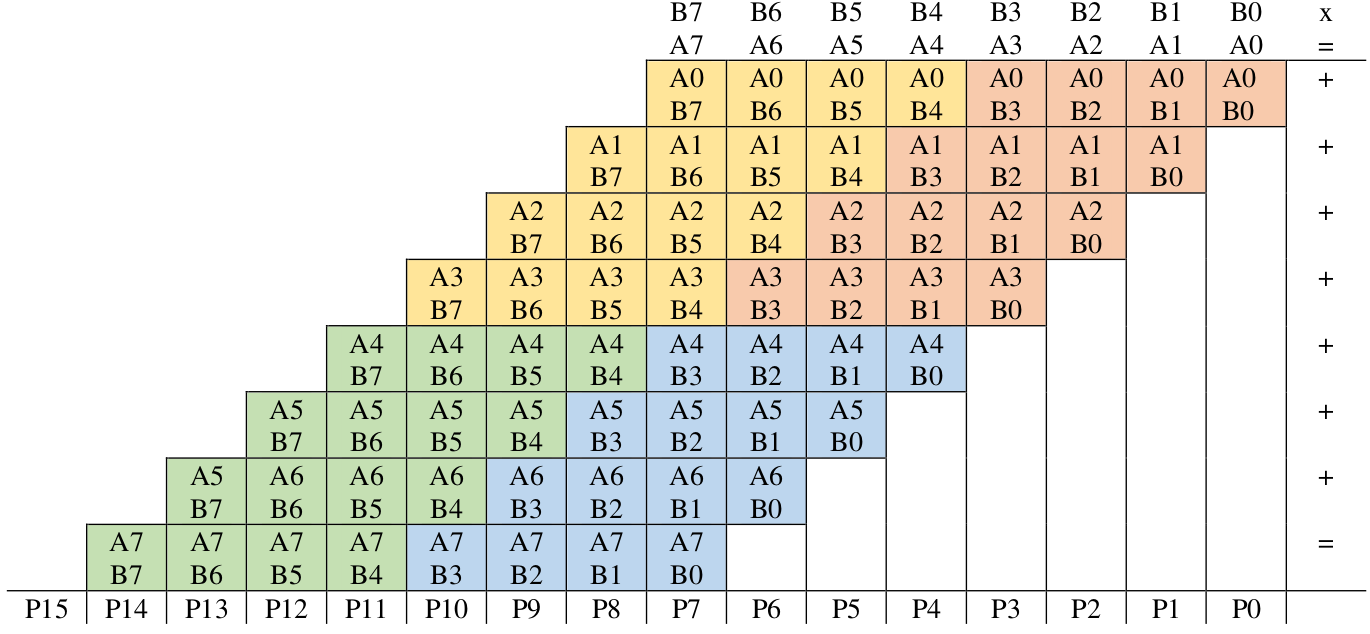
# **Studio delle metriche:**

# Tempo di propagazione (Tp) = Tp(And) + 6Tp(FA2) = 2 + 6 x 5 = 32;

# Tempo di contaminazione (Tc) = Tp(And) = 2;

# Area = 16A(And) + 12A(FA2) = 16 x 3 + 12 x 22 = 264.

# **MOLTIPLICATORE 8X8 BIT**

Realizzato utilizzando quattro moltiplicatori 4 bit × 4 bit e FA8. La struttura è stata realizzata partendo dalla moltiplicazione in colonna e notando che la parte dei prodotti parziali poteva essere divisa in quattro sottosezioni (come in figura) che sono state elaborate tramite dei moltiplicatori 4 bit × 4 bit. I risultati di queste sottosezioni sono stati poi sommati tramite i Full Adder da 8 bit sopra descritti.  


Seguendo lo schema sopra riportato, ogni sezione distinta da un colore diverso rappresenta una moltiplicazione 4 bit x 4 bit. Ciascuna riga dello schema colorato identifica una sequenza di prodotti parziali, sommando le righe si ottiene il risultato voluto, distinto dalla successione di . Ogni rappresenta un singolo bit.

# Struttura del moltiplicatore:

# 

# **Studio delle metriche:**

# Tempo di propagazione (Tp) = Tp(Mul4) + Tp(FA8) + Tp(FA12) = 62 + 40 + 60 = 162;

# Tempo di contaminazione (Tc) = Tp(Mul4) = 62;

# Area = 4A(Mul4x4) + 2A(FA8) + A(FA12) = 4 x 312 + 2 x 176 + 352 = 1248+ 352 + 264 = 1864.

# 

# Data Path

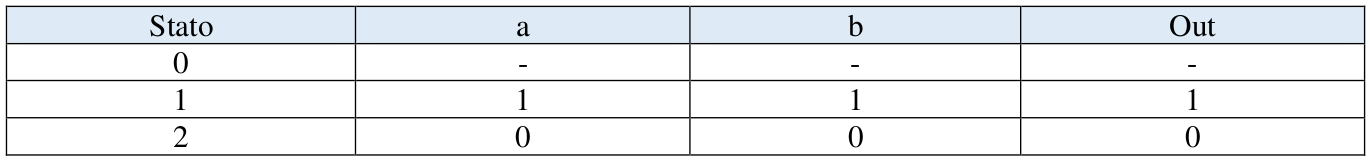
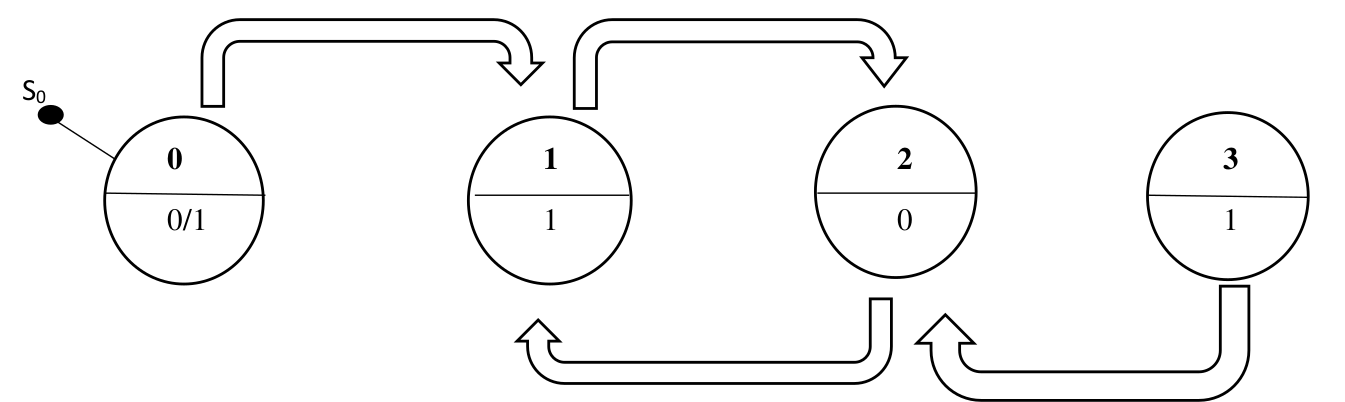
# 

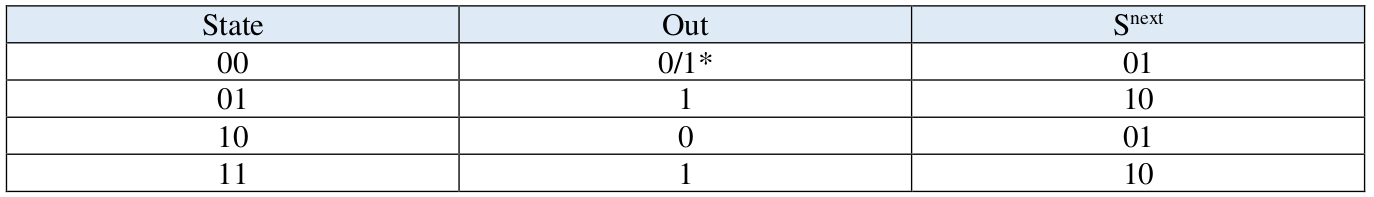
# Control Unit

### Specifica

La Control Unit è stata implementata per poter gestire i due MUX presenti nel circuito del quadrato. Questo componente deve essere attivato. Per gestire ciò viene utilizzato uno switch per fornire il segnale di avvio (“start”). Considerando che durante il primo ciclo di clock i multiplexer non sono necessari, il primo segnale di uscita della Control Unit è irrilevante. Utilizzando uno switch per far avviare la C.U., il primo segnale di uscita che rappresenterà il segnale di controllo per i MUX, sarà un segnale logico alto ma questo non influisce sulla corretta gestione dei multiplexer\*.

Tabella di verità per gestire i segnali di controllo che riceveranno i MUX. Come citato sopra, il primo stadio è irrilevante:

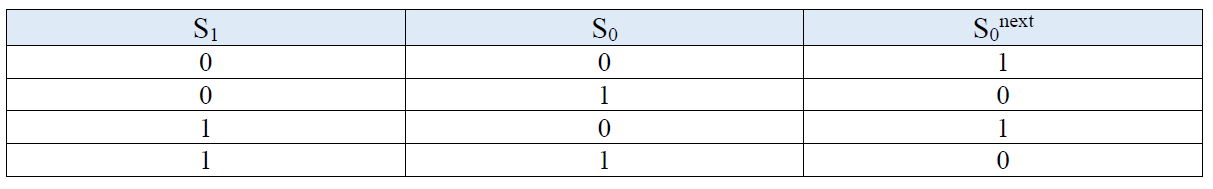
  
Il diagramma degli stati per simulare il comportamento della Control Unit è il seguente:  
  
Codifica degli stati:



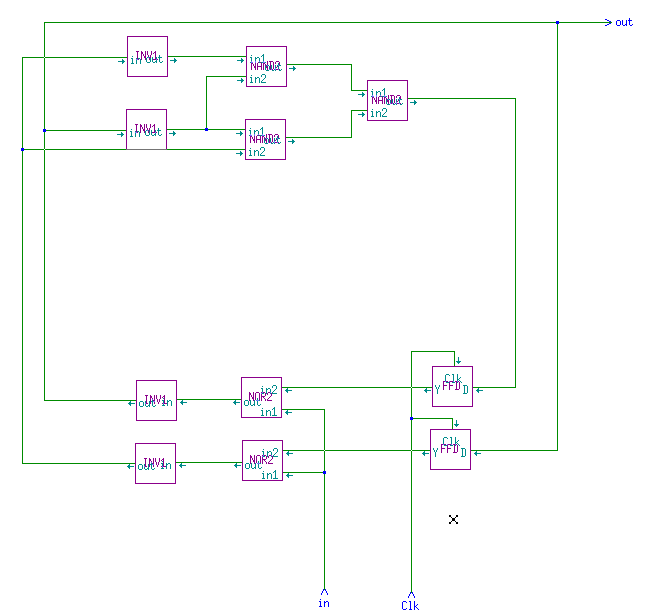
Il quarto stato rappresenterebbe una zona indesiderata in cui la C.U. non dovrebbe mai presentarsi. In caso la cosa si verificasse, lo stato futuro diventerà il secondo.

### Implementazione della macchina di Moore

Tabella di verità per lo stato futuro:



Quindi l’espressione booleana che si ottiene da questa tabella di verità è la seguente:

Circuito della Control Unit su Tkgate 2.1:  


# 

# 

# 

# 

# 

# 

# **Studio delle metriche:**

# Tempo di propagazione (Tp) = Tp(Nor) + Tp(Inv) = 1 + 1 = 2

# Tempo di contaminazione (Tc) = Tp(Nor) + Tp(Inv) = 1 + 1 = 2

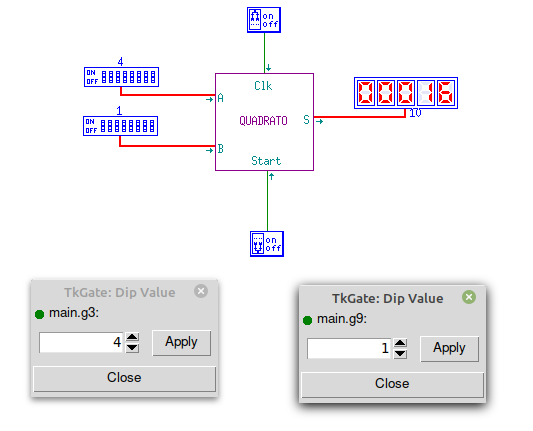
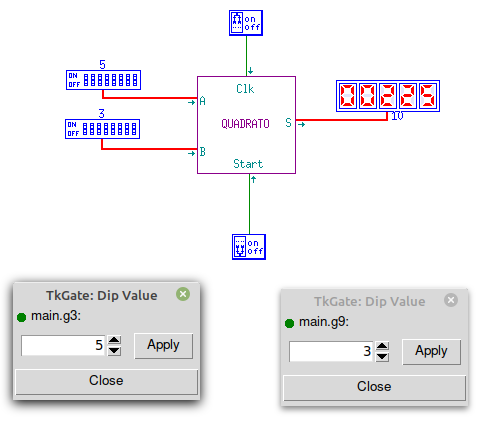
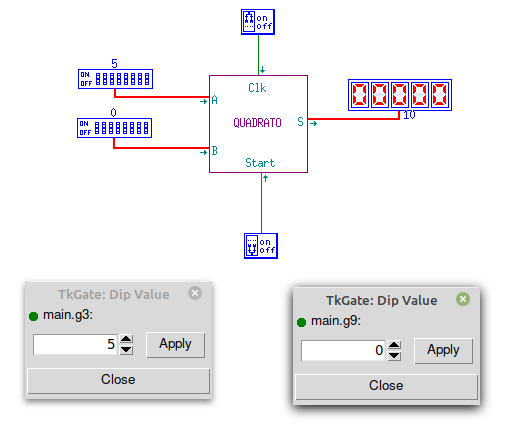
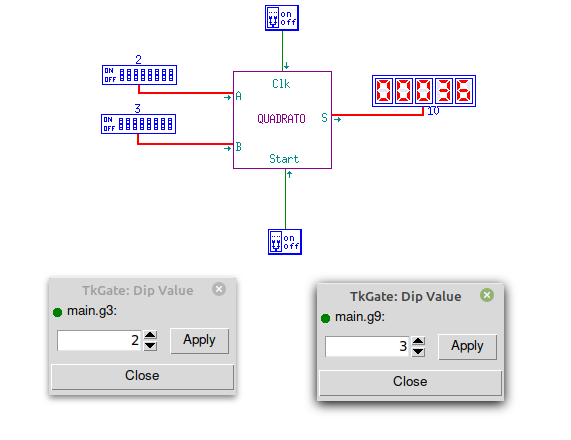
# Area = 4A(Not) + 3A(Nand) + 2A(Nor) + 2A(Flip Flop Edge Triggered) = 4 x 1 + 3 x 2 + 2 x 2 + 2 x 23 = 60.

L’Inverter e il NOR a due ingressi servono sia per ottenere il segnale da “start” per fare avviare la Control Unit che per stabilizzare i segnali retroattivi.

# 

# Simulazione e analisi del progetto

### Verifica funzionale

Di seguito vengono riportati alcuni test significativi per mostrare il corretto funzionamento del circuito.  
  


### 

### 

### 

### 

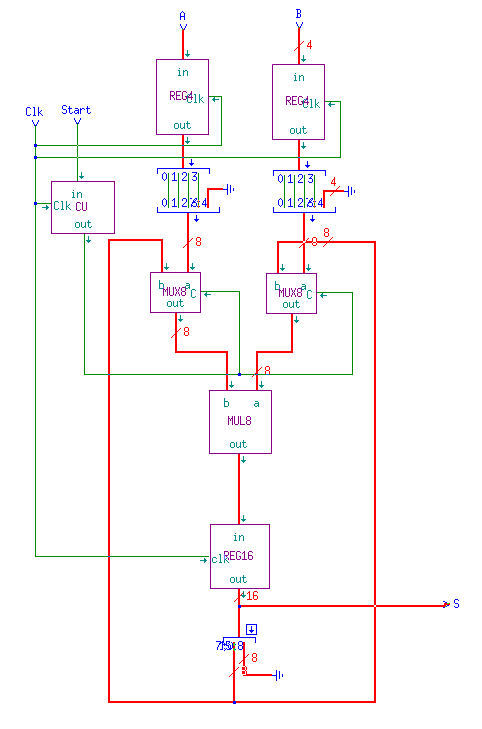
### 

### Valutazione di prestazioni e complessità

Seguendo il Data Path, è stata creata la macro funzionale che si occupa di calcolare

la specifica richiesta.

Sono stati impiegati il moltiplicatore 8 bit x 8 bit, per eseguire il prodotto iniziale tra i due parametri inseriti in input, i registri (per inserire opportuni sincronismi), la Control Unit e i Multiplexer. Nel moltiplicatore è implementata la condivisione di risorse, così facendo il risultato del quadrato è elaborato in tre cicli di clock.



### 

### 

### 

### 

**Studio delle metriche**:

* Tempo di propagazione(TP) = Tp(REG4) + 2Tp(MUX8) + 2Tp(MUL8) + 2Tp(REG16) = 10 + 2 x 3 + 2 x 162 + 2 x 10 = 360
* Tempo di contaminazione(TC) = Tc(REG4) + Tc(MUX8) + Tc(MUL8) + Tc(REG16) =

10 + 2 + 62 + 10 = 84

* Area = 2A(REG4) + 2A(MUX8) + A(MUL8) + A(REG16) + A(CU)= 2 x 92 + 2 x 56 + 1864 + 345 + 60 = 2565