PROGETTO PER IL CORSO DI:

RETI LOGICHE

Sessione estiva 2020/2021

Studente: Marzio Della Bosca

Numero di matricola: 306034

Iscritto al 1° anno del corso di laurea triennale in:

INFORMATICA APPLICATA

Indice

1. Specifiche di progetto:

Il progetto prevede la progettazione tramite Tkgate di un circuito in grado di eseguire, su operandi a 8 bit, le quattro operazioni base:

- Addizione;
- Sottrazione;
- Moltiplicazione;
- Divisione;

Specifica funzionale:

Addizione e sottrazione:

I due operando a otto bit vengono processati, nel caso dell'addizione viene necessario considerare un bit in più dato che la somma massima tra due operandi a otto bit ne richiede nove. Nel caso della sottrazione il secondo operando viene reso negativo in modo da poter ottenere una sottrazione.

Addizione: A + B = C
Sottrazione: A + (-B) = C

A, B = 8 bit;
C = 9 bit;

Moltiplicazione e divisione:

Nel caso della moltiplicazione i due operandi a otto bit vengono processati ed il risultato è espresso a 16 bit, dato che il valore massimo ottenibile da una moltiplicazione tra operandi a otto bit ne richiede 16, mentre per la divisione il risultato viene espresso a otto bit come per il resto in quanto il valore massimo risultante tra una divisione tra operandi a otto bit ne richiede otto, come per il resto.

Moltiplicazione: A x B = AB
Divisione: A / B = C + Resto

A,B = 8 bit;
AB = 16 bit;
C, Resto = 8 bit;

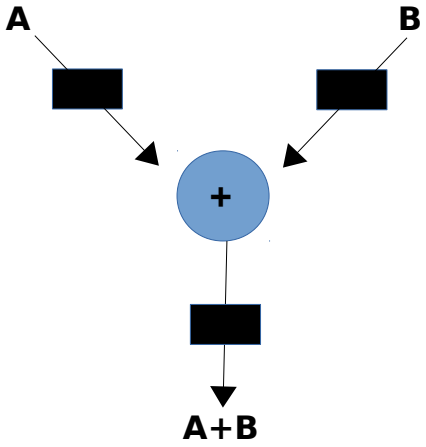
Specifica parametrica:

Per addizione e sottrazione è stato scelto di diminuire la complessità circuitale tramite resource sharing a discapito del numero di clock necessari, in quanto per ottenere i risultati ne sono necessari nove.

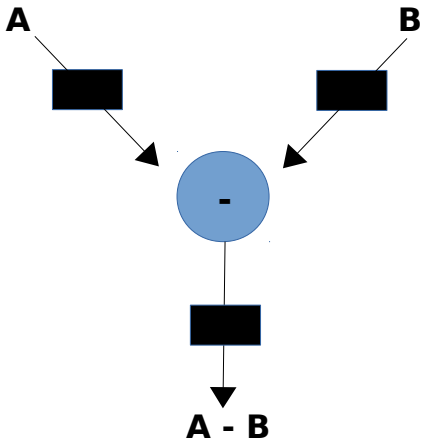
Mentre per moltiplicazione e divisione è stato scelto di non incidere sul numero di clock perché sarebbe stato proibitivo effettuare un numero così alto di cicli di clock manualmente.

2. Impostazione a livello RT:

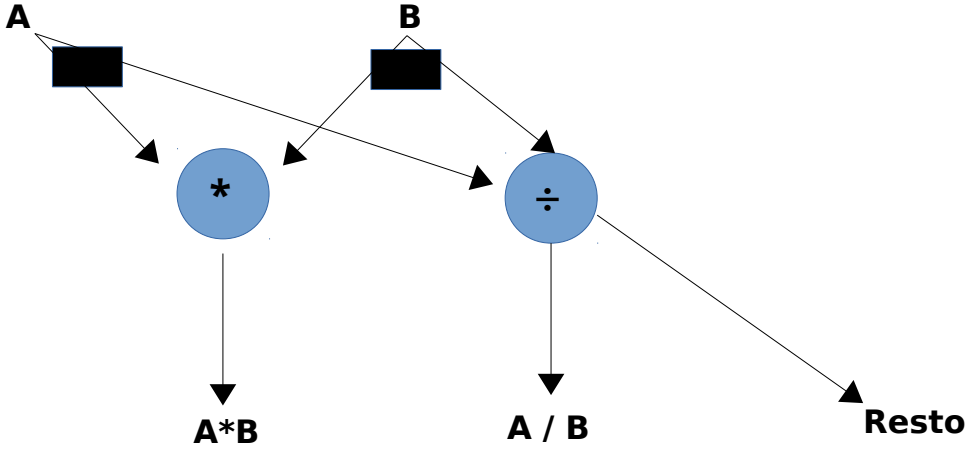
Addizione:



Sottrazione:



Moltiplicazione / Divisione:



3. Progettazione delle risorse:

Le risorse necessarie per implementare il seguente circuito sono:

- **Macro-aritmetiche:**

  un addizionatore/sottrattore con ingresso a 8 bit e 9 bit in uscita;

  un moltiplicatore con ingresso a 8 bit e 16 bit in uscita;

  un divisore con ingresso a 8 bit e 16 bit in uscita: 8 per il risultato e 8 per il resto;

- **Registri:**

  un registro a scorrimento/parallelo a 8 bit come ingresso del primo operando;

  un registro a scorrimento/parallelo a 8 bit in grado di convertire il secondo operando in complemento a 2;

  un registro a scorrimento a 9 bit come uscita per contenere il risultato di somma e sottrazione;

Per quanto riguarda le risorse utilizzate dalle macro-aritmetiche:

1. **Addizionatore/sottrattore:**

   E' composto (oltre che dai registri già visti) da:
   - 2 "inverter";
   - 1 "multiplexer" a due bit di entrata;
   - 1 "Full Adder" a due entrate;
   - 1 "Flip Flop D" a entrata singola;
   - 1 "AND" a due entrate;

2. **Moltiplicatore:**

   E' composto da:
   - 64 porte logiche "AND" ;
   - 48 "Full Adder" ;
   - 8 "Half Adder" ;

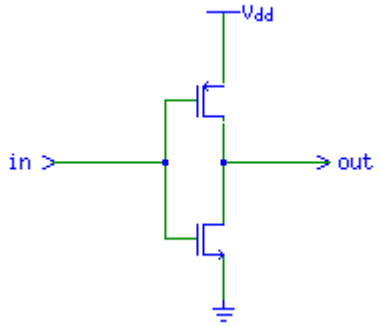
3. **Divisore:**
   - 64 "Full Substracter" ;
   - 57 "Multiplexer" ;
   - 8 "Inverter";

4. Porte logiche elementari:

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

- **INVERTER:**

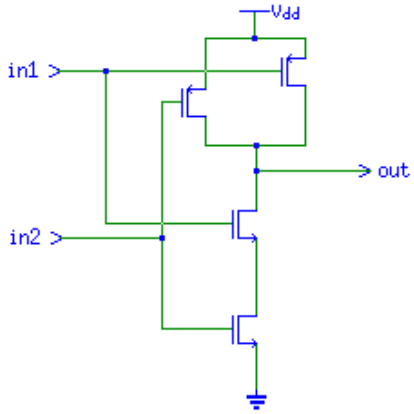
A	f (A)
0	1
1	0



La porta logica “inverter” rappresenta la negazione logica “NOT” negando appunto l’informazione in entrata.

- **NAND:**

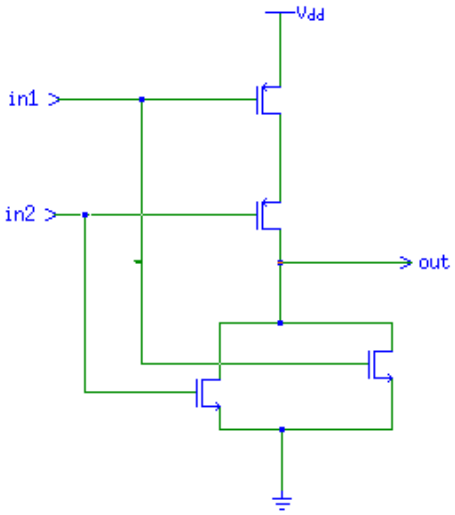
A	B	f(AB)
0	0	1
0	1	1
1	0	1
1	1	0



La porta logica “nand” rappresenta la congiunzione logica “and” in forma negata.

- **NOR:**

A	B	f(AB)
0	0	1
0	1	0
1	0	0
1	1	0



La porta logica “nor” rappresenta la disgiunzione logica “or” in forma negata.

5. Porte logiche composte:

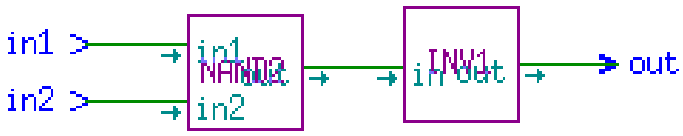
Le porte logiche composte utilizzate per la realizzazione del progetto sono state le seguenti:

- **AND:**

A	B	f(AB)
0	0	0
0	1	0
1	0	0
1	1	1

Metriche:

Tempo di contaminazione: 2
Tempo di propagazione: 2
Area: 3

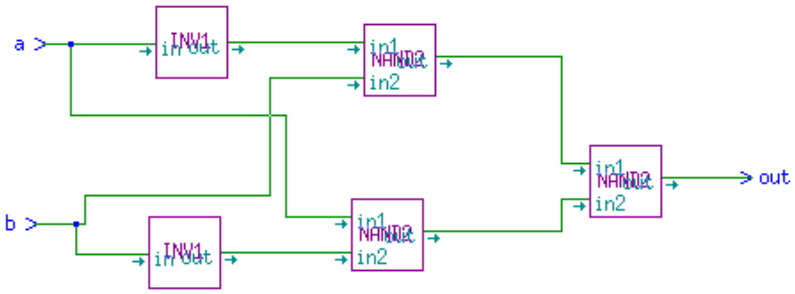


- **EXOR:**

A	B	f(AB)
0	0	0
0	1	1
1	0	1
1	1	0

Metriche:

Tempo di contaminazione: 2
Tempo di propagazione: 3
Area: 8

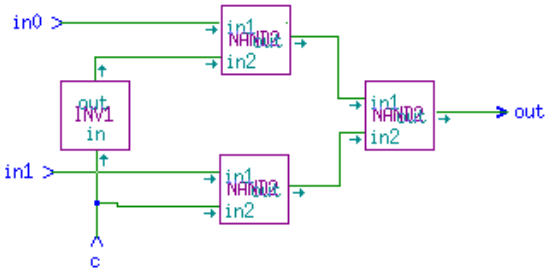


- **Multiplexer:**

I "multiplexer" , nel progetto denominati "MUX", nel progetto sono stati implementati per rendere possibile lo scarico sequenziale o parallelo a seconda dell'operazione richiesta.

Metriche:

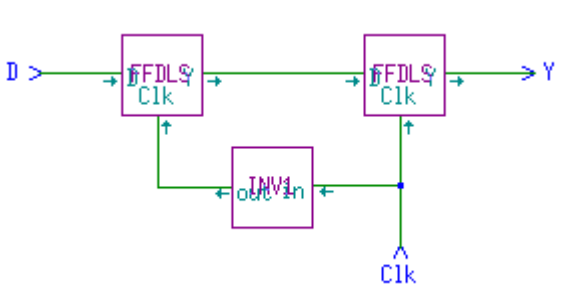
Tempo di contaminazione: 2
Tempo di propagazione: 3
Area: 7



Registri:

I registri nel progetto sono stati implementati per suddividere l'elaborazione in diversi cicli di clock. Un registro ad n bit presenta al suo interno n Flip Flop D Edge Triggered.

- **Flip Flop D Edge Triggered (FFD):**

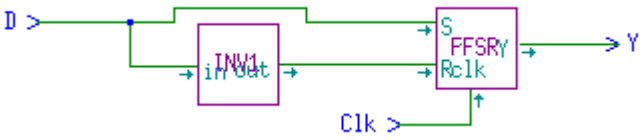


Metriche:

Tempo di contaminazione: 4
Tempo di propagazione: 9
Area: 23

Composto da un inverter e due "FFDLS" o Flip Flop D Level Sensitive:

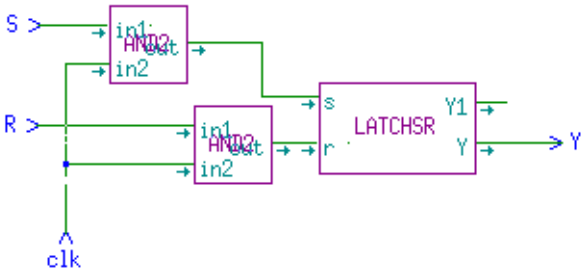
Metriche:



Tempo di contaminazione: 4
Tempo di propagazione: 4
Area: 11

Composto da un "inverter" e un "FFSR" o Flip Flop Set Reset:

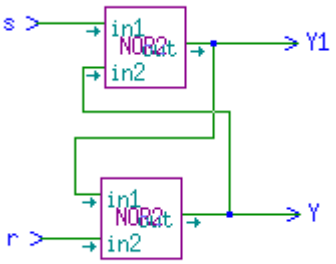
Metriche:



Tempo di contaminazione: 3
Tempo di propagazione: 3
Area: 10

Composto da due "AND" e un "LATCHSR" :

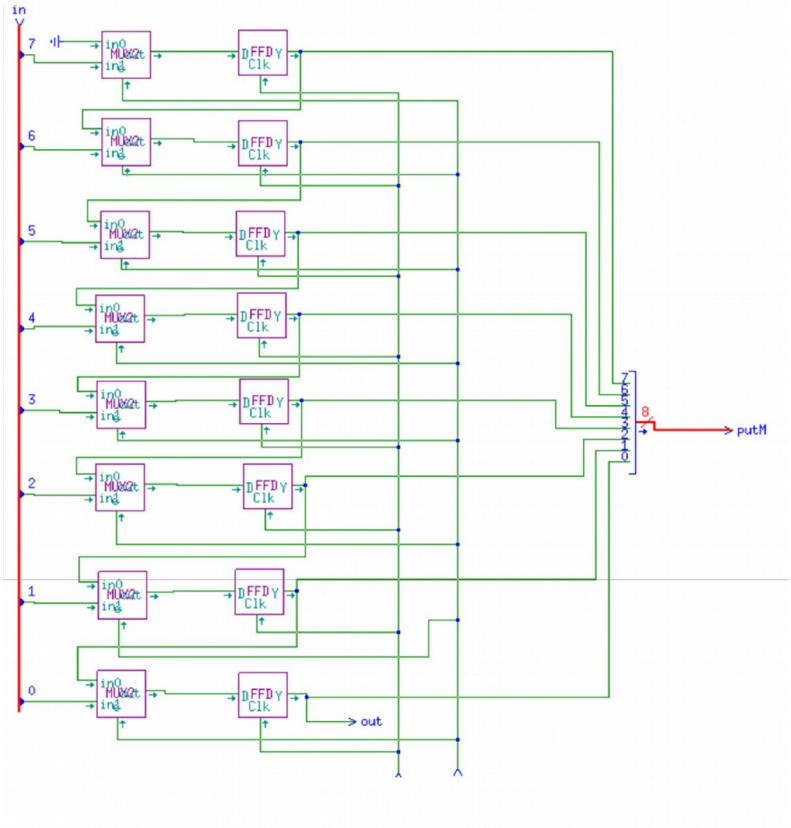
Metriche:



Tempo di contaminazione: 1
Tempo di propagazione: 1
Area: 4

Registri composti:

- **REG8sIN:**

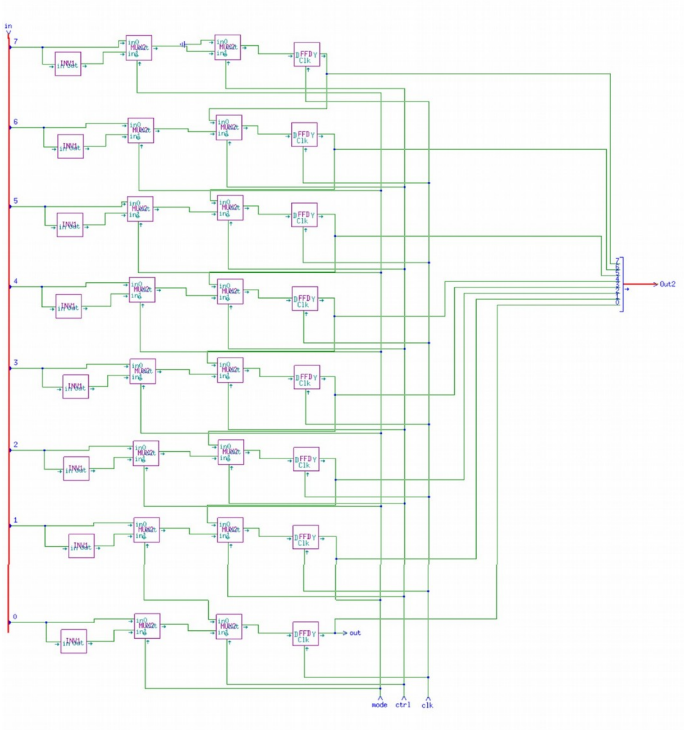


Metriche:

Tempo di contaminazione: 8
Tempo di propagazione: 14
Area: 216

Questo registro “scarica” l’informazione in maniera sequenziale o parallela a seconda del bit di controllo, nel caso in cui lavora in maniera sequenziale sono necessari 8 cicli di clock (se l’informazione contenuta ha dimensione a 8 bit).

- **Reg8sINaddsub:**

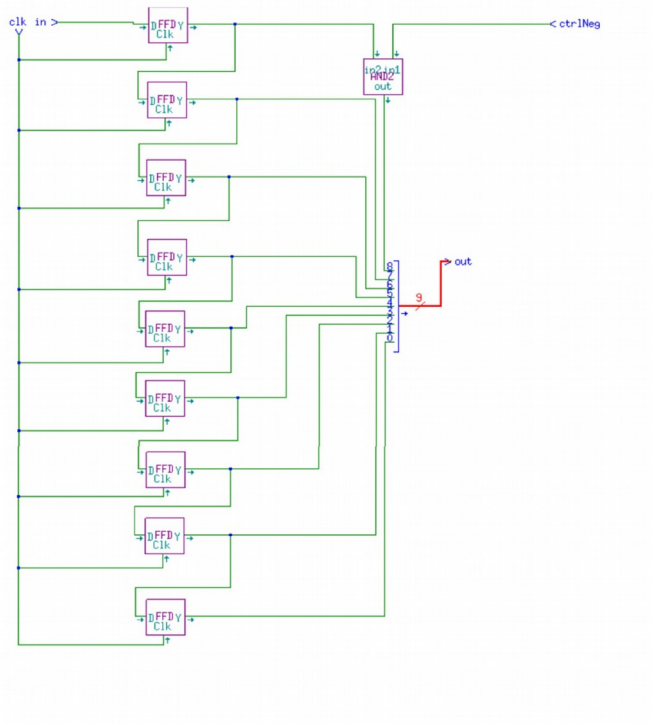


Metriche:

Tempo di contaminazione: 13
Tempo di propagazione: 20
Area: 304

Questa registro per quanto riguarda lo “scarico” di informazione sequenziale e parallela opera allo stesso modo del precedente, la differenza sta nel fatto che a seconda di un ulteriore bit di controllo converte il numero contenuto col suo reciproco attraverso la rappresentazione a complemento a 2.

- **Reg9sOUT:**

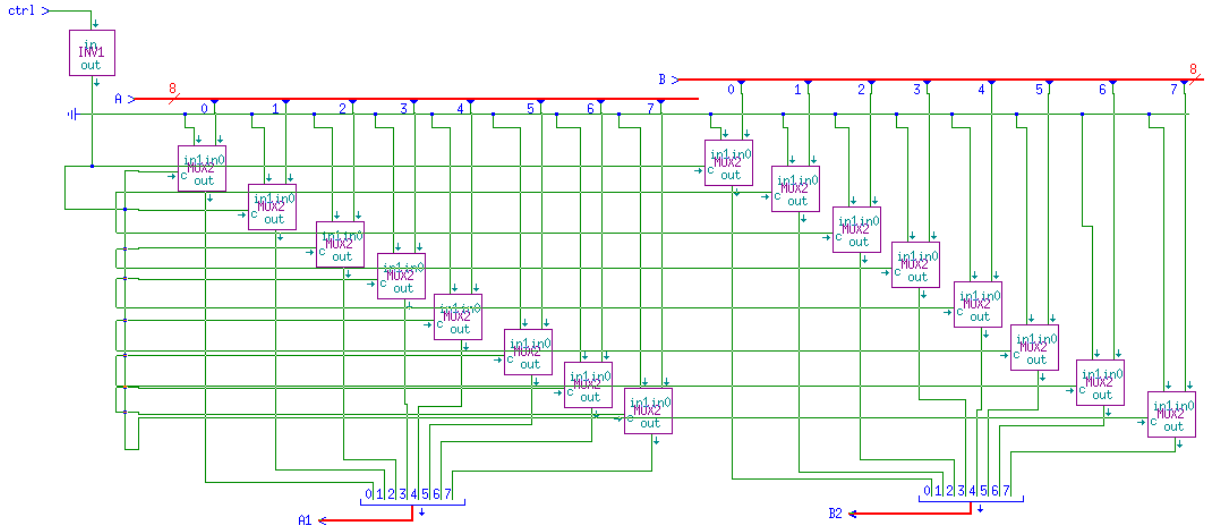


Metriche:

Tempo di contaminazione: 6
Tempo di propagazione: 36
Area: 209

Questo registro viene caricato in maniera sequenziale, quindi un bit per volta, e “scarica” parallelo. E’ stato aggiunto una porta logica “AND” che azzera il bit di peso 9 a seconda che il controllo dell’ operazione di sottrazione sia alto o basso, in questo modo non si falsa la sottrazione.

Nel progetto è stato implementato un **multiplexer a 2 entrate da 8 bit** ciascuna in modo da rendere le operazioni di somma e sottrazione più stabili in quanto ad ogni ciclo di clock il moltiplicatore e il divisore collegati direttamente ai registri degli operandi dovevano ricalcolare rendendo la simulazione su tkgate poco fluida. Il modulo in questione manda ai moduli di divisione e moltiplicazione i bit degli operandi solo quando i registri dei suddetti scaricano in parallelo ed il secondo operando non è in complemento a 2, in caso contrario vengono mandati segnali collegati a “terra”.



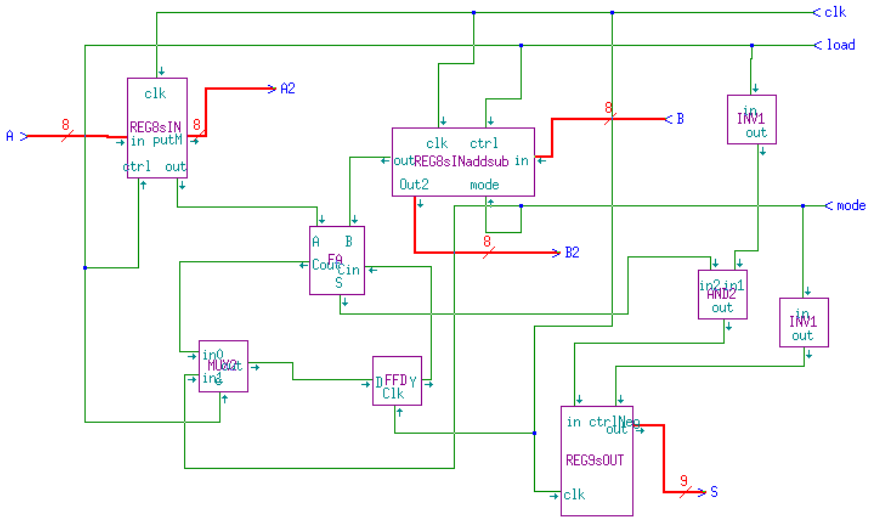
Metriche:

Tempo di contaminazione: 2
Tempo di propagazione: 3
Area: 113

6. Macro funzionali:

BIT SERIAL ADDER/SUBSTRACTER

Si tratta di un modulo che opera addizione e sottrazione attraverso un unico “Full Adder”, è una versione del “Bit serial Adder” modificata in modo da poter operare sia somme che sottrazioni sfruttando la rappresentazione in complemento a 2 dei numeri espressi in binario. Opera su operandi a 8 bit, e rappresenta correttamente soluzioni fino a 9 bit, poiché il numero massimo rappresentabile a 8 bit è il 255, e la somma 255 + 255 = 510 che richiede appunto 9 bit di informazione per essere rappresentato correttamente.



Metriche:

Tempo di contaminazione:12
Tempo di propagazione: 62
Area: 790

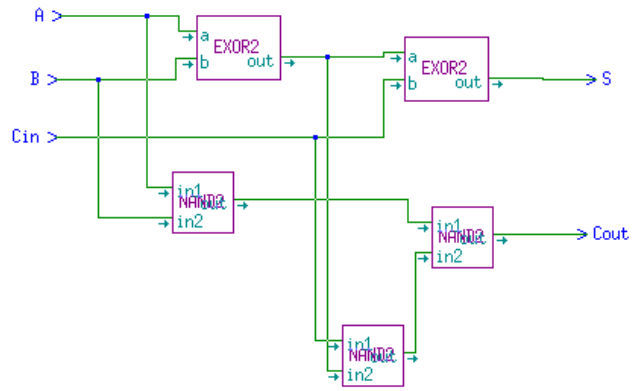
Full Adder:

Questo modulo prende due bit in ingresso (uno per ogni ingresso) e ne fa la somma tenendo conto dell’eventuale riporto:

A	B	C.in	C.out	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C.in = “carry in” ovvero il riporto in entrata.
C.out = “carry out” ovvero il riporto in uscita.
S = somma.
A,B sono gli operandi.
Metriche:

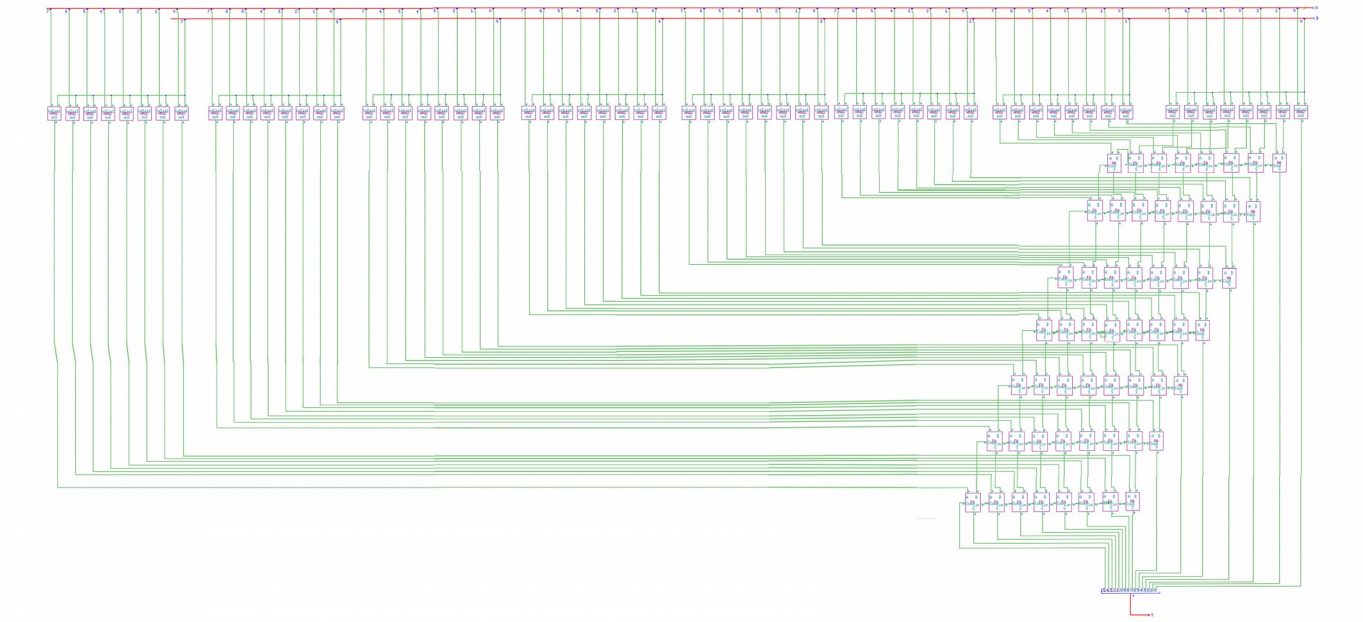
Tempo di contaminazione: 4
Tempo di propagazione: 6
Area: 26



MULTIPLIER:

Il modulo rappresentante nel progetto è denominato “Mul16b”, si tratta di un modulo che opera su operandi da 8 bit, ed opera principalmente in due passaggi, il calcolo dei prodotti parziali in cui ogni bit del primo operando viene messo in relazione con ogni bit del secondo operando, tramite una porta logica “AND” e poi si effettuano le somme delle righe contenenti i prodotti parziali. Può rappresentare correttamente prodotti che richiedono fino ad un massimo di 16 bit in quanto il massimo prodotto operabile con operandi a 8 bit è 255 * 255 = 65’025 che necessita di 16 bit per essere rappresentato correttamente.

								A7 B7	**A6 B6**	**A5 B5**	**A4 B4**	**A3 B3**	**A2 B2**	**A1 B1**	**A0 B0**	**X**
								A7B0	A6B0	A5B0	A4B0	A3B0	A2B0	A1B0	A0B0	+
							A7B1	A6B1	A5B1	A4B1	A3B1	A2B1	A1B1	A0B1		+
						A7B2	A6B2	A5B2	A4B2	A3B2	A2B2	A1B2	A0B2			+
					A7B3	A6B3	A5B3	A4B3	A3B3	A2B3	A1B3	A0B3				+
				A7B4	A6B4	A5B4	A4B4	A3B4	A2B4	A1B4	A0B4					+
			A7B5	A6B5	A5B5	A4B5	A3B5	A2B5	A1B5	A0B5						+
		A7B6	A6B6	A5B6	A4B6	A3B6	A2B6	A1B6	A0B6							+
	A7B7	A6B7	A5B7	A4B7	A3B7	A2B7	A1B7	A0B7								
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**	



Metriche:
Tempo di contaminazione: 2
Tempo di propagazione: 30
Area: 1464

Nell’implementazione del “multiplier” sono stati usati:

- 64 “AND” ;
- 48 “FULL ADDER” ;
- 8 “HALF ADDER” ;

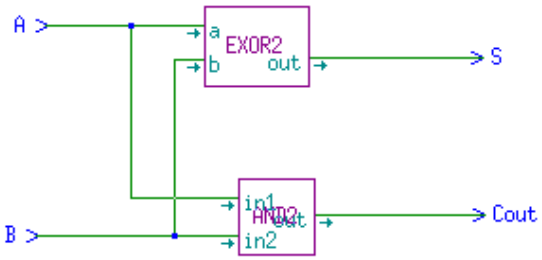
HALF ADDER

Il semi-sommatore o “Half Adder” è un componente che riceve in ingresso 2 bit e ne calcola la somma e il relativo riporto. Viene implementato utilizzando una porta EXOR e un AND.

A	B	C.out	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Metriche:

Tempo di contaminazione: 2
Tempo di propagazione: 3
Area: 11



DIVIDER:

Opera la divisione su operandi a 8 bit, in qui il primo “A” è il dividendo e “B” è il divisore. Ha due uscite a 8 bit ciascuna, una per il quoziente e una per il resto della divisione. Sono a 8 bit poiché nel caso che il dividendo venga diviso per 1 che è il valore minimo, il quoziente sarà eguale al dividendo che essendo di 8 bit resterà tale. Stesso discorso se il divisore è più grande del dividendo, il quoziente sarà 0 che richiede un solo bit per essere rappresentato ed il resto sarà uguale al dividendo, di 8 bit.

Algoritmo per la divisione:

A : dividendo; B : divisore; Q : quoziente; R : resto;

L’esempio sarà fatto su una divisione su operandi a 4 bit in modo da rendere l’esposizione più ordinata:

$Q_3\ Q_2\ Q_1\ Q_0$

$B_3\ B_2\ B_1\ B_0 \overline{|\ A_3\ A_2\ A_1\ A_0}$

$X_1 \longrightarrow B \overline{|\ 0\ 0\ 0\ A_3} \longrightarrow$ Il che implica 0 se B è più grande.

$Q_3 = X_1$

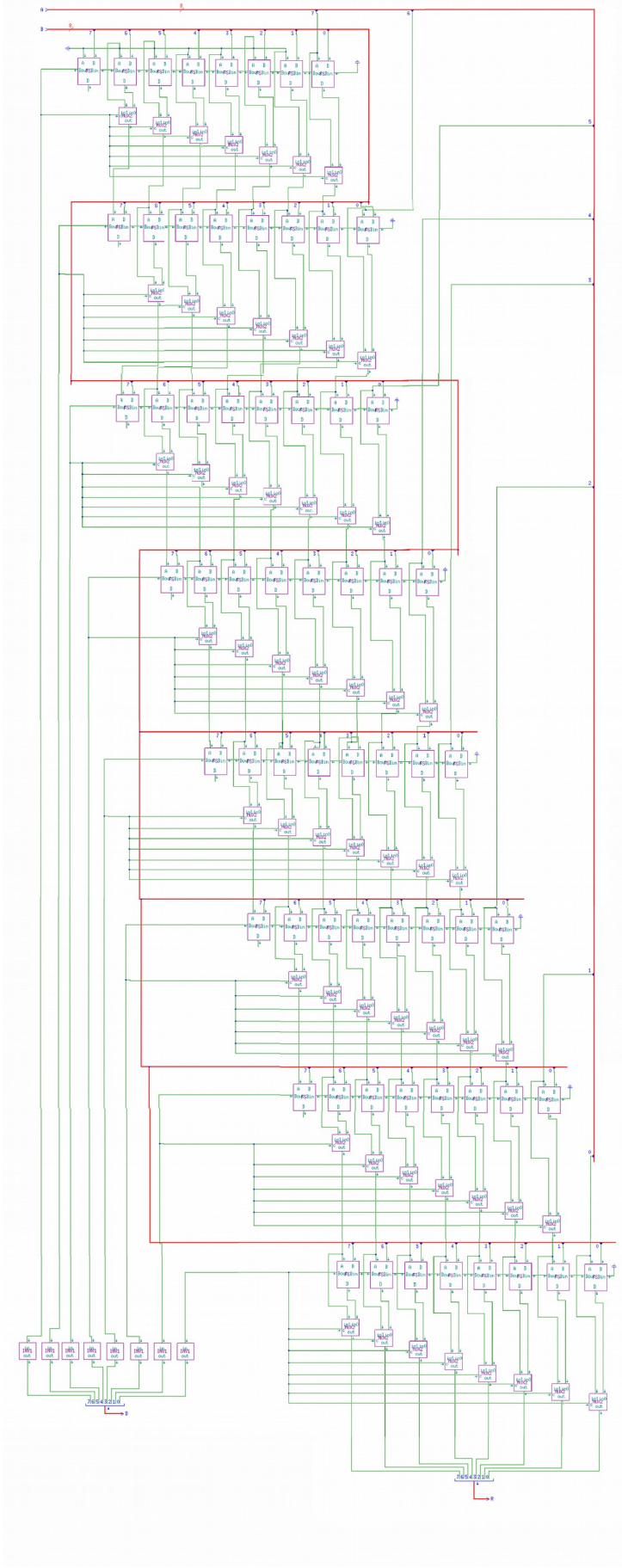
$Q_2 = X_2$ $\quad Y_1\ X_2 \longrightarrow B \overline{|\ 0\ 0\ Y_1\ A_2}$ (ES. $Y_1 = A_3 - B$)

$Q_1 = X_3$ $\quad Y_1\ Y_2\ X_3 \longrightarrow B \overline{|\ 0\ Y_1\ Y_2\ A_1}$

$Q_0 = X_4$ $\quad Y_1\ Y_2\ Y_3\ X_4 \longrightarrow B \overline{|\ Y_1\ Y_2\ Y_3\ A_0}$

$\longrightarrow R_3\ R_2\ R_1\ R_0$

Il divider, rappresentato dal modulo “Div8b” nel programma, esegue questo algoritmo, sfruttando la sottrazione per capire se il divisore è più grande del dividendo (parziale). il B.out del “substracter” impiegato nella sottrazione in esame va a riempire il corrispondente posto del quoziente e nel fare questo comanda quale entrata del multiplexer deve essere fatta scendere per la prossima divisione parziale, cioè se prendere il bit del dividendo, nel caso che B sia più grande del parziale di A, viene fatto “scendere” il bit di A_x poiché il B.out, che sarebbe il Q_x , è 0. In caso contrario vengono fatte scendere i risultati (ognuno in un “full substracter” differente) delle sottrazioni tra i bit di A e B. Il che è esattamente come espresso dall’algoritmo della divisione. All’ultimo passo, cioè quando viene preso in esame l’ultimo bit del dividendo, viene fatta la medesima operazione un ultima volta e ciò che scende sarà il resto della divisione in numeri interi (espressa in binario).



Metriche:

Tempo di contaminazione: 7
Tempo di propagazione: 88
Area: 2135

Full Substracter

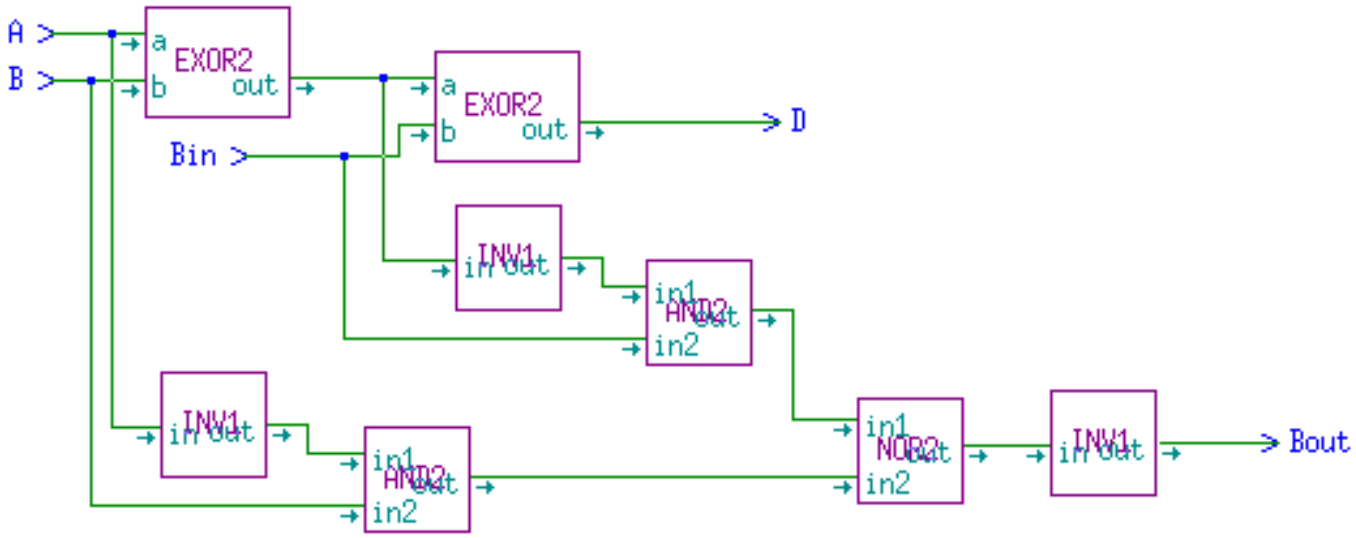
Questo modulo prende due bit in ingresso (uno per ogni ingresso) e ne fa la sottrazione tenendo conto dell'eventuale riporto:

A	B	B.in	B.out	f(AB)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

B.in = resto in entrata
B.out = resto in uscita
D = differenza

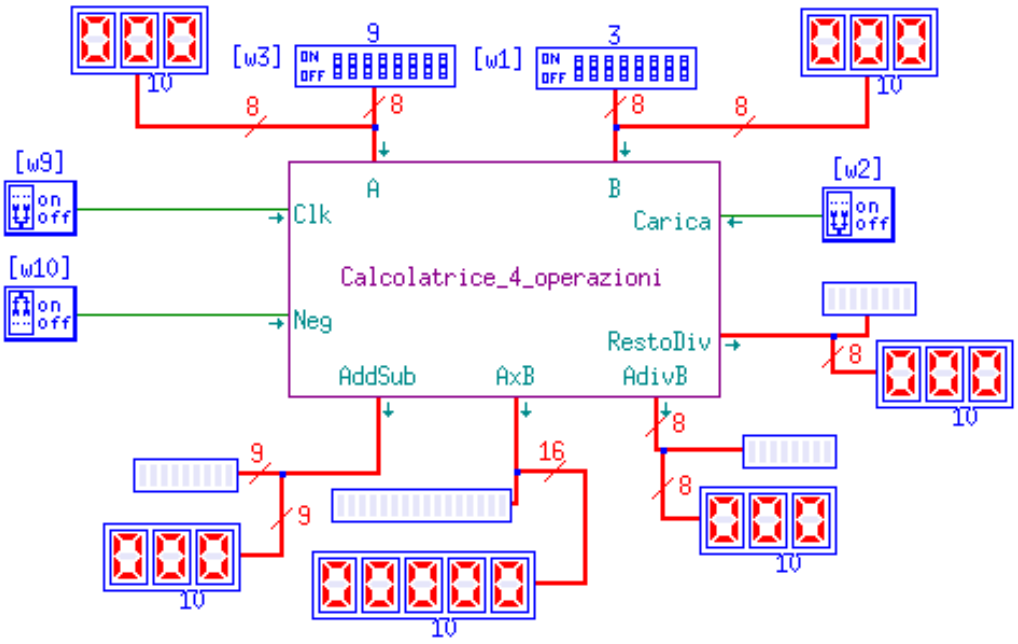
Metriche:

Tempo di contaminazione: 6
Tempo di propagazione: 9
Area: 27

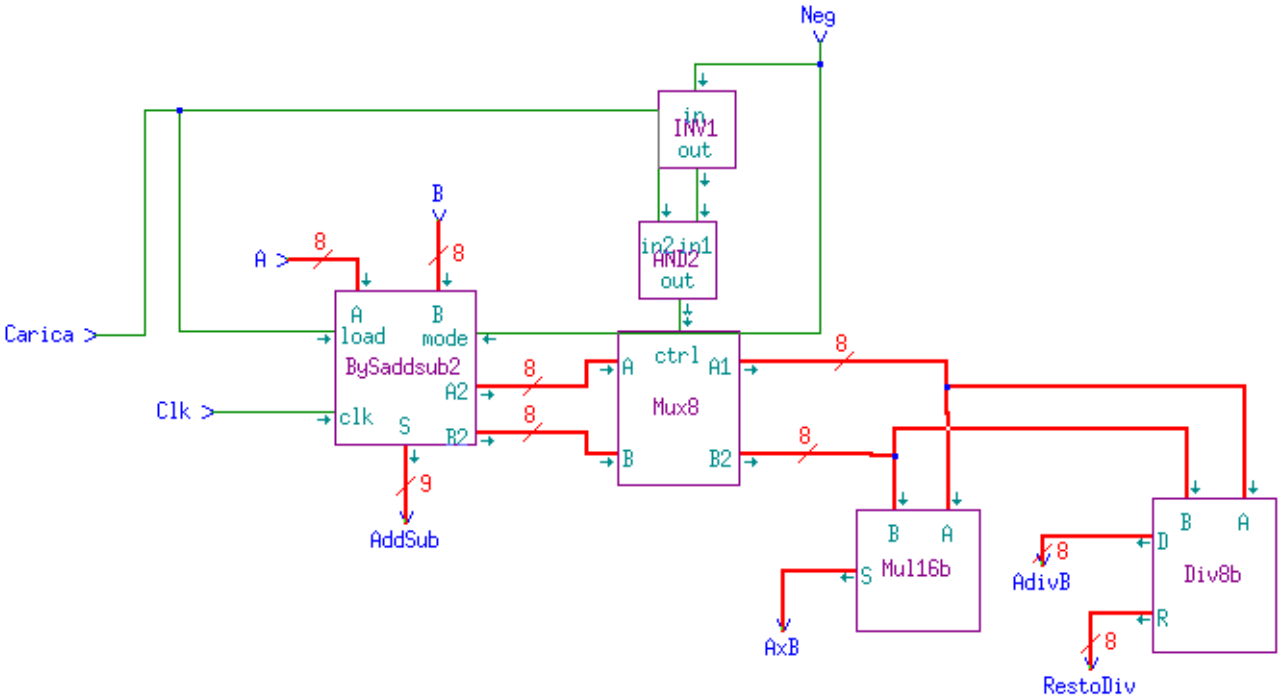


7. Grafica del progetto:

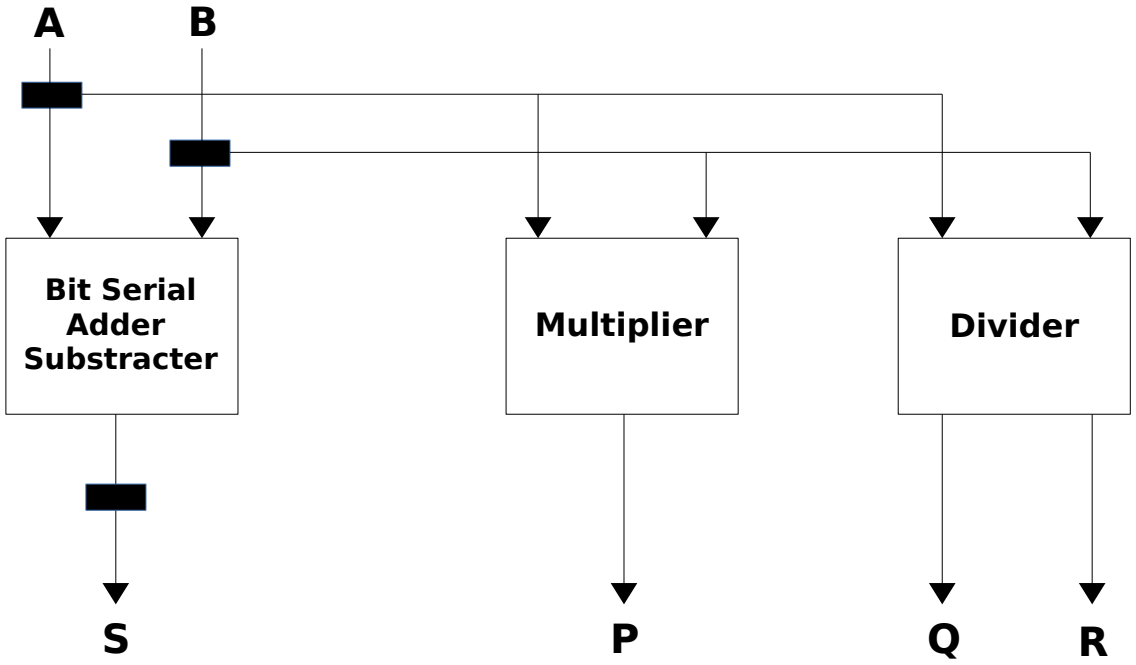
Main del progetto:



Struttura della calcolatrice:



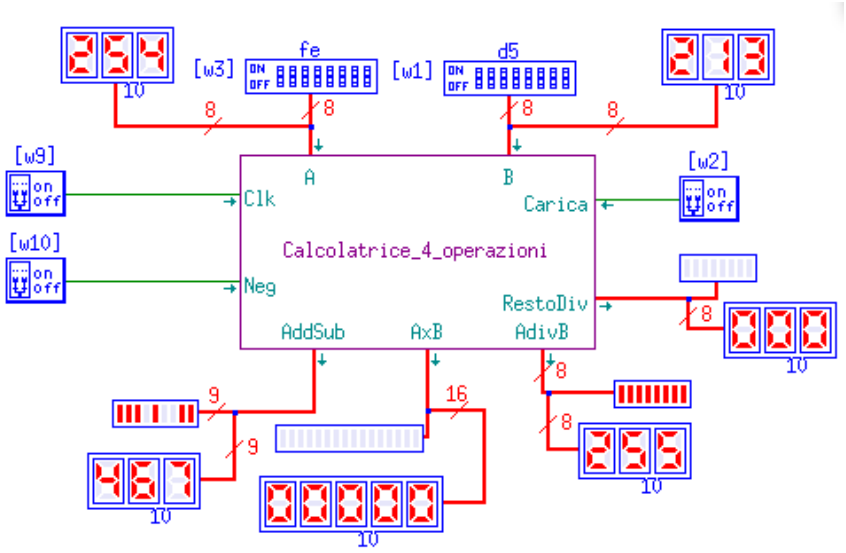
8. Data path:



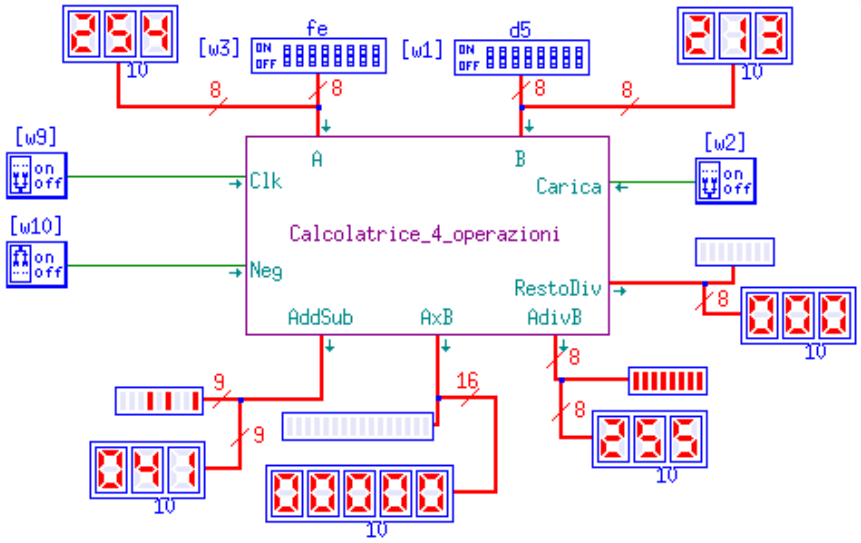
A, B = operandi;
S = somma o sottrazione (a seconda dell’operazione richiesta);
P = prodotto;
Q = quoziente;
R = resto della divisione;

9. Verifica e analisi funzionale:

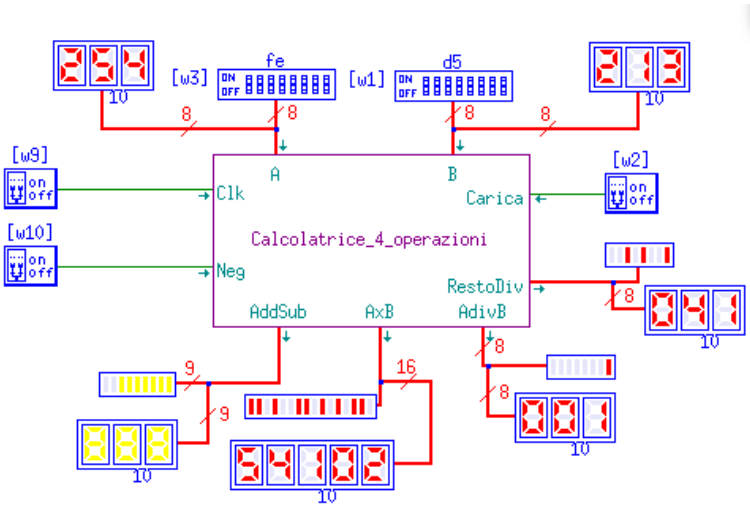
Somma:



Sottrazione:



Moltiplicazione e sottrazione:



Stime della complessità circuitale:

L’area dei diversi moduli è stata calcolata contando il numero di ingressi delle porte logiche elementari di ciascun componente.

	T. Contaminazione	T. Propagazione	Area
AND	2	2	3
INV	1	1	1
EXOR	2	3	8
MUX2	2	3	7
FFD	4	9	23
HA	2	3	11
FA	4	6	26
FS	6	9	27
MUX2x8	2	3	113
REG8sINaddsub	13	20	304
REG9sOUT	6	36	209
REG8sIN	8	14	216
BySaddsub2	12	62	709
Mul16b	2	30	1464
Div8b	7	88	2135

COMPLESSITÀ CIRCUITALE: 4425

Constatazioni:

In riferimento alle macro funzionali, divisore e moltiplicatore aumentano la complessità circuitale in maniera quadratica (per n bit degli operandi) rispetto all’area e al tempo di propagazione, mentre il tempo di contaminazione resta costante. La complessità del bit serial adder/substracter per quanto riguarda l’area ed il tempo di contaminazione resta costante (per n bit degli operandi) e il tempo di propagazione aumenta in maniera lineare.