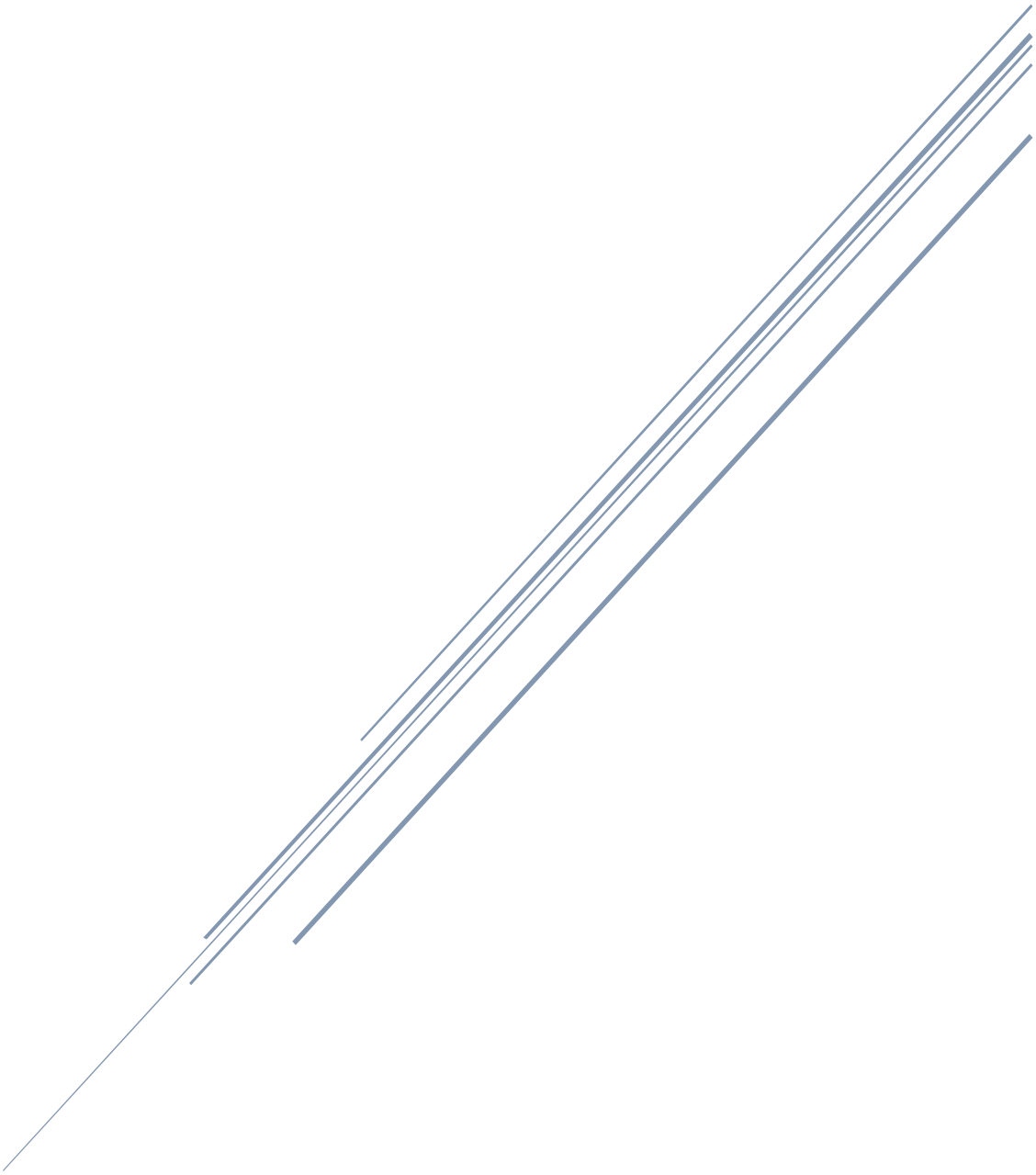
SOMMATORE IEEE 754

Implementazione di un sommatore floating point single precision



Politecnico di Milano, sede di Cremona
Beccari Giulio, Cuzzocrea Luca

Indice

1 Introduzione

1.1 Descrizione del progetto

Lo scopo di questo progetto è la realizzazione in linguaggio VHDL di un sommatore floating point secondo lo standard IEEE 754. Il sommatore è in grado di trattare operandi normalizzati, non normalizzati ed operandi speciali (NaN e ∞).

1.2 Formato IEEE 754

Lo standard IEEE 754 prevede che gli operandi (single precision) abbiano la seguente struttura:

- S: segno, 1 bit;
- G: esponente, 8 bit;
- T: mantissa, 23 bit.

A seconda dei valori di G e T, possiamo identificare i seguenti casi:

S	G	T	CASO	VALORE
-	11111111	00000000000000000000000	**Infinito**	± ∞
-	11111111	Diverso da tutti 0	**Not a number**	NaN
-	00000000	Qualunque valore	**Numeri piccoli**	$(-1)^S \times 2^{-127} \times (0 + 2^{-23} \times T)$
-	Altro	Qualunque valore	**Normale**	$(-1)^S \times 2^{G-127} \times (1 + 2^{-23} \times T)$

1.3 Somma

Il procedimento della somma tra 2 numeri floating point single precision è il seguente:

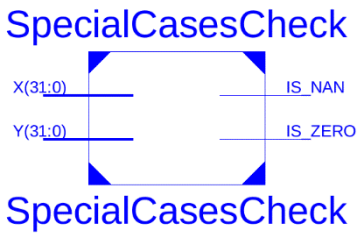
1. **Controllo dei casi particolari**: Ci sono alcuni casi particolari che non seguono il flusso normale della somma. Questi sono gli input che hanno come risultato: NaN, ± ∞ oppure 0.
2. **Controllo del maggiore e scambio degli operandi**: L'operando con valore assoluto maggiore deve essere messo a sinistra. In questo modo, le operazioni seguenti possono essere semplificate.
3. **Calcolo del segno finale e dell'operazione**: La somma/sottrazione viene eseguita sul valore assoluto degli operandi.
   Il segno finale sarà il segno dell'operatore più grande.
   L'operazione da fare sui valori assoluti sarà:
   - una somma, se gli operandi hanno segno concorde;
   - una sottrazione, se gli operatori hanno segno discorde.
4. **Preparazione delle mantisse**: Le mantisse hanno il primo bit rappresentato implicitamente nell'esponente. Questo va aggiunto esplicitamente al numero, portandolo su 24 bit. Il valore è 0 se G è tutto a 0, 1 nel caso normale.
   Le mantisse vanno poi rappresentate con la stessa potenza di 2 (quella dell'esponente più grande). Per fare questo, la mantissa del numero più piccolo viene a destra della differenza tra gli esponenti. Le mantisse devono anche essere estese a 48 bit per evitare perdite di informazione.
5. **Somma/Sottrazione mantisse**: Le mantisse estese possono essere sommate o sottratte usando gli input generati nelle operazioni precedenti. Il risultato è a 48 bit + 1 bit di overflow.
6. **Normalizzazione del risultato**: A questo punto abbiamo il segno del risultato, il suo esponente (che equivale all'esponente dell'operando più grande) e la mantissa su 48 bit (+ overflow).
   Il segno è quello calcolato precedentemente.
   L'esponente va diminuito in base al numero di zeri iniziali della mantissa.

La mantissa va shiftata in modo da rimuovere gli zeri iniziali e il primo 1. (Attenzione: Se l'esponente finale è il minore possibile non vanno rimossi tutti gli zeri, ma sono quanti ne bastano per portare l'esponente a –127). Poi si scartano i bit meno significativi e si mantengono solo i primi 23 bit.
In questo modo si ottiene il risultato della somma in formato IEEE754 a 32 bit.

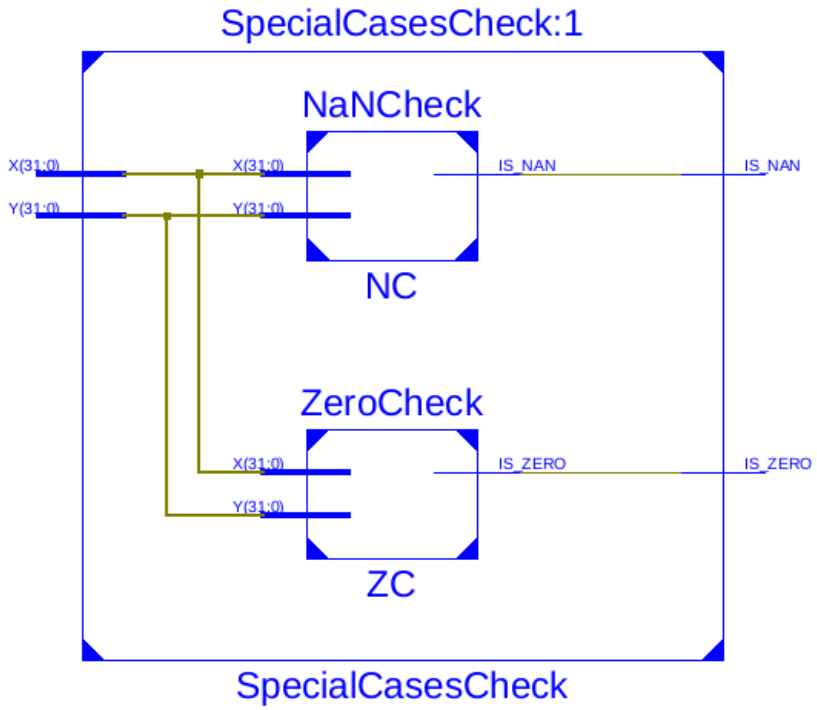
2 Moduli

2.1 SpecialCasesCheck

Il modulo SpecialCasesCheck riceve in input i due numeri da sommare/sottrarre, X e Y, e si occupa di verificare, attraverso i moduli NaNCheck e ZeroCheck, se questi danno origine ad un risultato speciale restituendo in uscita due flag IS_NAN e IS_ZERO.

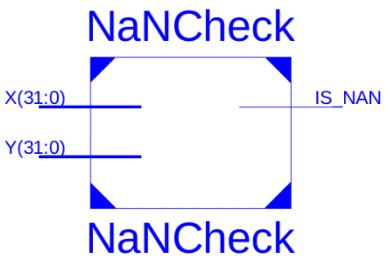


SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	32 bit	Primo operando della somma/sottrazione.
Y	in	32 bit	Secondo operando della somma/sottrazione.
IS_NAN	out	1 bit	Flag: vale ‘1’ se X e Y generano NaN.
IS_ZERO	out	1 bit	Flag: vale ‘1’ se X e Y generano ‘0’.

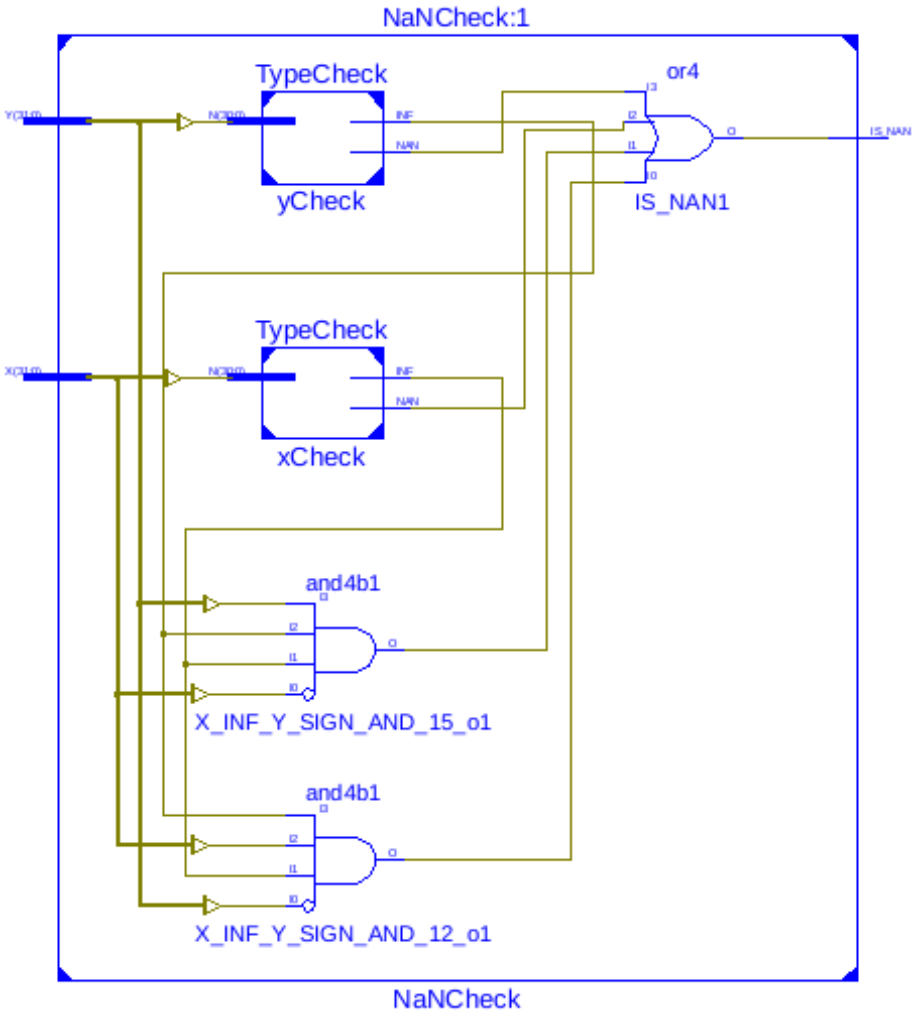


2.1.1 NaNCheck

NaNCheck viene utilizzato dal modulo SpecialCasesCheck per verificare se gli operandi della somma/sottrazione danno origine al valore NaN. Per effettuare questo controllo si serve di TypeCheck.

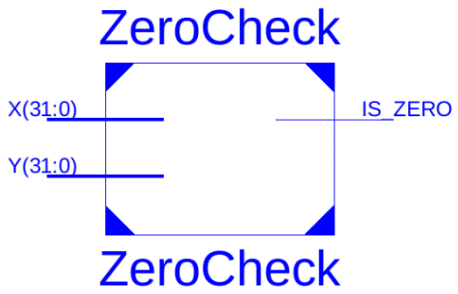


SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	32 bit	Primo operando della somma/sottrazione.
Y	in	32 bit	Secondo operando della somma/sottrazione.
IS_NAN	out	1 bit	Flag: vale ‘1’ se X e Y generano NaN.

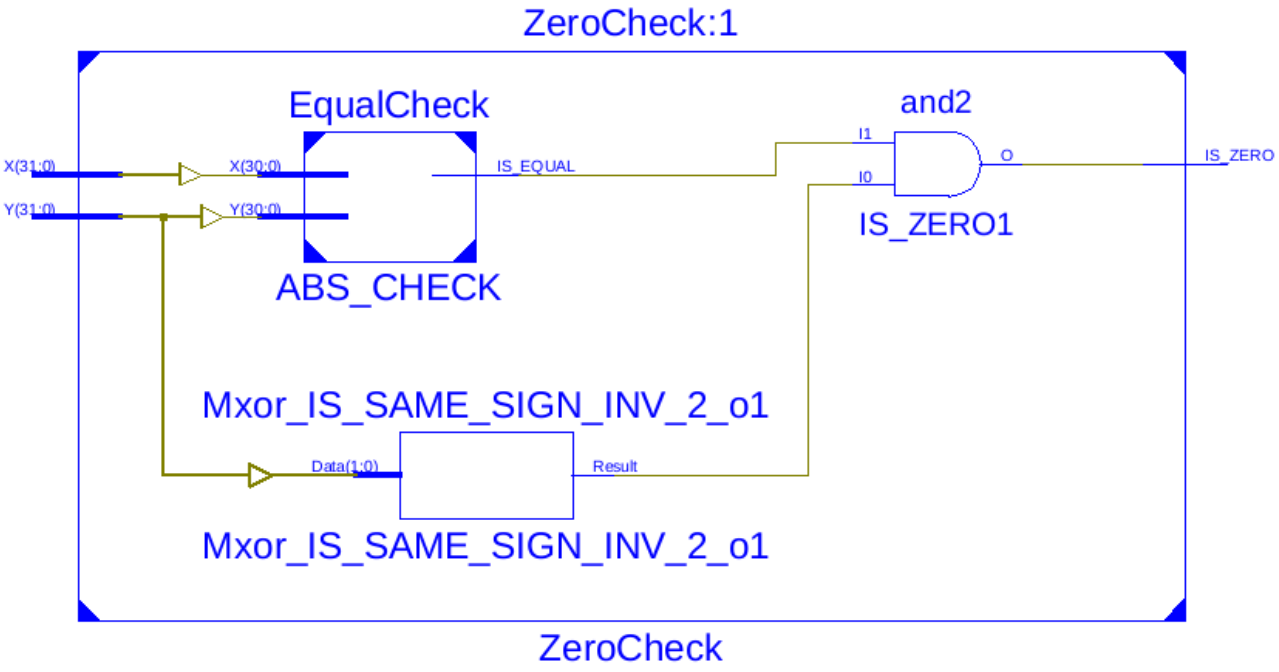


2.1.2 ZeroCheck

ZeroCheck è anch'esso impiegato dal modulo SpecialCasesCheck e verifica se il risultato della somma è '0'.

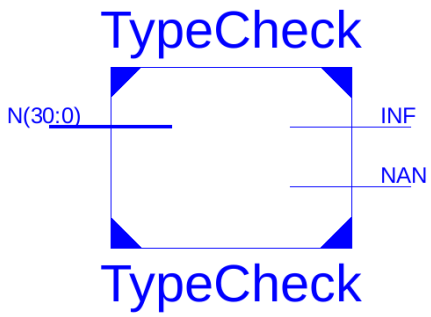


SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	32 bit	Primo operando della somma/sottrazione.
Y	in	32 bit	Secondo operando della somma/sottrazione.
IS_ZERO	out	1 bit	Flag: vale '1' se X e Y generano '0'.



2.1.3 TypeCheck

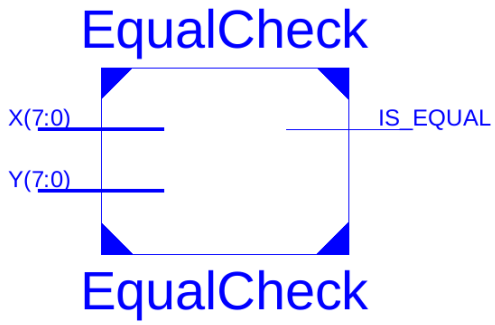
TypeCheck prende in input un numero N in 32 bit, controlla esponente e mantissa per verificare se N è NaN o ∞. Restituisce due flag per indicare se il valore di N è uno di questi.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
N	in	32 bit	Numero da controllare.
NAN	out	1 bit	Flag: vale ‘1’ se il numero è NaN.
INF	out	1 bit	Flag: vale ‘1’ se il numero è ∞.

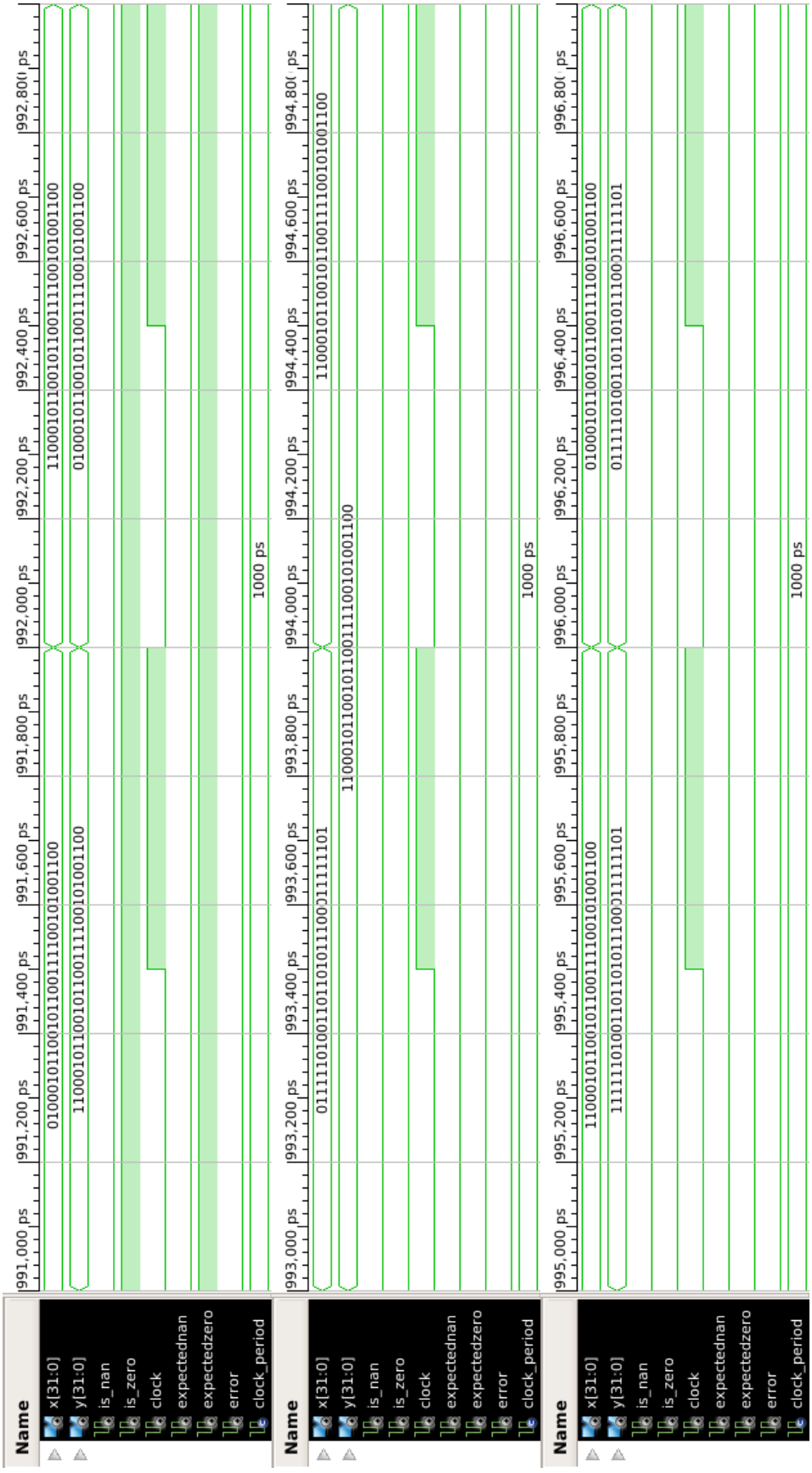
2.1.4 EqualCheck

Il modulo EqualCheck si occupa di verificare se i valori assoluti dei due operandi, X e Y, sono uguali a meno del segno.



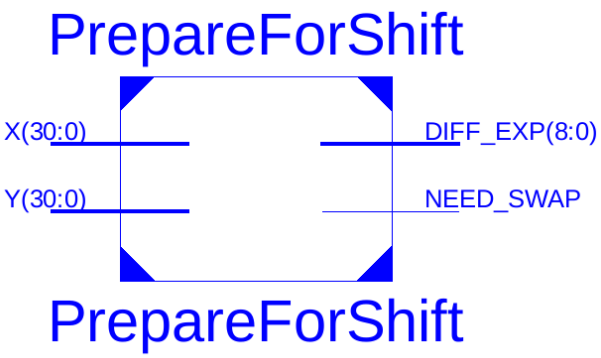
SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	31 bit	Valore assoluto del primo operando.
Y	in	31 bit	Valore assoluto del secondo operando.
IS_EQUAL	out	1 bit	Flag: vale ‘1’ se X = Y.

2.1.5 Test bench

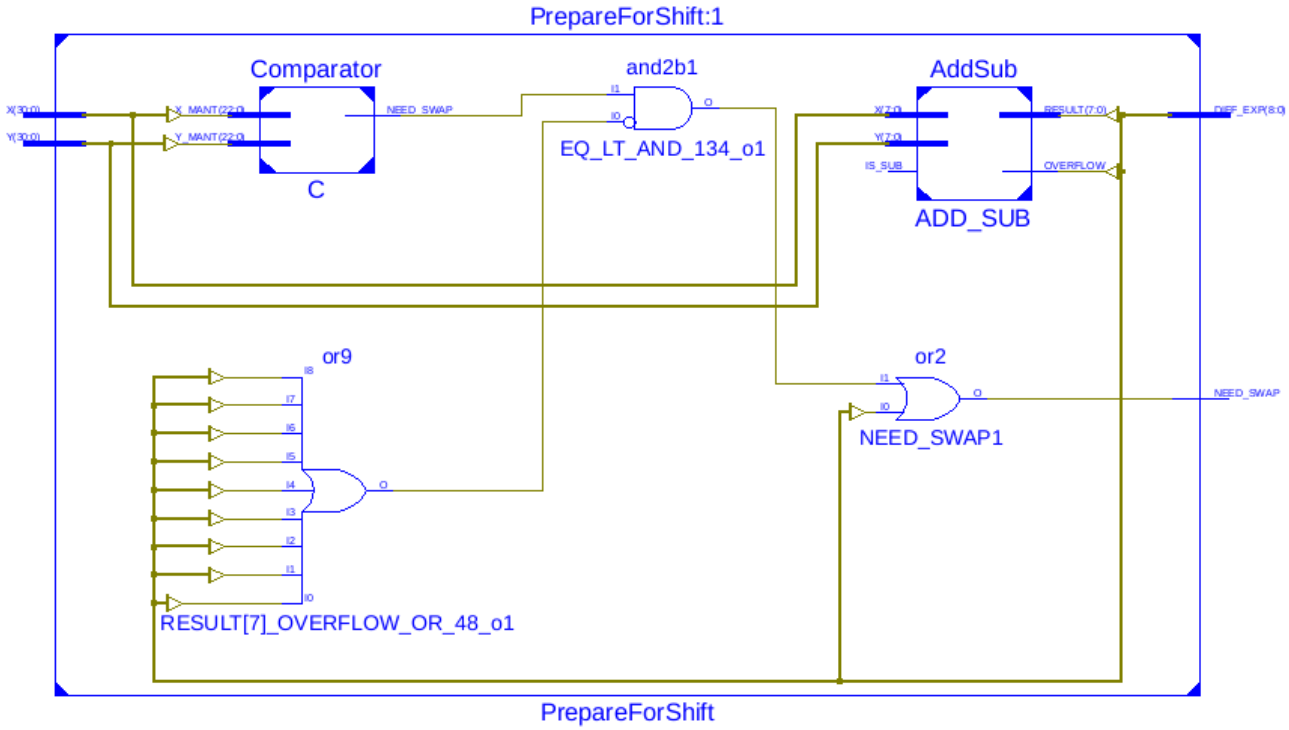


2.2 PrepareForShift

PrepareForshift ha il compito di calcolare la differenza tra gli esponenti di X e Y, necessaria per fare lo shift della mantissa dell'addendo inferiore. Inoltre verifica se Y è maggiore di X e in tal caso abilita il segnale SW per fare lo swap dei due numeri (per convenzione X è l'addendo maggiore).

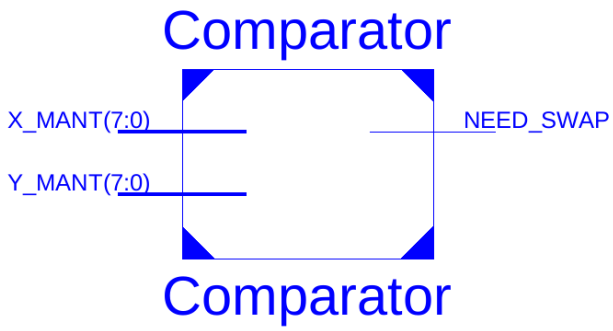


SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	31 bit	Valore assoluto del primo operando.
Y	in	31 bit	Valore assoluto del secondo operando.
DIFF_EXP	out	9 bit	Differenza tra esponenti di X e Y in complemento a 2.
NEED_SWAP	out	1 bit	Flag: vale '1' se Y > X.



2.2.1 Comparator

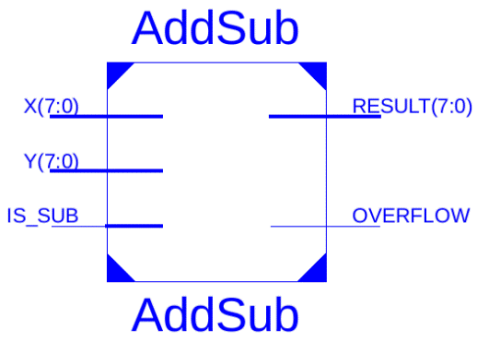
Comparator confronta le mantisse di X e Y per trovare la maggiore delle due in previsione dello swap. La logica nel modulo PrepareForShift si occuperà poi di controllare anche gli esponenti per capire se lo swap dovrà essere effettuato oppure no.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X_MANT	in	generic	Mantissa di X.
Y_MANT	in	generic	Mantissa di Y.
NEED_SWAP	out	1 bit	Flag: vale ‘1’ se Y_MANT > X_MANT.

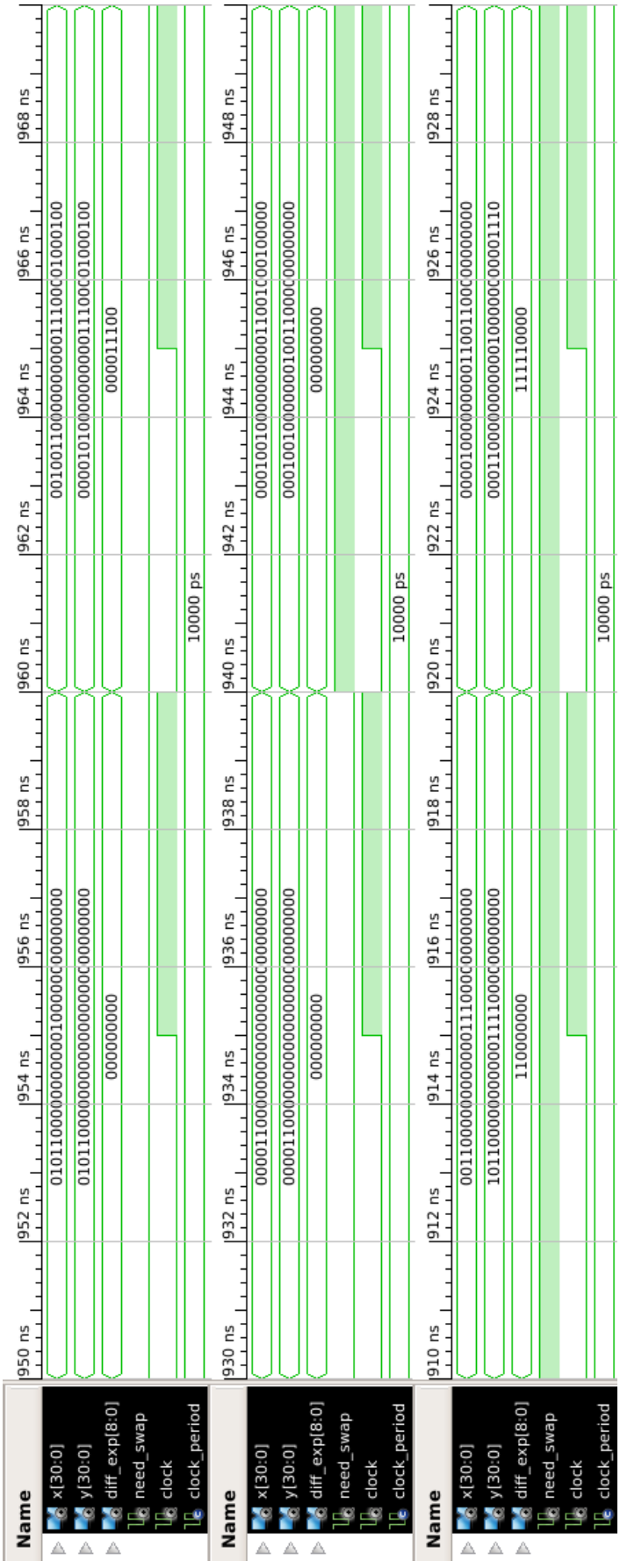
2.2.2 AddSub

AddSub effettua la somma/sottrazione degli ingressi X e Y. Il flag IS_SUB segnala al modulo l’operazione da effettuare. In output troviamo il risultato (RESULT) più l’eventuale overflow (OVERFLOW) del sommatore/sottrattore.



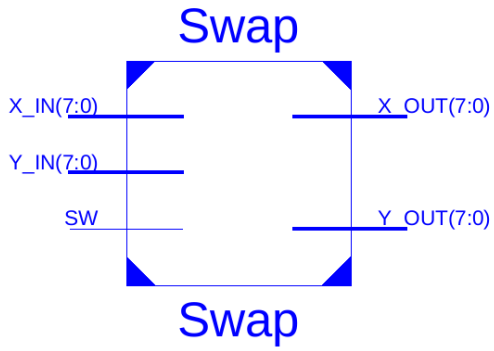
SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	generic	Primo operando.
Y	in	generic	Secondo operando.
IS_SUB	in	1 bit	Flag: vale ‘1’ se l’operazione da effettuare è la differenza.
RESULT	out	generic	Risultato dell’operazione.
OVERFLOW	out	1 bit	Bit di overflow dell’operazione.

2.2.3 Test bench

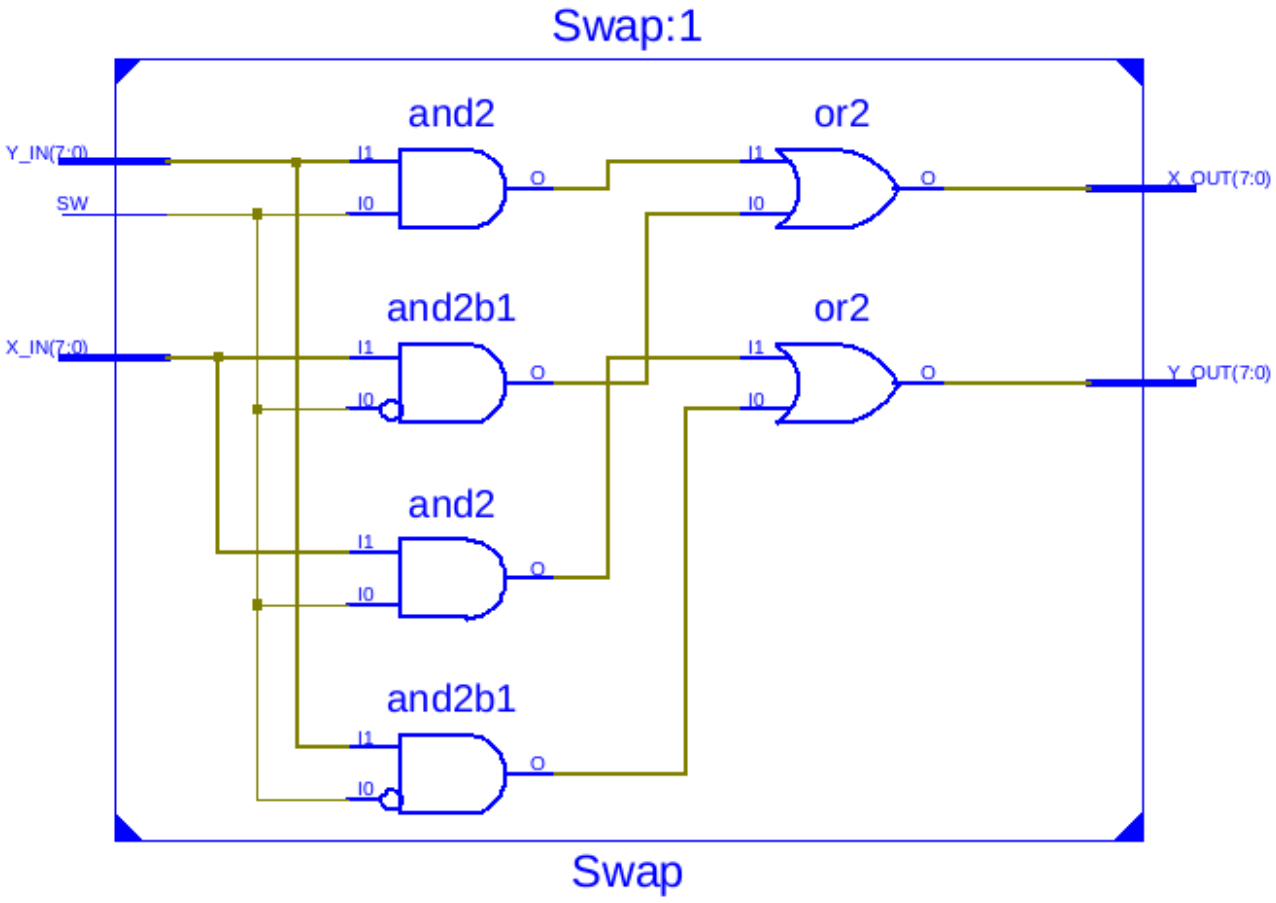


2.3 Swap

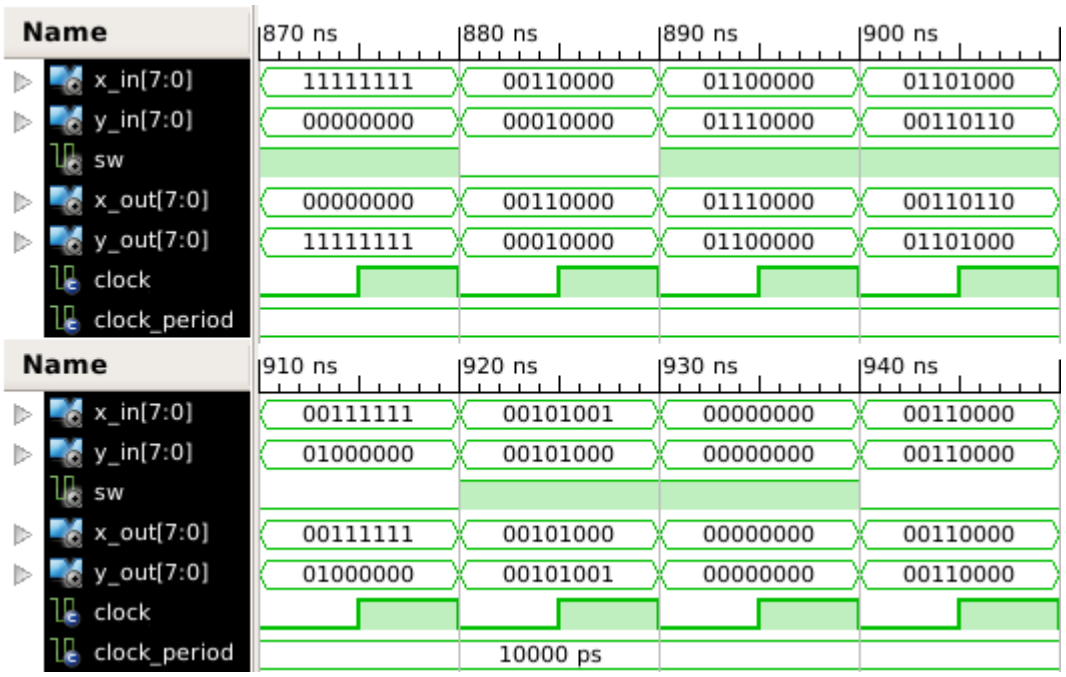
Il modulo Swap riceve in ingresso gli addendi X e Y per effettuare, se necessario, lo swap tra i due.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X_IN	in	generic	Primo operando.
Y_IN	in	generic	Secondo operando.
SW	in	1 bit	Flag: vale ‘1’ se deve essere effettuato lo swap.
X_OUT	out	generic	Nuovo primo operando.
Y_OUT	out	generic	Nuovo secondo operando.

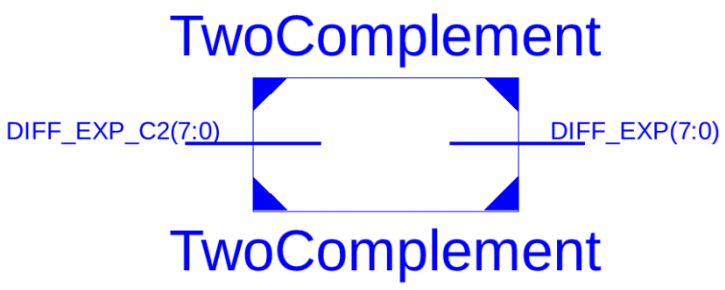


2.3.1 Test bench



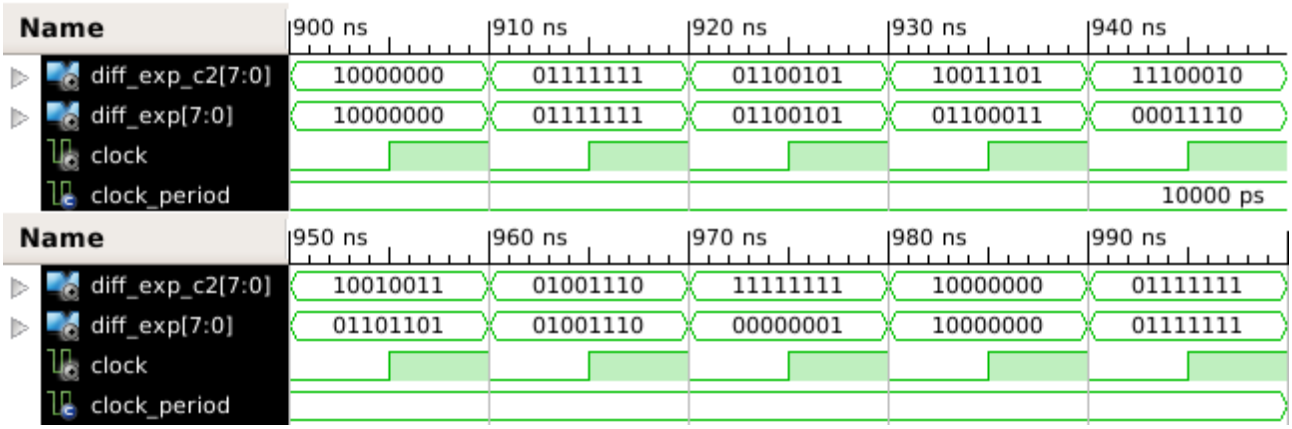
2.4 TwoComplement

TwoComplement riceve in ingresso un numero in complemento a 2 e ne restituisce il valore assoluto in codifica binaria naturale.



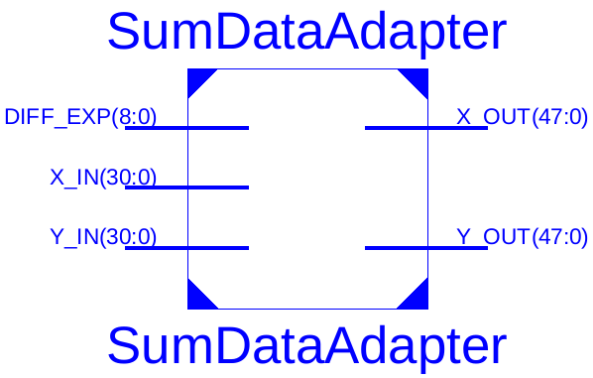
SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
DIFF_EXP_C2	in	generic	Numero in complemento a 2.
DIFF_EXP	out	generic	Valore assoluto del numero in input, in codifica binaria naturale.

2.4.1 Test bench

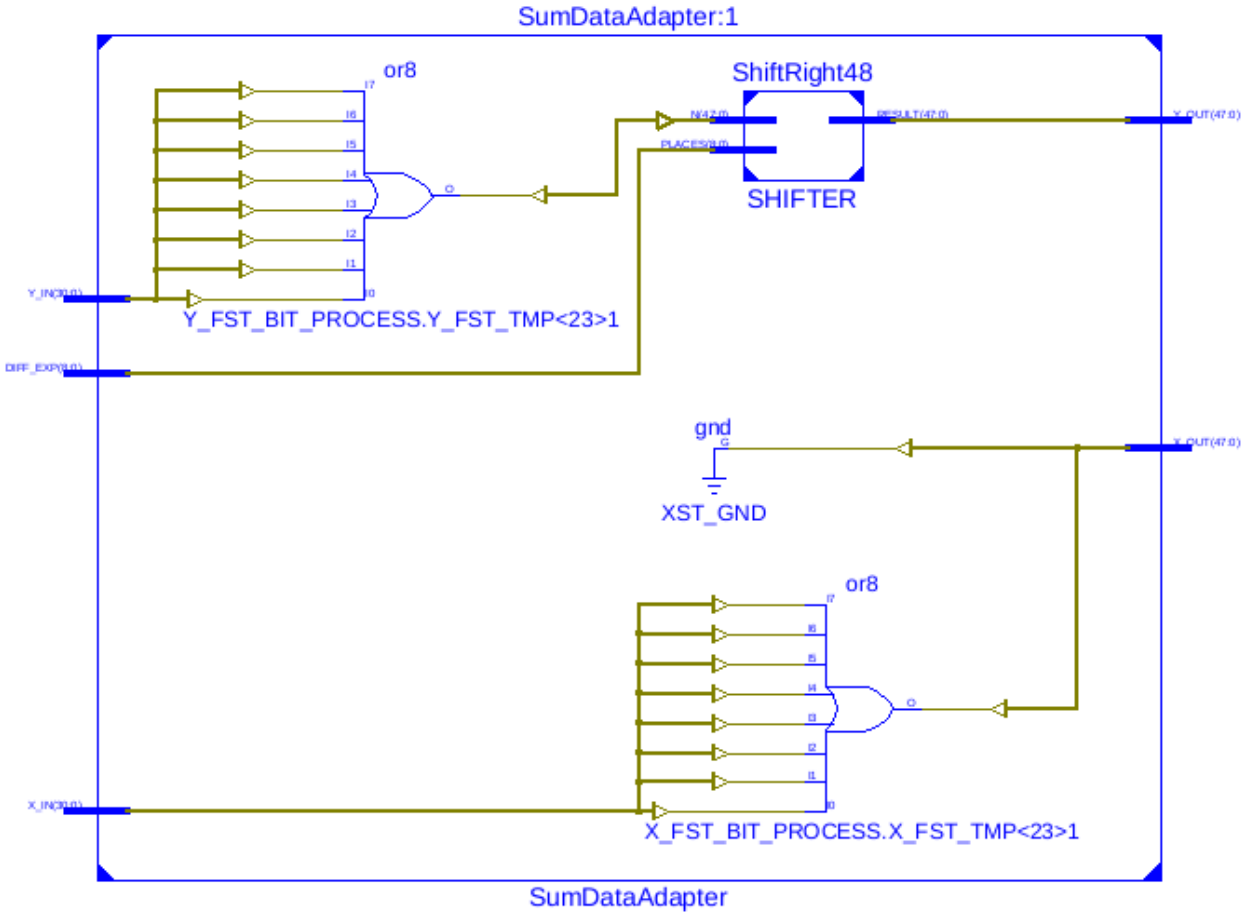


2.5 SumDataAdapter

SumDataAdapter si occupa di preparare gli operandi X e Y per la somma/sottrazione. Ricevuti in ingresso X e Y dal modulo Swap, calcola, esaminando gli esponenti, il bit che precede la mantissa (0 o 1) attraverso una OR a 8 bit. Successivamente estende le mantisse da 23 a 47 bit e aggiunge in testa il bit calcolato precedentemente. Infine, attraverso il modulo ShiftRight48, effettua lo shift della mantissa di Y di un numero di posizioni pari a DIFF_EXP (ricevuto da TwoComplement).

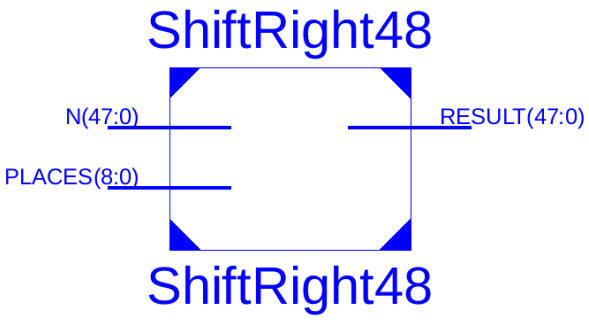


SEGANLE	TIPO	LUNGHEZZA	DESCRIZIONE
X_IN	in	31 bit	Valore assoluto del primo operando.
Y_IN	in	31 bit	Valore assoluto del secondo operando.
DIFF_EXP	in	9 bit	Numero di posizioni per lo shift della mantissa di Y.
X_OUT	out	48 bit	Mantissa estesa di X.
Y_OUT	out	48 bit	Mantissa estesa di Y.



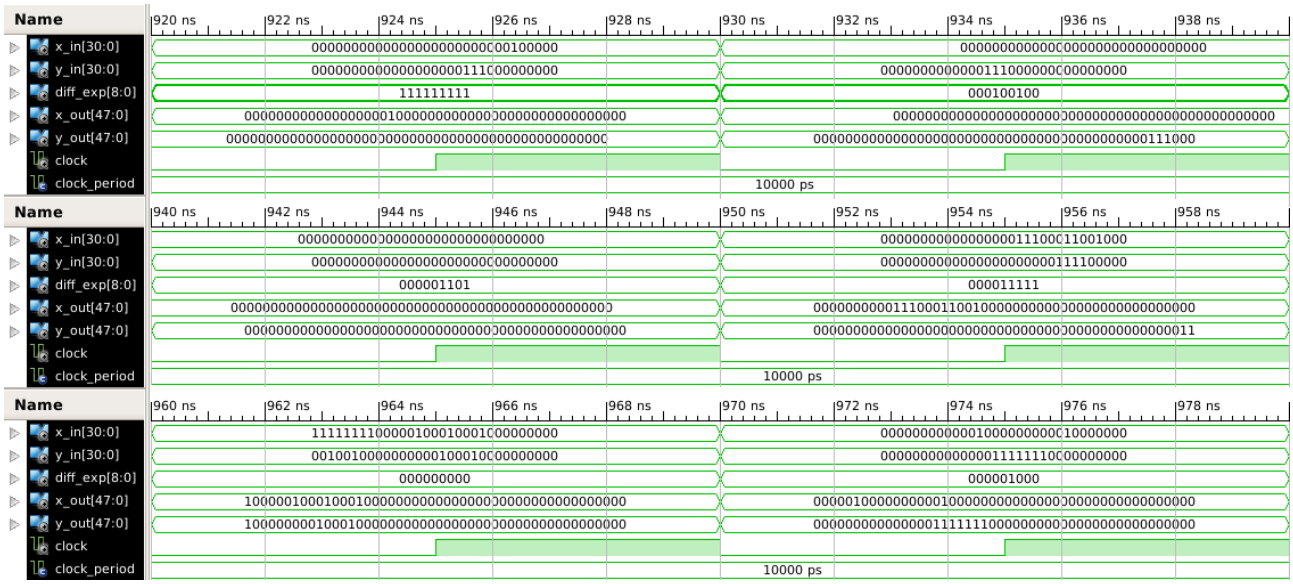
2.5.1 ShiftRight48

ShiftRight48 riceve in ingresso un numero a 48 bit e ne effettua lo shift di un numero di posizioni pari a PLACES.



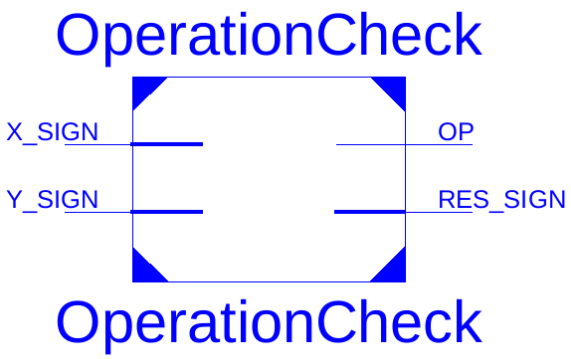
SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
N	in	48 bit	Numero di cui fare lo shift.
PLACES	in	9 bit	Numero di posizioni per lo shift di N.
RESULT	out	48 bit	Risultato dello shift.

2.5.2 Test bench

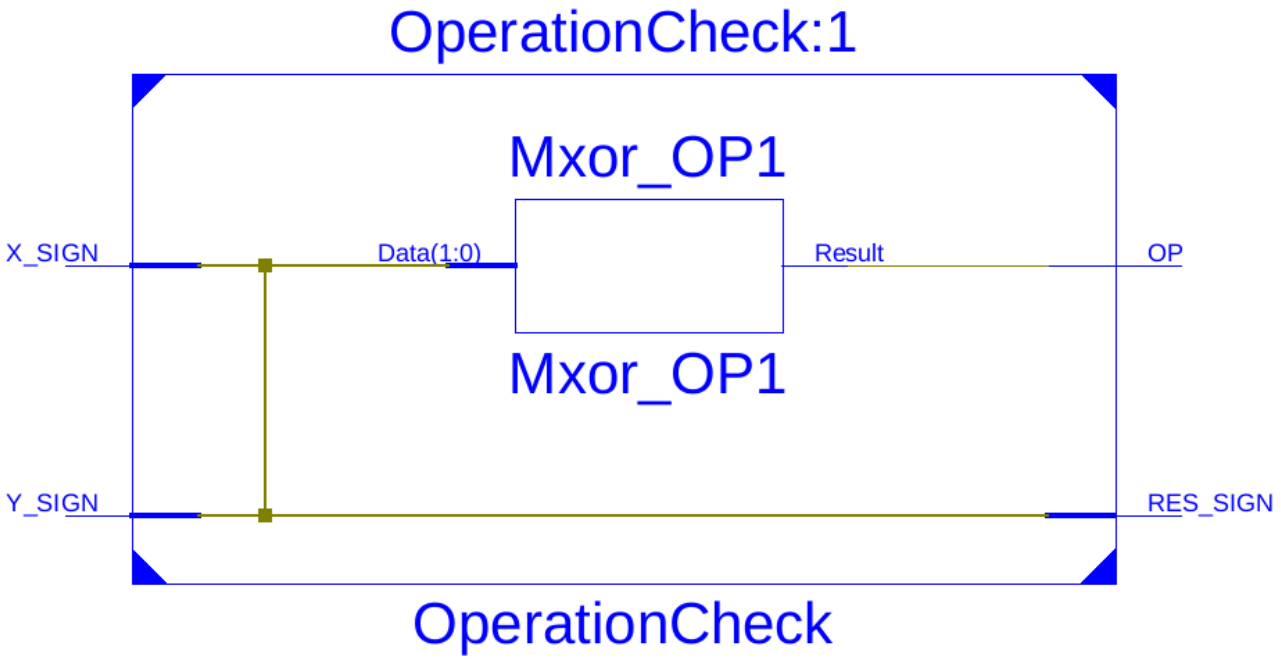


2.6 OperationCheck

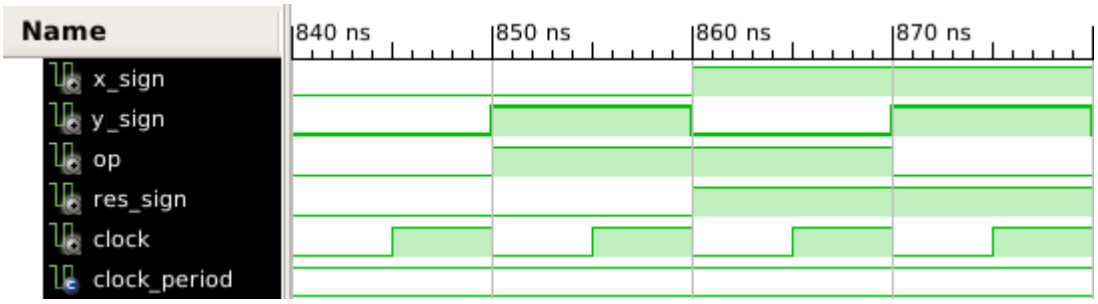
OperationCheck calcola, a partire dai segni dei due operandi, il segno del risultato finale e l'operazione da svolgere (somma o sottrazione).



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X_SIGN	in	1 bit	Segno del primo operando.
Y_SIGN	in	1 bit	Segno del secondo operando.
OP	out	1 bit	Flag: vale '1' se deve essere effettuata la differenza.
RES_SIGN	out	1 bit	Segno del risultato finale.

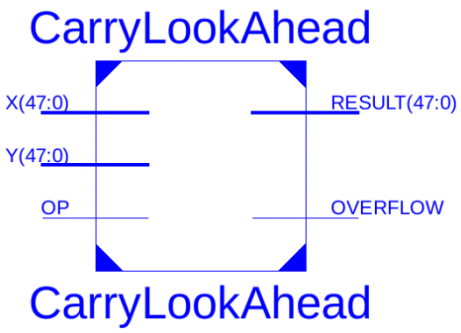


2.6.1 Test bench

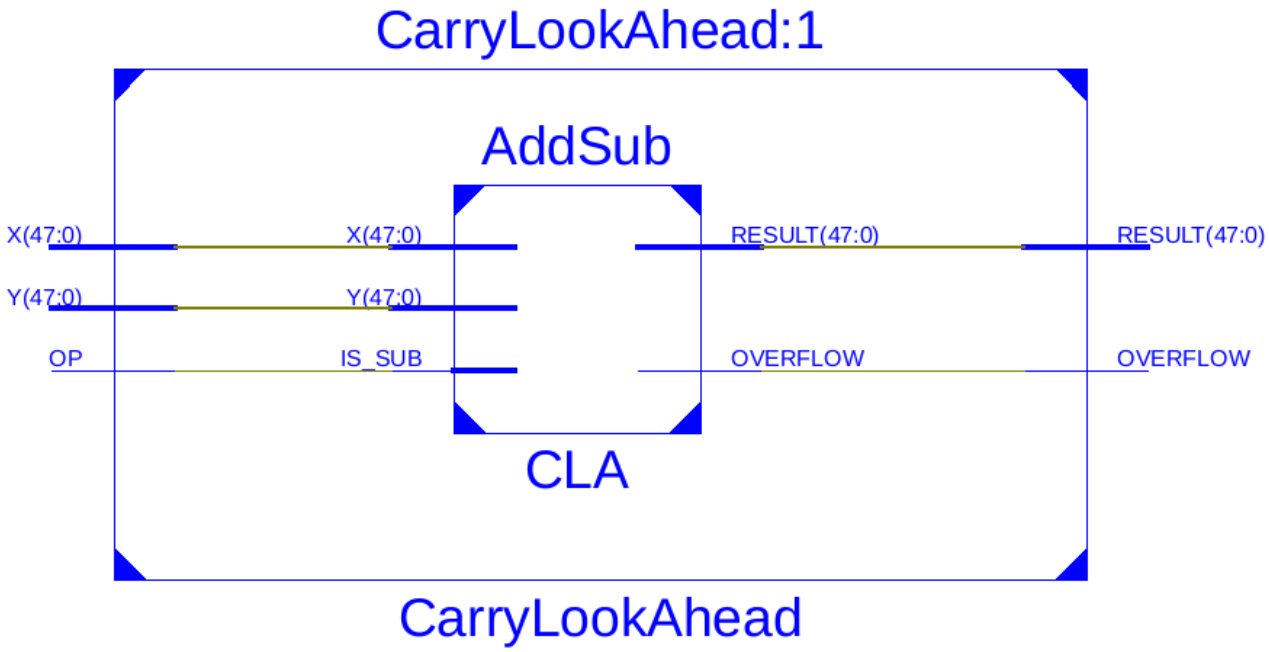


2.7 CarryLookAhead

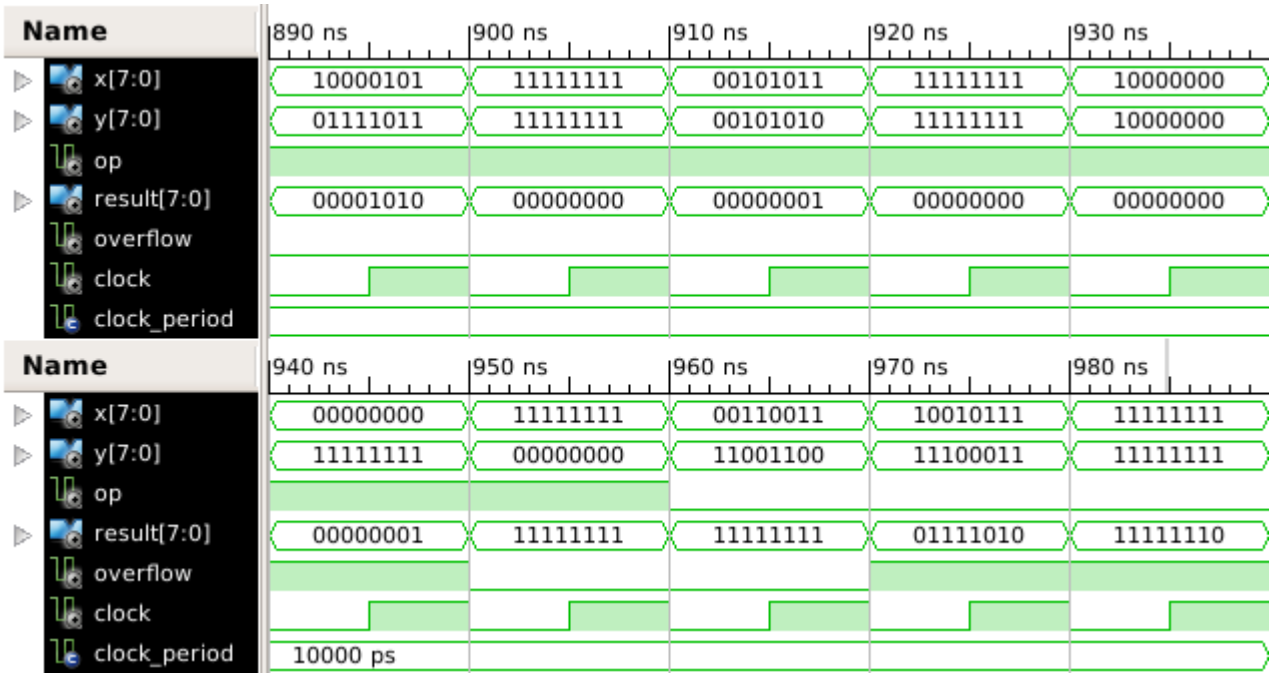
CarryLookAhead riceve le mantisse estese X e Y da SumDataAdapter e in base al valore del flag OP, ricevuto da OperationCheck, ne fa la somma o la differenza.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	48 bit	Mantissa estesa del primo operando.
Y	in	48 bit	Mantissa estesa del secondo operando.
OP	in	1 bit	Flag: vale ‘1’ se deve essere effettuata la differenza.
RESULT	out	48 bit	Risultato della somma/differenza.
OVERFLOW	out	1 bit	Bit di overflow della somma/differenza.

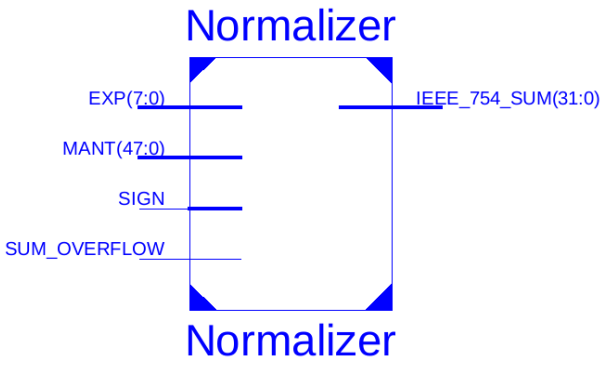


2.7.1 Test bench



2.8 Normalizer

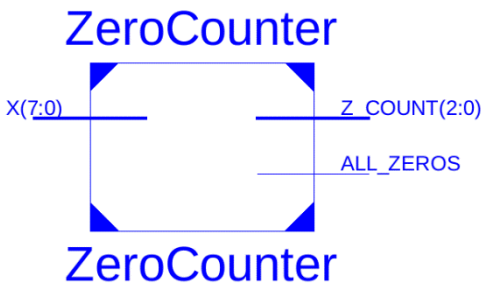
Normalizer è il modulo che si occupa di generare il numero IEEE754 a 32 bit che rappresenta la somma degli operandi.
Si occupa di adattare l'esponente, aumentandolo o diminuendolo. Poi normalizza il risultato della somma/sottrazione (MANT) ricevuto da CarryLookAhead rimuovendo gli zeri iniziali.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
SIGN	in	1 bit	Segno del risultato della somma/sottrazione.
EXP	in	8 bit	Esponente dell'ingresso più grande.
MANT	in	48 bit	Mantissa del risultato della somma/sottrazione.
SUM_OVERFLOW	in	1 bit	Bit di overflow della somma delle mantisse.
IEEE_754_SUM	out	32 bit	Somma dei due operandi X e Y in formato IEEE 754.

2.8.1 ZeroCounter

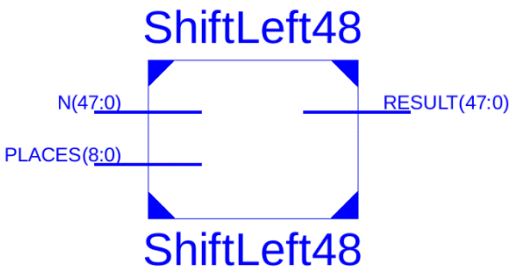
ZeroCounter conta il numero di '0' che precedono il primo '1' presenti nell'input X.



SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
X	in	generic	Numero su cui effettuare il conteggio.
Z_COUNT	out	⌈ log2 (generic) ⌉	Numero di '0' che precedono il primo '1'.
ALL_ZEROS	out	1 bit	Flag: vale '1' se X non contiene '1'.

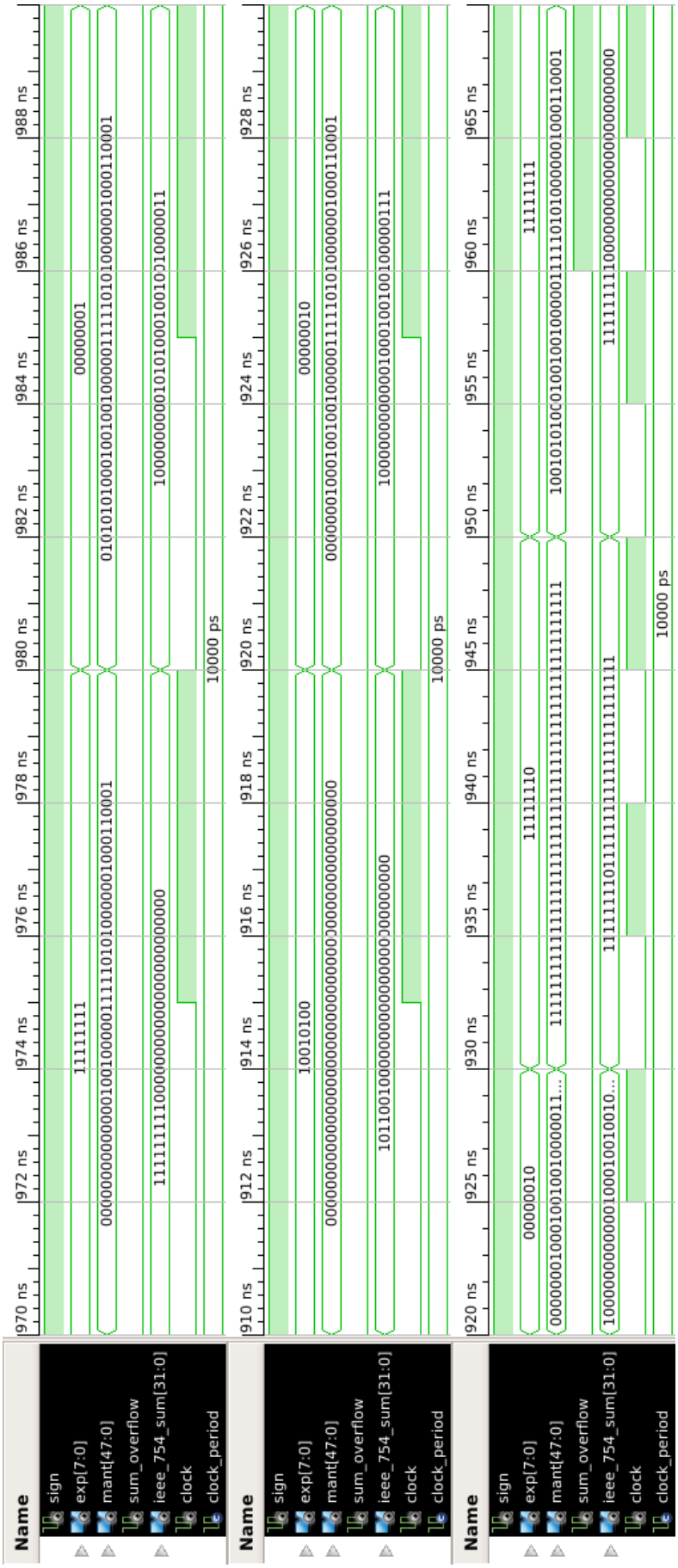
2.8.2 ShiftLeft48

ShiftLeft48 effettua lo shift a sinistra di N, di un numero di posizioni indicato da PLACES.



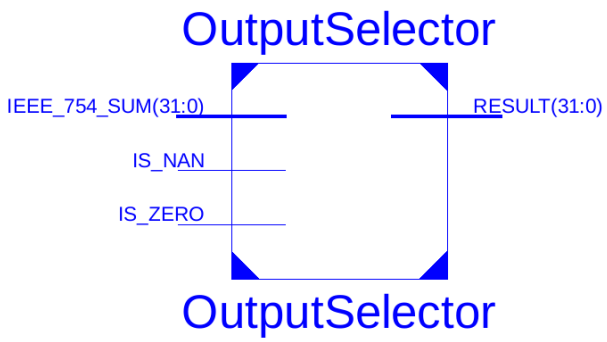
SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
N	in	48 bit	Numero di cui fare lo shift.
PLACES	in	9 bit	Numero di posizioni per lo shift di N.
RESULT	out	48 bit	Numero dopo lo shift.

2.8.3 Test bench



2.9 OutputSelector

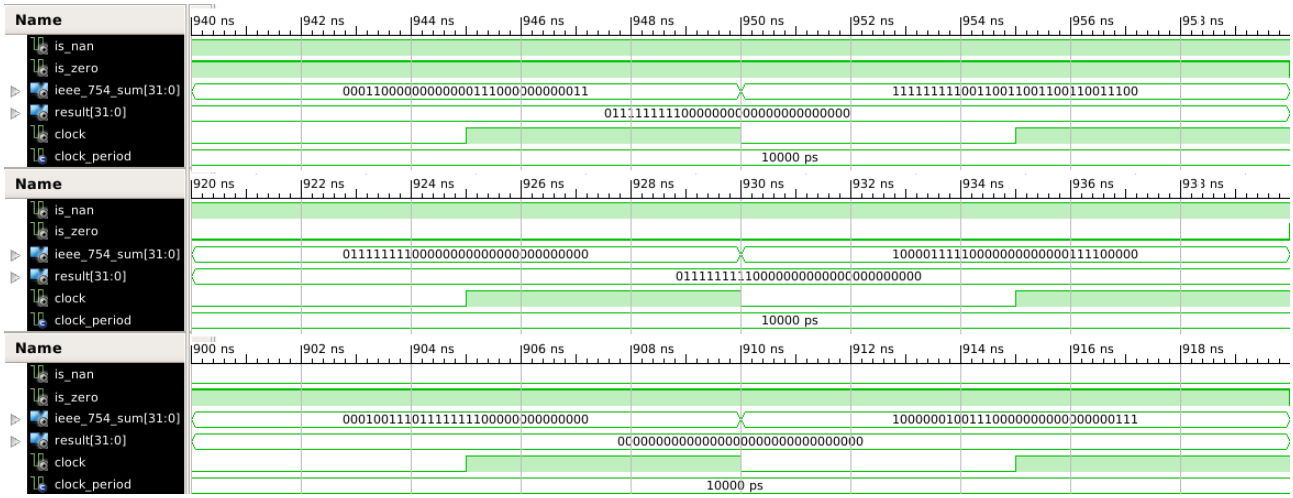
OutputSelector riceve da SpecialCasesCheck i flag IS_NAN, IS_ZERO e da Normalizer il risultato della somma/sottrazione normalizzato (IEEE_754_SUM). Quindi seleziona il risultato finale:



RISULTATO	VALORE
NAN	0 11111111 10000000000000000000000
ZERO	0 00000000 00000000000000000000000
SOMMA CALCOLATA	IEEE_754_SUM

SEGNALE	TIPO	LUNGHEZZA	DESCRIZIONE
IS_NAN	in	1 bit	Flag: vale ‘1’ se il risultato deve essere NaN.
IS_ZERO	in	1 bit	Flag: vale ‘1’ se il risultato deve essere zero.
IEEE_754_SUM	in	32 bit	Risultato della somma/sottrazione.
RESULT	out	32 bit	Risultato finale.

2.9.1 Test bench



3 Pipeline

3.1 Analisi dei ritardi combinatori

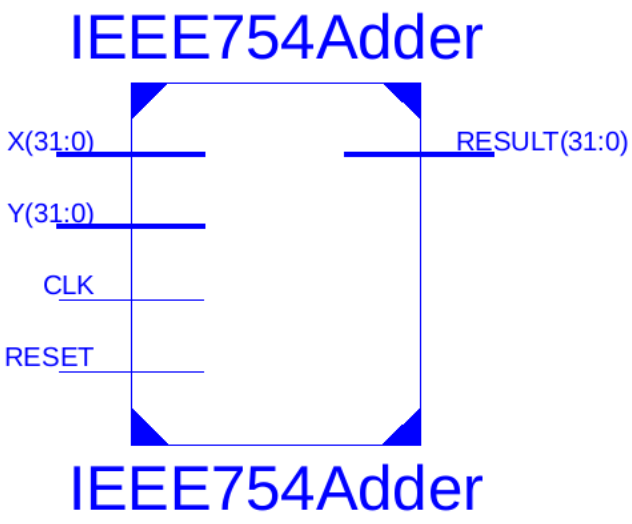
I report di sintesi dei diversi moduli hanno evidenziato i seguenti ritardi:

MODULO	RITARDO
SPECIAL CASES CHECK	7,57 ns
PREPARE FOR SHIFT	11,894 ns
SWAP	5,810 ns
TWO COMPLEMENT	6,406 ns
OPERATION CHECK	5,296 ns
SUM DATA ADAPTER	9,801 ns
CARRY LOOK AHEAD	27,242 ns
NORMALIZER	28,233 ns
OUTPUT SELECTOR	5,971 ns

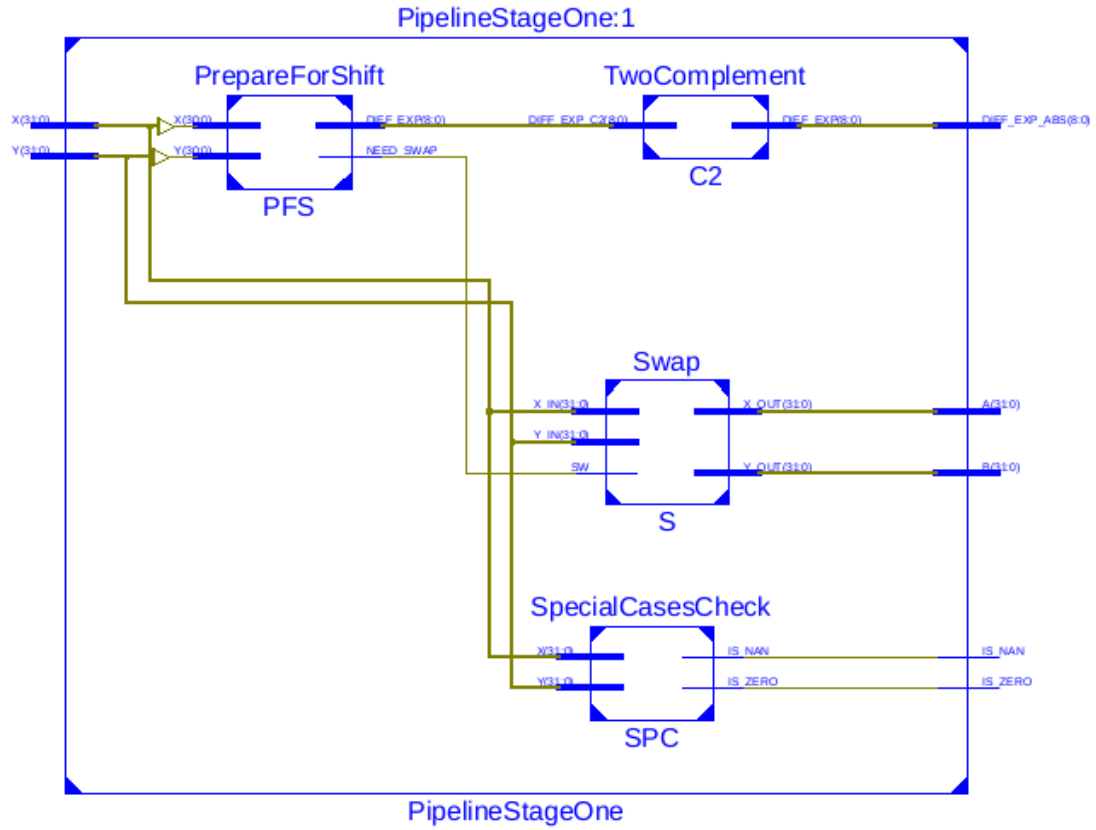
Dall'analisi della tabella sopra riportata emerge che il ritardo più consistente è pari a 28,233 ns, quindi si è deciso di strutturare la pipeline in tre stage così composti:

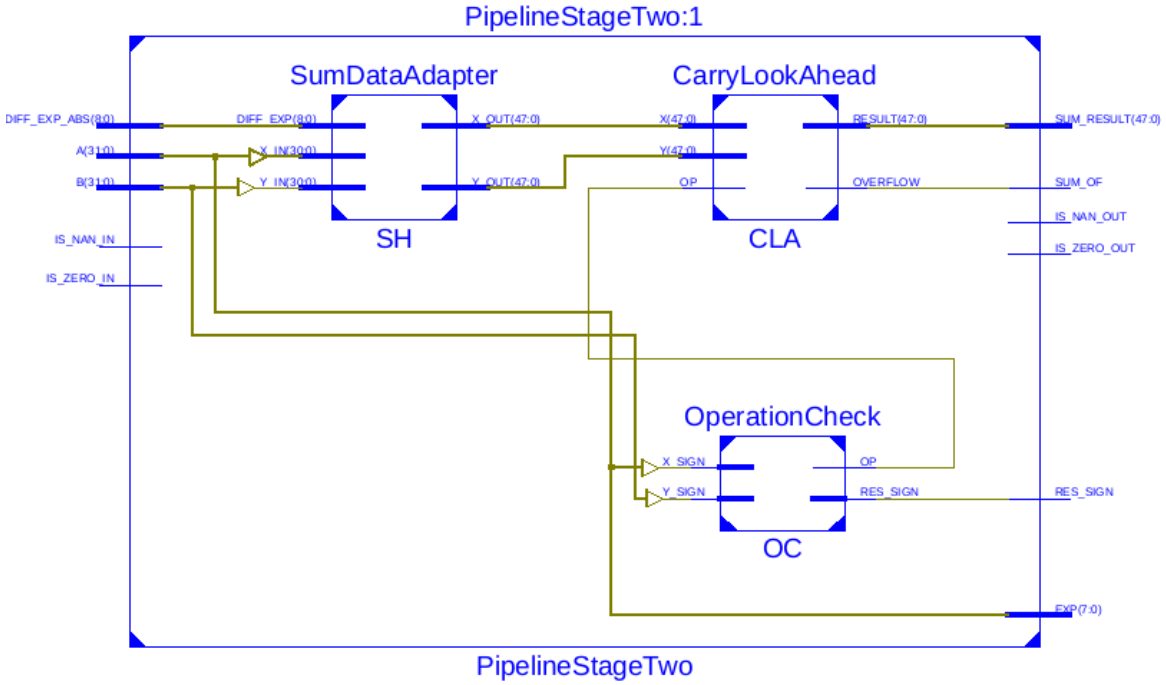
STAGE	MODULI	RITARDO
PIPELINE STAGE ONE	• SpecialCasesCheck; • OperationCheck; • PrepareForShift; • Swap; • TwoComplement.	14,095 ns
PIPELINE STAGE TWO	• SumDataAdapter; • CarryLookAhead.	24,421 ns
PIPELINE STAGE THREE	• Normalizer; • OutputSelector.	27,942 ns

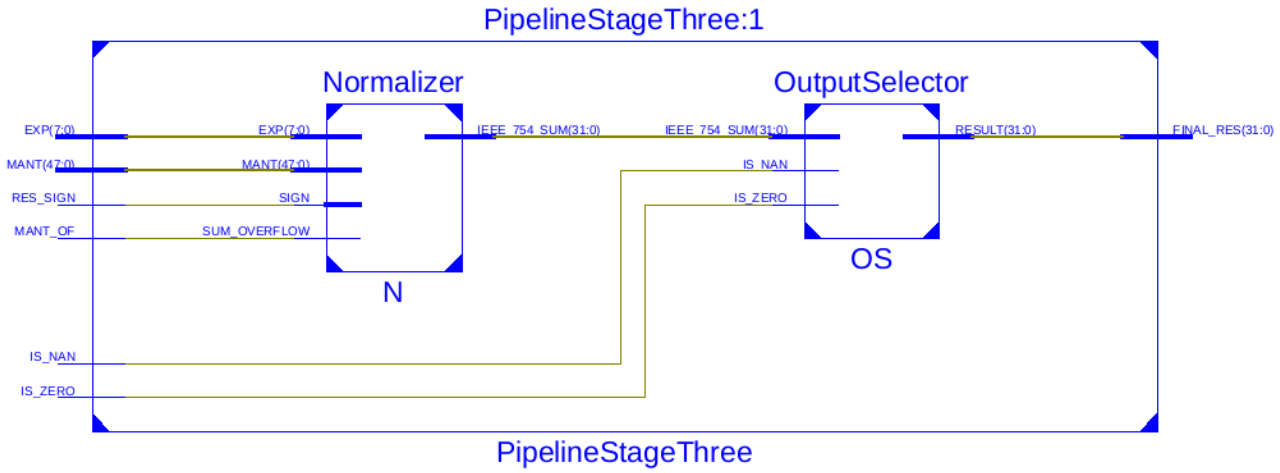
In risposta all'ottimizzazione nella sintesi dei tre stage sopra descritti, emerge che il ritardo maggiore è pari 27,942 ns (per giunta inferiore a quello del solo Normalizer!), mentre il ritardo introdotto dai Flip-Flop di tipo D risulta essere pari a 3,597 ns.
Si è quindi scelto un periodo di clock pari a 40 ns.



3.2 RTL Stage

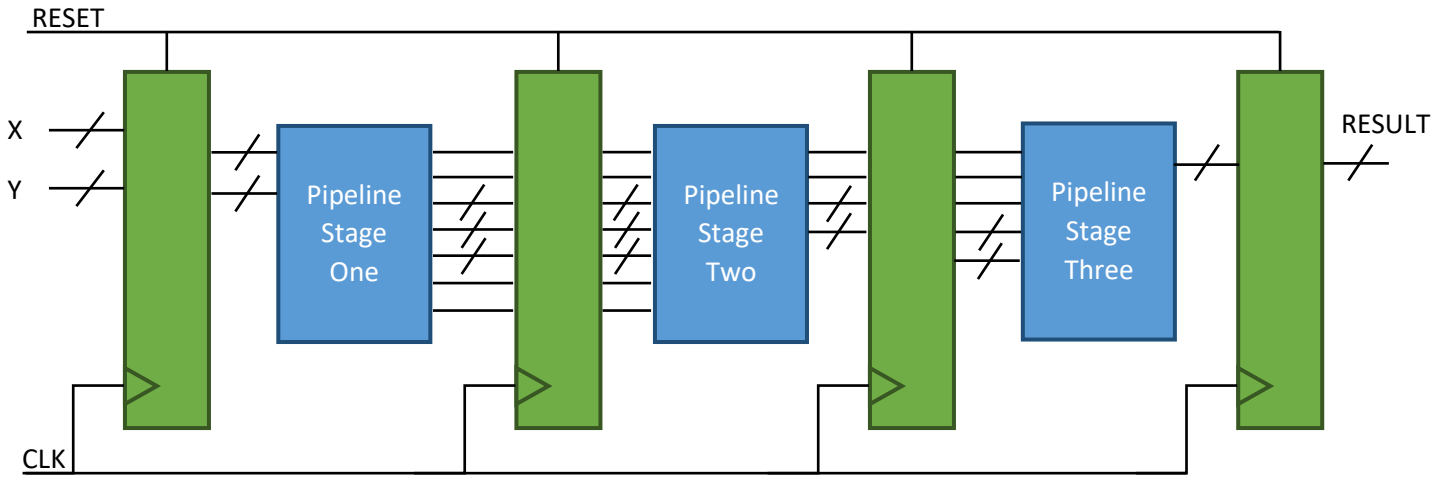

PipelineStageOne:1
PrepareForShift
TwoComplement
PFS
C2
Swap
S
SpecialCasesCheck
SPC
PipelineStageOne


PipelineStageTwo:1
SumDataAdapter
CarryLookAhead
SH
CLA
OperationCheck
OC
PipelineStageTwo



3.3 Struttura della pipeline

La pipeline risulta infine così strutturata:



3.4 Test architettura completa