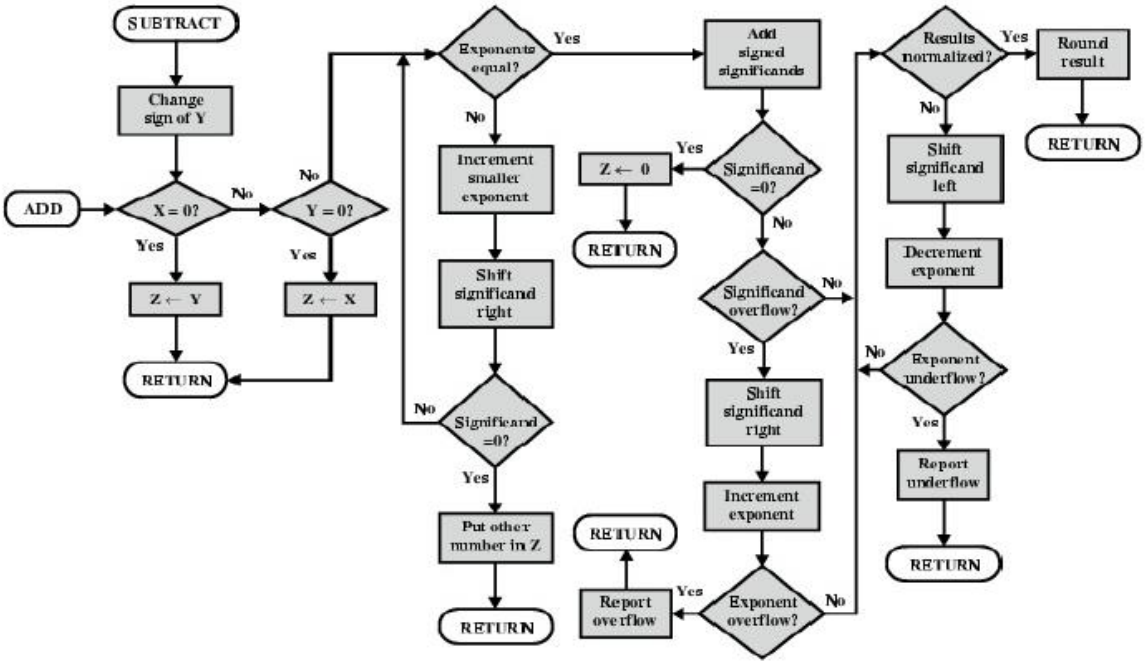
Progetto di Reti Logiche

Anno Accademico 2022-2023

Progetto 1: Sommatore floating-point IEEE754 [2P]

Si realizzi un sommatore floating-point secondo lo standard IEEE754. Il sommatore deve trattare operandi normalizzati e non e gli operandi speciali NaN, Infinity. Uno schema di massima del flusso di operazioni richieste è il seguente:

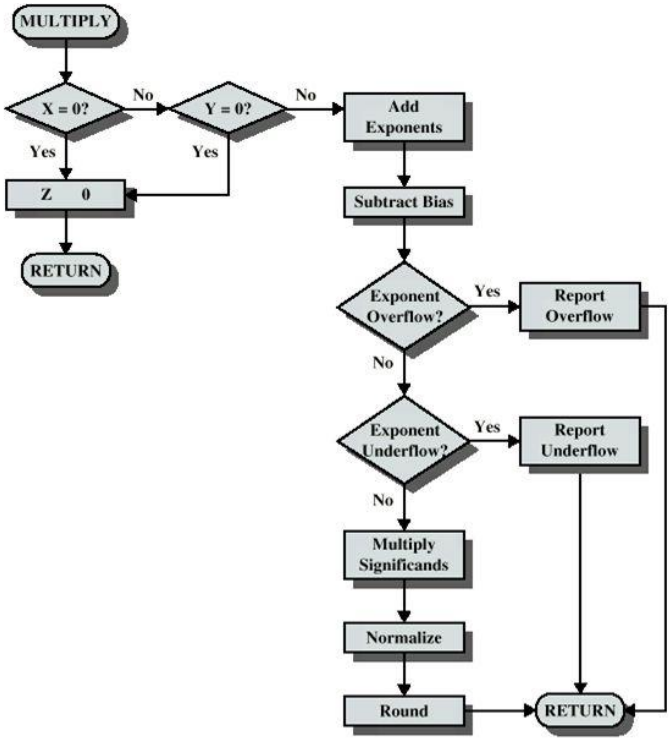


Il sommatore deve essere realizzato mediante una architettura pipelined, scegliendo la suddivisione in stadi il più possibile bilanciati.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

Progetto 2: Moltiplicatore floating-point IEEE754 [2P]

Si realizzi un moltiplicatore floating-point secondo lo standard IEEE754. Il moltiplicatore deve trattare operandi normalizzati e non e gli operandi speciali NaN, Infinity. Uno schema di massima del flusso di operazioni richieste è il seguente:

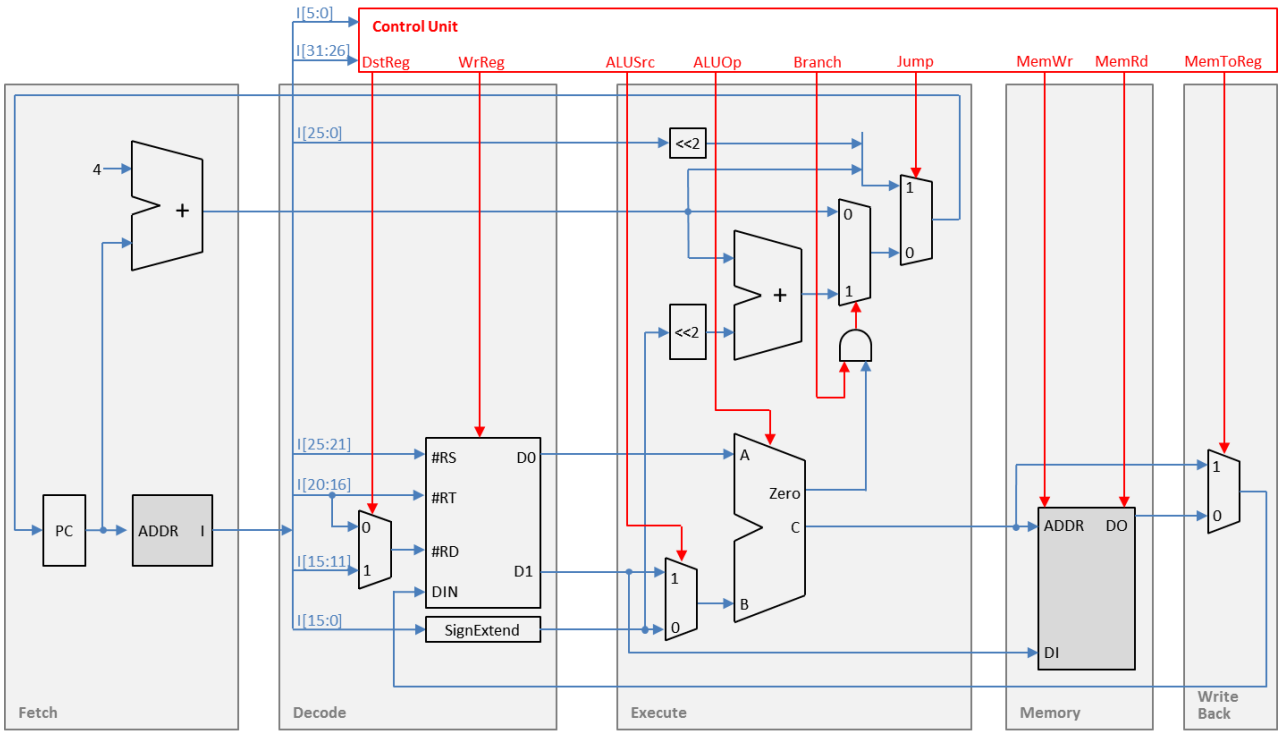


Il moltiplicatore deve essere realizzato mediante una architettura pipelined, scegliendo la suddivisione in stadi il più possibile bilanciati.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

Progetto 3: MIPS [3P]

Si progetti l'architettura MIPS senza pipeline e senza forwarding, così come mostrata dalla figura seguente.

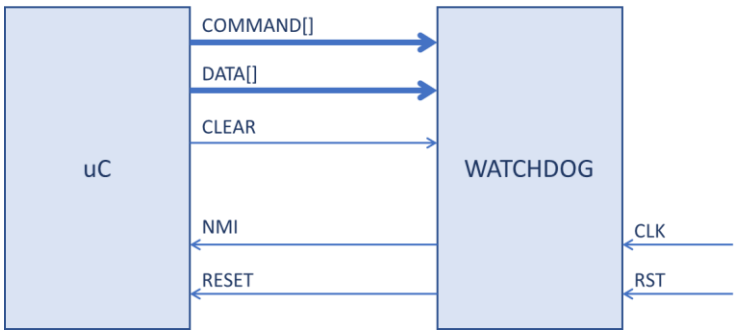


Ai fini della progettazione, la memoria può essere semplicemente modellata a livello behavioral come array di parole (non è richiesta l'istanziazione delle memorie della FPGA). È richiesto un incontro preliminare per discutere e concordare alcune semplificazioni all'architettura e alcune limitazioni dell'instruction set da supportare.

Una volta realizzato il microprocessore, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento di un semplice programma.

Progetto 4: Windowed watchdog counter programmabile [1P]

Si progetti un windowed watchdog counter in grado di generare anche un segnale di pre-allame. Lo schema seguente mostra un possibile utilizzo del watchdog accoppiato ad un microcontrollore.

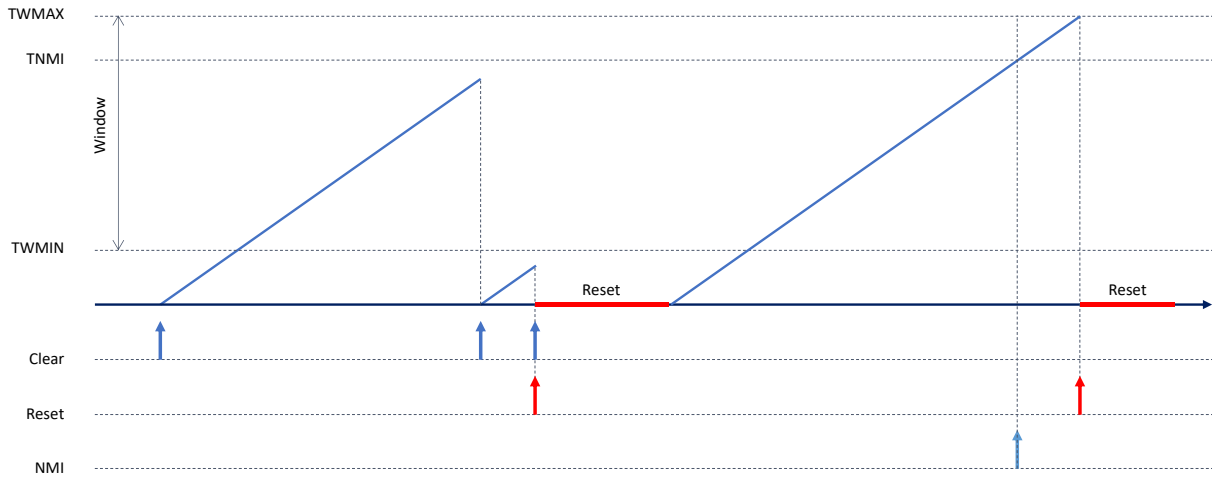


La procedura che il microcontrollore dovrà eseguire consiste per prima cosa nella configurazione del watchdog (vedi oltre) seguita da un comando di start. Ciò mette in funzione in watchdog che si comporterà come descritto nel seguito. Il microcontrollore, per evitare di ricevere un segnale di reset dal watchdog dovrà generare periodicamente un impulso sul segnale di CLEAR.

Il watchdog utilizza un contatore a 16 bit ed è caratterizzato da tre soglie temporali (cioè di conteggio):

- TWMIN: se il segnale di clear arriva prima di TWMIN, scatta il RESET
- TWMAX: se il segnale di clear arriva dopo TWMAX, scatta il RESET
- TNMI: se il contatore raggiunge il conteggio TNMI, viene generato un interrupt.

La seguente figura rappresenta il comportamento del watchdog:



Il watchdog è alimentato da un generico clock di sistema che viene diviso da un PRESCALER. In particolare, il prescaler può dividere il clock di ingresso per un fattore pari ad una potenza del due da 1 a 1024. Il clock così diviso costituisce la base dei tempi del contatore interno del watchdog.

Il watchdog espone inoltre una interfaccia di configurazione a comandi mediante due ingressi COMMAND e DATA che consentono di eseguire le seguenti impostazioni:

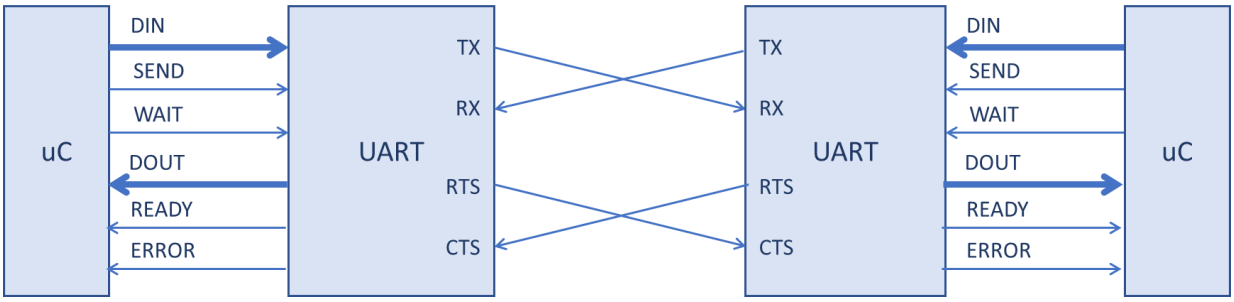
- Impostazione del fattore di divisione del prescaler
- Impostazione della soglia TWMIN
- Impostazione della soglia TWMAX
- Impostazione della soglia TNMI
- Start

Si tenga presente che deve sempre essere TWMIN < TNMI < TWMAX. Una volta fornito il comando di Start, il watchdog non può più essere riconfigurato e pertanto deve ignorare gli eventuali comandi che dovesse ricevere.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

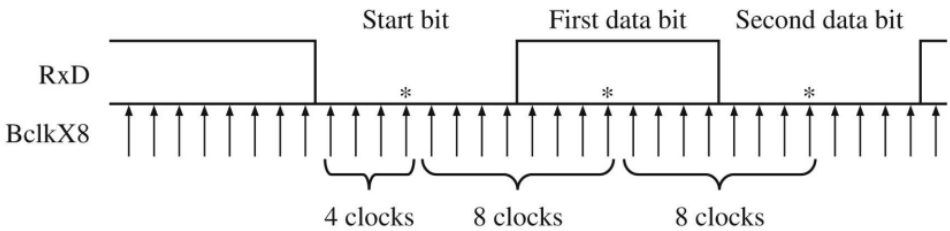
Progetto 5: Interfaccia seriale asincrona UART [2P]

Si progettino un trasmettitore ed un ricevitore UART dotati di segnalazione hardware del controllo del flusso. La trasmissione UART si basa su 4 linee digitali: TX (Trasmissione), RX (Ricezione), RTS (Request to send) e CTS(Clear to send). Queste linee sono connesse secondo lo schema mostrato di seguito.



La linea di trasmissione assume normalmente valore 1. Quando il trasmettitore intende iniziare un trasferimento dati abbassa la linea a 0 (start bit) quindi trasmette 8 bit di dati (un byte) ed un bit di parità (si assuma una parità pari), seguiti da un ulteriore 1 (stop bit). La trasmissione avviene sempre e solo in blocchi di 8 bit. La figura seguente mostra la trasmissione di un byte.

Trattandosi di una trasmissione asincrona, i fronti di transizione dei bit non sono allineati ad un clock condiviso fra trasmettitore e ricevitore ma avvengono ad una frequenza fissata, seppur spesso non precisa. Le frequenze tipiche di funzionamento (normalmente indicate come baudrate) sono 9600, 19200, 38400, 57600, 115200, 230400, 460800 e 921600. Dato che la frequenza effettiva di trasmissione/ricezione è spesso diversa da quella nominale e diversa tra trasmettitore e ricevitore, i segnale sulle linee di trasmissione e di ricezione vengono sovra-campionati con un clock 8 o 16 volte più veloce della frequenza effettiva di funzionamento. La figura seguente mostra tale situazione nel caso di sovra-campionamento di un fattore 8.



Quando il ricevitore osserva sulla linea una transizione da 1 a 0 ipotizza che si tratti dell'inizio di uno start bit. Per verificare questa condizione, attende 4 cicli di clock (se il fattore di sovra-campionamento è 8, l'attesa è 8/2 = 4) e campiona il segnale: se il valore campionato è 0, allora interpreta la transizione come uno start bit. In questo caso il momento in cui è stato campionato il segnale (indicato da un asterisco nella figura) cade approssimativamente a metà del tempo di bit. Procedendo di 8 cicli alla volta, il ricevitore quindi può campionare il segnale di ingresso per rilevare il valore di ogni successivo bit, fino ad averne ricevuti 9, di cui i primi otto di dati ed il nono di parità. Ciò fatto campiona un ulteriore bit e verifica se si tratta dello stop bit, cioè se la linea ha valore 1. Nel caso in cui non vengano rilevati lo start bit, lo stop bit oppure la parità ricevuta sia diversa dalla parità calcolata sui bit ricevuti, il ricevitore deve segnalare un errore.

L’interfaccia UART implementa inoltre il controllo di flusso hardware, gestito dai due segnali RTS e CTS. Quando un dispositivo (ricevente) è disponibile ed è in grado di accettare dati, mantiene il segnale RTS asserito: data la connessione tra RTS e CTS l’altro dispositivo (trasmittente) vede il segnale CTS asserito e trasmette. Poco prima che il ricevente diventi indisponibile (per esempio quando il suo buffer di ricezione è quasi pieno), questo deasserisce il segnale RTS, che viene rilevato sul CTS del trasmittente, il quale interrompe immediatamente la trasmissione.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi, inclusi differenze di fase e (piccola) differenza di frequenza tra i clock delle due interfacce.

Progetto 6: Divisore intero con resto [1P]

Si progetti un divisore intero su 32 bit basato sul metodo detto di “divisione lunga”. Siano N il dividendo, D il divisore, Q il quoziente ed R il resto. Tutti i valori sono da intendersi rappresentati in codifica binaria naturale. Il metodo iterativo è descritto dal seguente pseudocodice:

```
if( D == 0  ) {
  error();
}

Q = 0
R = 0
for( n = 31; n >= 0; n-- ) {
  R = R << 1
  R[0] = N[n]
  if( R ≥ D ) {
    R = R – D
    Q[i] = 1
  } else {
    Q[i] = 0
  }
}
```

Si progetti una rete sequenziale che implementa il divisore basato su tale algoritmo. Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

Progetto 7: Filtro esponenziale fixed-point [2P]

Si progetti un circuito sequenziale per la realizzazione di un filtro esponenziale configurabile. Il sistema riceve una sequenza continua di campioni X_t rappresentati in virgola fissa su 32 bit in cui 16 bit rappresentano la parte frazionaria e produce in uscita il segnale filtrato Y_t rappresentato con la stessa notazione e calcolato come:

$$Y_t = \alpha X_t + (1 - \alpha) Y_{t-1}$$

Per semplicità implementativa si limita il coefficiente del filtro α alle potenze negative del 2. In questo modo si ha:

$$\alpha = \frac{1}{2^k} \qquad 1 - \alpha = \frac{2^k - 1}{2^k}$$

Il circuito deve pertanto essere configurabile rispetto al valore di k che determina il coefficiente del filtro.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi. In particolare è necessario verificare il comportamento del filtro rispetto alla risposta al gradino.

Progetto 8: Moltiplicatore di Booth [1P]

Si progetti un moltiplicatore intero - puramente combinatorio - in grado di calcolare il prodotto di operandi con segno mediante la codifica di Booth nota come radix-2. Il moltiplicatore riceve in ingresso due operandi di 16 bit in codifica in complemento a due e produce in uscita un risultato su 32 bit, sempre in complemento a due. La codifica di Booth pertanto è utilizzata solo internamente e non è visibile all'interfaccia del componente.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.

Progetto 9: Moltiplicatore sequenziale [1P]

Si progetti un moltiplicatore sequenziale in grado di calcolare il prodotto di sue operandi A e B codificati in complemento a 2 su 32 bit. Si assuma inoltre che – nonostante la codifica – l'operando B sia sempre positivo. Il moltiplicatore riceve gli ingressi ed un segnale di START che avvia il prodotto, quindi procede a calcolare il risultato operando in modo sequenziale accumulando i prodotti parziali in un registro interno che, a prodotto ultimato, sarà fornito in uscita unitamente ad un segnale di ready.

Il comportamento del moltiplicatore può essere riassunto dalla seguente espressione:

$$Q_n = Q_{n-1} + |A| \cdot b_n$$

In cui Q_n indica il valore dell'accumulatore al passo n-esimo dell'operazione e b_n il bit di B in posizione n. Il risultato finale è ottenuto dopo 32 passi e pertanto coincide con Q_{32}. Si noti che l'operazione viene eseguita sul valore assoluto di A: per questa ragione il risultato finale deve poi essere corretto con il segno adeguato.

Una volta realizzato il componente, è richiesto di realizzare un test-bench per la simulazione e la verifica del corretto funzionamento nei diversi casi.