1 Minimizzazione reti combinatorie

Eseguire gli esercizi riportati nei lucidi e durante le lezioni frontali.

2 Reti combinatorie Esercizi svolti con l'ausilio di SIS e Mapping Tecnologico

Riproporre degli esempi che riportano il funzionamento di SIS e dei comandi studiati a lezione.

3 Latch/Flip Flop

Sviluppare i circuiti illustrati nel documento sui flip-flop. Eseguire per ciascun esercizio una simulazione comportamentale e post-sintesi, illustrando i passaggi salienti.

4 Display a 7 Segmenti

Illustrare la realizzazione di un'architettura che consenta di mostrare su un array di 4 display a 7 segmenti un valore intero. Tale può essere una parola da 16 bit, composta cioè di 4 cifre esadecimali, ciascuna espressa su di un nibble (4 bit). Sviluppare la traccia discutendo l'approccio di design adottato.

5 Clock generator

Sviluppare un progetto di sintesi di un DCM.

6 Scan Chain

Progettare una rete composta da una serie di N Flip Flop D abilitati ad operare nei seguenti due modi:

1. Modalità normale: l'array si comporta come un registro di N posizioni;
2. Modalità controllo: i flip flop possono essere scritti e letti individualmente configurandoli in cascata come uno shift register.

Utilizzare una rete di controllo in grado di alimentare il primo stadio con un valore e generare tanti colpi di clock quanto è la distanza del primo stadio dalla cella da raggiungere.

7 Finite State Machine

1. Realizzare un riconoscitore di una generica sequenza a N bit (e.g. 1011001) in VHDL utilizzando dapprima la descrizione behavioral a singolo o doppio processo (opzionale: poi i costrutti con guardia e simulare con GHDL).

2. A partire dal riconoscitore di sequenza realizzato al punto 1 con i costrutti behavioral a singolo o doppio processo del VHDL, sintetizzare la macchina specificando al tool di sintesi Xilinx ISE diverse codifiche per gli stati. Per quelle rilevanti estrapolare la codifica assegnata dal sintetizzatore, area occupata e frequenza massima di lavoro, apportando eventuali commenti.

8 Ripple Carry

Realizzare un'architettura di tipo Ripple Carry per un sommatore ad N bit generico. Il circuito deve essere realizzato a partire da blocchi di Full Adder, espresso mediante porte logiche XOR/AND/OR. Riportare considerazioni sull'area occupata e tempo di calcolo al variare di N e commentare il risultato con le formule teoriche.

9 Carry Look Ahead

Realizzare un'architettura di tipo Carry Look Ahead per un sommatore ad 8 bit. Il circuito deve essere realizzato a partire dai blocchi:

1. Propagation/Generation calculator

2. Carry Look Ahead

3. Full Adder

Opzionale: rendere il CLA generico.

10 Carry Save

Realizzare un esempio di addizionatore basato sulla modalità Carry Save.

11 Carry Select

Realizzare un sommatore Carry Select generico ad N bit. Il circuito deve essere realizzato a partire da blocchi di Full Adder, espresso mediante porte logiche XOR/AND/OR. Riportare considerazioni sull'area occupata e tempo di calcolo al variare di N e commentare il risultato con le formule teoriche.

12 Addizionatore 7 operandi

Progettare in VHDL un sommatore ad N bit capace di sommare 7 operandi. Il risultato della somma deve essere completo senza overflow, cioè la somma in uscita deve essere espressa su un numero di bit tali da consentire il non verificarsi di condizioni di overflow. Nel caso specifico i bit in uscita devono essere pari ad N+3. Il progetto può essere realizzato mediante composizione di Full Adder. Per ciascuna colonna si può effettuare un conteggio, producendo in uscita un valore binario espresso su 3 bit. Tenendo conto della posizione dei riporti un sommatore (e.g. ripple carry) può sommare tutti i riporti generati, restituendo il risultato finale. I riporti generati possono propagarsi sino alla cifra i+2 (e.g. la somma di 5 volte 1 genera 1 con riporto 10). Eventuali altre cifre binarie, come i riporti generati da somme precedenti, vanno considerati: quindi si verifica che un riporto può generarsi fino alla i+4-esima cifra binaria.

13 Moltiplicatori

Nota: degli esercizi 2 e 3 occorre farne almeno uno a piacere, l'altro è opzionale

1. Realizzare in VHDL un circuito di moltiplicazione a celle MAC di N bit. La cella MAC deve contenere un Full Adder (descritto già in esercizi precedenti) ed una porta AND per la moltiplicazione parziale. Tale cella deve essere replicata in una struttura ordinata (per righe e colonne) per comporre il circuito intero di moltiplicazione. Effettuare considerazioni di occupazione di area e di tempi di propagazione dei segnali al variare di N per valori significativi, apportando eventuali commenti salienti.

2. (ozionale se si fa il 3) Realizzare in hardware l'algoritmo della moltiplicazione secondo Robertson per operandi ad 8 bit. L'architettura deve essere realizzata sulla base dello schema di progettazione PO/PC (Parte Operativa e Parte di Controllo).

3. (opzionale se si fa il 2) Realizzare in hardware l'algoritmo della moltiplicazione secondo Booth per operandi ad 8 bit. L'architettura deve essere realizzata sulla base dello schema di progettazione PO/PC (Parte Operativa e Parte di Controllo).

4. (Opzionale) Realizzare in VHDL un circuito per la moltiplicazione con la struttura di somma per righe, oppure somma per colonne oppure somma per diagonale. Discutere l'architettura al variare di N.

14 Divisori

Nota: degli esercizi 1 e 2 occorre farne almeno uno, mentre l'altro è lasciato opzionale.

1. Realizzare in hardware l'algoritmo della divisione *Restoring* per operandi ad 8 bit. L'architettura deve essere realizzata sulla base dello schema di progettazione PO/PC (Parte Operativa e Parte di Controllo).

2. (opzionale) Realizzare in hardware l'algoritmo della divisione *Non Restoring* per operandi ad 8 bit. L'architettura deve essere realizzata sulla base dello schema di progettazione PO/PC (Parte Operativa e Parte di Controllo).

15 UART

Realizzare un dispositivo VHDL che implementa il protocollo UART (a partire da quello diffuso dalla Digilent). Collegare internamente, oppure tramite interfaccia fisica esterna alla board stessa, ad un'altra board oppure ad un PC previo utilizzo di un physical RS232, due interfacce per trasmette e ricevere ottetti.

Svolgere l'esercizio riutilizzando il VHDL messo a disposizione da Digilent (e disponbile nel materiale del corso) commentando eventuali ristrutturazioni del codice.

(Opzionale) Sviluppare un'architettura per l'implementazione del protocollo UART secondo il paradigma PO/PC ex-novo, evidenziando le similarità/dissimilarità con il progetto Digilent.

16 GPIO

Realizzare un dispositivo VHDL che implementa la logica tristate.

17 Firma digitale

Vedi PDF traccia finale sul sito Embedded.