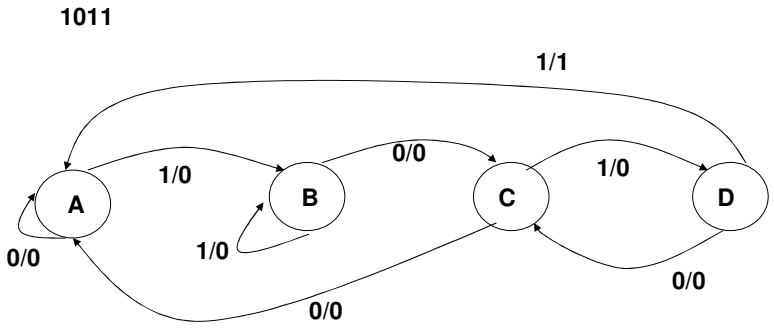
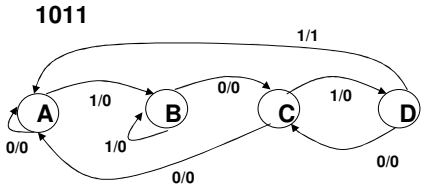
Esercizio 9

- Si ricavi il diagramma e la tabella degli stati di un sistema sequenziale la cui uscita vada ad 1 una volta riconosciuta la sequenza 1011.
- Le sequenze riconosciute non devono essere sovrapposte, ovvero: una volta riconosciuta una sequenza 1011, e prodotto un 1, il sistema ritorna nello stato di partenza. (In altre parole: l'ultimo 1 di una sequenza non può sovrapporsi col primo 1 di una sequenza successiva).

Esercizio 9 - Svolgimento



Esercizio 9 - La tabella degli stati



	0	*1*
A	A/0	B/0
B	C/0	**B/0**
C	A/0	D/0
D	C/0	A/1

Codifica degli stati

- Scegliendo la codifica naturale, bastano due bit:
- Stato Codifica
- A 00
- B 01
- C 11
- D 10

- Per comodità si è seguito l'ordine di inserimento nelle mappe di Karnaugh.

Tabella delle transizioni

Sostituzione degli stati con la loro codifica.

Q1Q0	Ingresso I	
	0	1
00	00/0	01/0
01	11/0	01/0
11	00/0	10/0
10	11/0	00/1

Scelta degli Elementi di Memoria

- Scegliamo, come elementi di memoria, flip flop di tipo D, per cui la tabella delle eccitazioni coincide con la tabella delle transizioni.
- Tra l'altro, il programma SIS utilizza gli stessi elementi di memoria.

Tabella delle eccitazioni

Q1Q0	Ingresso I	
	0	1
00	00/0	01/0
01	11/0	01/0
11	00/0	10/0
10	11/0	00/1

Nelle caselle sono indicati, nell'ordine, l'ingresso D1 del flip flop Q1, l'ingresso D0 del FF Q0 e l'uscita U.

Sintesi

- Riportiamo la tabella solo per D1 e semplifichiamo.
- Risulta:
- D1=!I !Q1 Q0 + I Q1 Q0 + !I Q1 !Q0

Q1Q0	Ingresso I	
	0	1
00	0	0
01	1	0
11	0	1
10	1	0

Sintesi

- Riportiamo la tabella solo per D0 e semplifichiamo.
- Risulta:
- D0=!Q1 Q0 + I !Q1 + !I Q1 !Q0

Q1Q0	Ingresso I	
	0	1
00	0	1
01	1	1
11	0	0
10	1	0

Sintesi

- Riportiamo la tabella solo per U e semplifichiamo.
- Risulta:
- U = I Q1 !Q0

Q1Q0	Ingresso I	
	0	1
00	0	0
01	0	0
11	0	0
10	0	1

File blif

- Il file blif che descrive il sistema in esame è il seguente:
- .model sequenza
- .inputs I
- .outputs U
- .start_kiss
- .i 1
- .o 1
- .s 4
- .r a
- #tabella delle transizioni
- 0 a a 0
- 0 b c 0
- 0 c a 0
- 0 d c 0
- 1 a b 0
- 1 b b 0
- 1 c d 0
- 1 d a 1
- .end_kiss
- #codifica degli stati
- .code a 00
- .code b 01
- .code c 11
- .code d 10
- .end

File blif

- Dopo il comando stg_to_network per creare le funzioni di stato prossimo e di uscita, il comando wl ci mostra:
- .model sequenza
- .inputs I
- .outputs U
- # sono stati generati due latch per gli stati (ingresso, uscita, uscita nello stato di reset “a=00”)
- .latch [7] LatchOut_v1 0
- .latch [8] LatchOut_v2 0
- .start_kiss
- .i 1
- .o 1
- .p 8
- .s 4
- .r a
- 0 a a 0
- 1 a b 0
- 0 b c 0
- 1 b b 0
- 0 c a 0
- 1 c d 0
- 0 d c 0
- 1 d a 1
- .end_kiss

File blif

- .latch_order LatchOut_v1 LatchOut_v2
- # l'ordine di codifica è LatchOut_v1=bit più significativo e LatchOut_v2=bit meno significativo
- .code a 00
- .code b 01
- .code c 11
- .code d 10
- .names I LatchOut_v1 LatchOut_v2 [7]
- # ingresso del latch v1 (D1 nella sintesi precedente)
- 111 1
- 001 1
- 010 1
- .names I LatchOut_v1 LatchOut_v2 [8]
- # ingresso di v2 (D0 nella sintesi precedente)
- 10- 1
- 001 1
- 010 1
- .names I LatchOut_v1 LatchOut_v2 U
- # uscita U
- 110 1
- .end

File blif

- L'istruzione write_eqn ci da:

- INORDER = I LatchOut_v1 LatchOut_v2;
- OUTORDER = [7] [8] U;
- [7] = !I*LatchOut_v1*!LatchOut_v2 + !I*!LatchOut_v1*LatchOut_v2 + I*
- LatchOut_v1*LatchOut_v2;
- [8] = !I*LatchOut_v1*!LatchOut_v2 + !I*!LatchOut_v1*LatchOut_v2 + I*!
- LatchOut_v1;
- U = I*LatchOut_v1*!LatchOut_v2;
- Che è lo stesso risultato ottenuto manualmente.

File blif

- Il comando di minimizzazione degli stati “state_minimize stamina” da come risultato sempre 4 stati.
- Il comando “state_assign jedi” provvede alla ricodifica degli stati. Dopo ogni minimizzazione degli stati bisogna sempre ricodificare gli stati stessi

File blif

- La semplificazione della rete combinatoria con “fs” e “source script.rugged” da come risultato:

```
.model sequenza
.inputs I
.outputs U
.latch    [15] LatchOut_v1    0
.latch    [16] LatchOut_v2    1
.start_kiss
.i 1
.o 1
.p 8
.s 4
```

File blif

- .r a
- 0 a a 0
- 1 a b 0
- 0 b c 0
- 1 b b 0
- 0 c a 0
- 1 c d 0
- 0 d c 0
- 1 d a 1
- .end_kiss

File blif

- .latch_order LatchOut_v1 LatchOut_v2
- .code a 01
- .code b 10
- .code c 00
- .code d 11
- .names I U [15]
- 10 1
- .names LatchOut_v1 U [18] [16]
- -1- 1
- 0-1 1
- .names LatchOut_v1 [18] U
- 10 1
- .names I LatchOut_v2 [18]
- 0- 1
- -0 1
- .end
- Da notare la nuova codifica degli stati e la rete multilivello risultante.

File blif

- Infine, con il comando “we” risulta:

- INORDER = I LatchOut_v1 LatchOut_v2;
- OUTORDER = [15] [16] U;
- [15] = I*!U;
- [16] = !LatchOut_v1*[18] + U;
- U = LatchOut_v1*![18];
- [18] = !LatchOut_v2 + !I;

- P.S. Il disegno del circuito elettronico risultante è lasciato agli studenti di buona volontà.