COMANDI PER IL PROGRAMMA:

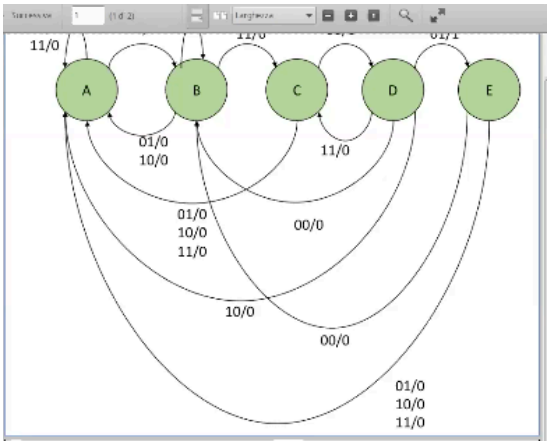
.model <nome modello>
.inputs <tutti gli inputs separati da uno spazio>
.outputs <tutti gli outputs>

.names <inserire tutti gli inputs e un solo output alla volta che verranno utilizzati nelle righe successive>

(Per ogni nuovo output si deve fare un .names)

.exdc (va messo nella riga prima del .names per descrivere il don't care set)
— —
(Posso scomporre in sotto tabelle di verità un'operazione complessa e per fare ciò in .output metto solo gli outputs finali mentre nel .names posso aggiungere degli outputs intermedi da utilizzare prima di raggiugnere quelli dichiarati)

.start_kiss (introduco una macchina a stati inserendo una riga sotto l'altra:
.i <numero segnali di ingresso>
.o <numero segnali di uscita>
.s <numero di stati>
.p <numero di transizioni>
.r <stato di reset: ATT/NUL>.




```
4
5  .start_kiss
6  .i 2
7  .o 1
8  .p 20
9  .s 5
10 .r A
```

successivamente scrivo la tabella delle transizioni inserendo:
(ingressi, stato presente, stato prossimo, uscita)
es:

0- - ATT ATT 0
-10 11 ATT 0. E così via…

.end_kiss

```
11
12 #tabella degli stati
13 #i0 i1 stato_presente stato_futuro out
14 00 A B 0
15 01 A A 0
16 10 A A 0
17 11 A A 0
18 00 B B 0
19 01 B A 0
20 10 B A 0
21 11 B C 0
22 00 C D 0
23 01 C A 0
24 10 C A 0
25 11 C A 0
26 00 D B 0
27 01 D E 1
28 10 D A 0
29 11 D C 0
30 00 E B 0
31 01 E A 0
32 10 E A 0
33 11 E A 0
34 .end_kiss
35
36 .end
```

.code <simbolo/nome dello stato….spazio e valore binario che vado ad associare a quello stato >
(serve a codificare gli stati, è opzionale perché può essere calcolata automaticamente tramite il comando state_assign jedi sul terminale)

.subckt <nome del modello parametro formale= parametro attuale>.
Seguito da .search <nome file componente.blif>.
(servono per richiamare dei componenti da altri modelli) (metterli nella stessa cartella)

```
.model xnor
.inputs A
.outputs B

.names A B
00 1
11 1

.end
```

```
.model UGUALE4
.inputs A3 A2 A1 A0 B3 B2 B1 B
.outputs O
.subckt xnor A=A3 B=B3 X=X3
.subckt xnor A=A2 B=B2 X=X2
.subckt xnor A=A1 B=B1 X=X1
.subckt xnor A=A0 B=B0 X=X0
.names X3 X2 X1 X0 O
1111 1
.search xnor.blif
.end
```

Utilizzo alla fine .search <nome file.blif>

.latch <input><output> <type> <control> <init-val> (al posto di .names per descrivere una macchina sequenziale)

.end (per concludere)

COMANDI SUL TERMINALE (SIS):

1. Sis (apre la sezione sis)
2. read_blif <nome file> .blif (legge il file)
3. write_blif (printa sul terminale il file letto) e si può usare per copiare in un nuovo file
4. simulate <inserire gli inputs distanziati di uno spazio tra loro> (simula il funzionamento del programma inseriti degli input a scelta)
5. print_stats (

Stampa a video importanti informazioni sul circuito: il numero di segnali in input (`PI`), il numero di segnali in output (`PO`), in numeto di nodi (`nodes`), il numero di elementi di memoria (`latches`) e il numero di letterali (`lits`).

PARTE DI MINIMIZZAZIONE
6. full_simplify (esegue l'ottimizzazione con SIS)
7. write_eqn (visualizza l'espressione booleana ovvero in somma di prodotti corrispondente al modello rappresentato in blif e varia prima e dopo aver utilizzato la full_simplify)
8. Sweep (elimina i nodi con un'unica linea di ingresso e i nodi con valore costante)
9. Eliminate (elimina un nodo interno alla rete, ovvero sostituisce l'espressione Booleana di un nodo, con una variabile) esempio y= (a+b)*c.
10. Resub (sostituisce un nodo interno con un'insieme di nodi la cui funzionalità sia equivalente, ciò serve a diminuire la complessità del nodo)
11. .fx (esegue l'operazione di estrazione)

12. Source script.rugged (ricerca l'ottimizzazione migliore del circuito) (non è assicurato il risultato)

13. Read_library <libreria> (mcnc.genlib o synch.genlib)
14. Print_library (visualizza informazioni inerenti la libreria caricata)

15. Map (esegue l'operazione di mapping): -m 0 —> minimizza rispetto all'area
-n 1 —> minimizza rispetto al ritardo
dopo il map -m/n 0/1 uso -s —> permette di visualizzare le informazioni di area e ritardo post mapping (total gate area e maximum arrival time)

17. Print_delay (stampa informazioni relative al ritardo del circuito nodo per nodo e più dettagliato)

18. Reduce_depth (riduce la lunghezza dei cammini critici e quindi del ritardo del modello)

19. State_minimize_stamina (minimizza gli stati con l'algoritmo di Pohl anger)

20. Stg_to_network (crea le funzioni di stato prossimo e di uscita)

21. State_assign jedi (aggiunge la codifica automatica degli stati)

22. Print_state (dice su che stato ci si trova)