Università degli Studi di Verona

AA 2018/2019

Corso di laurea in Informatica

Elaborato SIS

Laboratorio di Architettura degli elaboratori

A cura di:

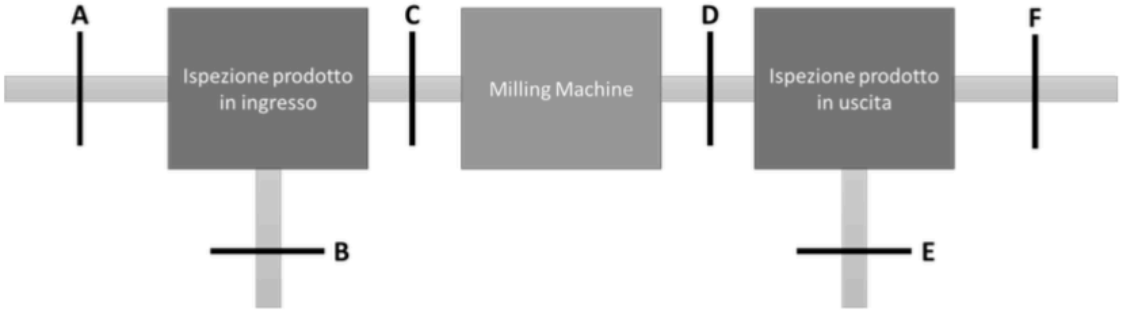
Tacchella Enrico - VR429589

Procaccio Angela - VR437075

Indice

Specifiche

Si progetti un dispositivo per la gestione di un centro di lavoro per asportazione di truciolo.
Il dispositivo è cosi composto:



Il dispositivo riceve in ingresso 7 bit che assumono significato diverso in base allo stato in cui si trovano.
Il principio di funzionamento dell'impianto è strutturato in diverse fasi:

➢ L'impianto è inizialmente spento (O/I=0). L'impianto si accende quando il comando di accensione (ON) di 3 bit è composto dalla sequenza 111 (ON=111). L'impianto si accende (O/I=1) e inizia a lavorare aprendo il gate A (GA=1) e permettendo il caricamento di un pezzo grezzo nell'unità di controllo in ingresso. Il dispositivo deve inoltre incrementare un contatore dei pezzi in ingresso (NA).

➢Nel modulo di ispezione prodotti in ingresso viene controllata la qualità del pezzo grezzo (QI, 1=buono, 0=scarto) e ne viene misurato il volume (VIN) espresso in 6 bit. Se il pezzo risulta scarto (QI=0), il dispositivo deve aprire il gate B (GB=1) mandando il pezzo in un deposito di scarti e aumentare il contatore dei pezzi scartati in ingresso (NB). Al ciclo successivo il dispositivo ricomincia il ciclo aprendo il gate A. Se il pezzo risulta buono (QI=1) si apre il gate C (GC=1) ed il pezzo può passare alla fresa incrementando il contatore NC.

➢Nella macchina il pezzo grezzo viene fresato per ottenere un pezzo lavorato; la macchina fornisce un valore binario EM che indica la riuscita o meno della lavorazione, ed il volume del pezzo lavorato (VOUT) espresso in 6 bit.
se la macchina ha generato un errore (EM=0) l'impianto deve spegnersi generando il codice di errore ERR=001.

Se la lavorazione è andata a buon fine (EM=1), il pezzo viene scaricato attraverso il gate D (GD=1) e passa al controllo qualità in uscita. La differenza tra il volume del pezzo grezzo VIN e quello del pezzo lavorato VOUT è scarto che va a finire nel serbatoio della macchina. Quando la quantità di scarto nella macchina supera il valore 200 l'impianto deve spegnersi generando il codice di errore ERR=010.

➢Il prodotto lavorato entra nel modulo di ispezione in uscita che restituisce l'indicatore di qualità del pezzo lavorato (QO, 1=buono, 0=scarto).
Se il pezzo risulta buono (QO=1), il dispositivo apre il gate F (GF=1) per mandare il pezzo a magazzino, incrementando il contatore dei pezzi lavorati correttamente (NF).
Se il pezzo risulta scarto (QO=0), il dispositivo apre il gate E (GE=1) per mandare il pezzo in un deposito scarti lavorati ed aumenta il contatore relativo (NE).
In entrambi i casi al ciclo successivo la macchina riprende il ciclo aprendo il gate A.

➢Tutti i contatori dei pezzi (NA, NB, NC, NE e NF) sono espressi in 4 bit. Nel caso in cui il numero di pezzi nel deposito degli scarti in ingresso (NB) generi un overload l'impianto deve spegnersi con codice di errore ERR=101. Analogamente per quanto riguarda il deposito degli scarti in uscita (NE), generando il codice ERR=110.

➢Ad ogni ciclo di clock il dispositivo deve restituire una sequenza di 30 bit indicativi delle seguenti variabili (nell'ordine di seguito indicato!):

O/I ERR GA GB GC GD GE GF NA NB NC NE NF

NB: ad ogni ciclo di clock può essere aperto al massimo un gate!

Introduzione

Questa relazione viene fornita assieme ai sorgenti .blif del progetto.

I file più importanti sono:

- **controllo.blif** : è il file che controlla in che stato si trova la macchina prima dell'assegnamento;

- **controlloNew.blif** : è il file che controlla in che stato si trova la macchina dopo l'assegnamento di essi;

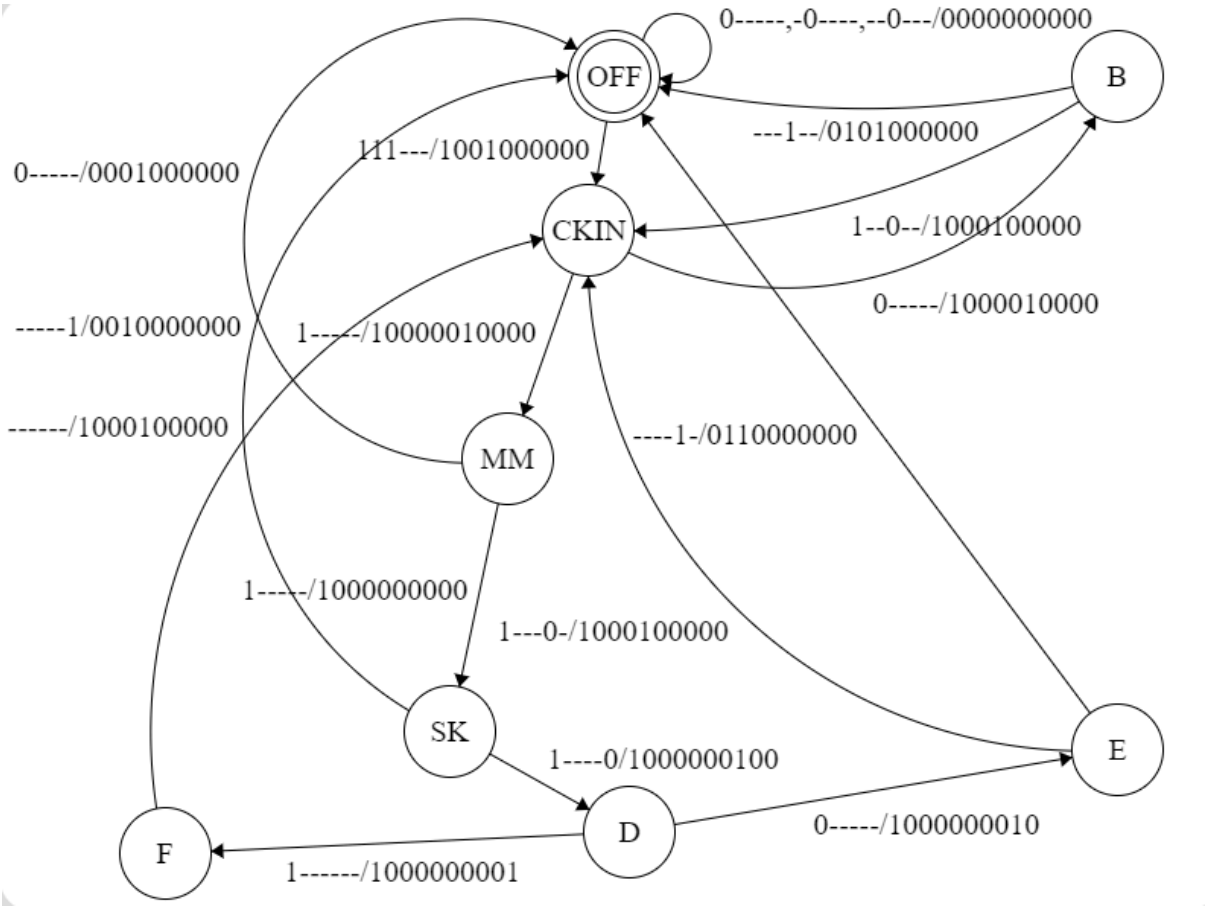
- **datapath.blif**: è il file datapath;

- **FSMD.blif**: è il file contenente il circuito completo che raggruppa controlloreNew.blif e datapath.blif.

Controllore (FSM)

Il Controllore è la parte che si occuperà della scelta del Gate e della strada da prendere in base alle condizioni del pezzo da lavorare, terminando in una

riuscita della lavorazione o negli eventuali stati di errore dove la lavorazione non è andata a buon termine o dove quando il serbatoio che contiene gli scarti raggiunge capienza massima.



Descrizione

Il controllore è formato dai seguenti input, 6 in tutto:
- i primi tre servono per dare il comando di accensione (111);
- gli ultimi tre, a seconda dello stato in cui si trova, definiscono se la macchina va in errore;

gli output invece sono 10 bit:
- il primo indica se la macchina è accesa o spenta;
- i 3 successivi indicano l'errore in cui si trova la macchina nel caso appunto che venga riscontrato un problema durante la lavorazione;
- I 6 bit finali sono rispettivamente GA, GB, GC, GD, GE e GF, quando il bit vale 1 vuol dire che quel Gate è aperto.

Il controllore è composto da 8 stati:
- **OFF**: è lo stato di reset e quando la macchina si trova in questo stato vuol dire che è spenta. Viene accesa solamente quando il segnale composto da 3 bit 111 non arriva alla macchina. Questo stato è anche il successivo di tutti gli errori che potrebbero generarsi durante la lavorazione;
- **CHECKIN (CKIN)**: è il primo stato che la macchina incontra quando si accende, corrisponde la Gate A;
- **B**: stato che viene raggiunto se il pezzo che viene preso in consegna non è nelle condizioni ottimali, può terminare in errore;
- **MILLING MACHINE (M.M.)**: stato che viene raggiunto se il pezzo che viene preso in consegna è in buone condizioni, può terminare in errore;
- **SCARTO (SK)**: dopo la lavorazione del pezzo, dove ovviamente sono stati asportati pezzi, quesi ultimi finiscono in un serbatoio, se il serbatoio si riempie termina in errore;
- **D**: se il serbatoio non è pieno, questo stato valuta le condizioni del pezzo;
- **E**: se il pezzo non ha ricevuto una buona lavorazione, viene scartato e quando questi scarti supereranno 15 (1 1 1 1) si genererà un errore;
- **F**: se il pezzo ha ricevuto una buona lavorazione, viene conteggiato e la macchina ritornerà al CHECKIN per una nuova lavorazione, unico caso in cui la macchina non si spegne.

Realizzazione in SIS

Per la realizzazione del controllore in SIS sono stati creati 2 file blif, che ora analizzeremo:

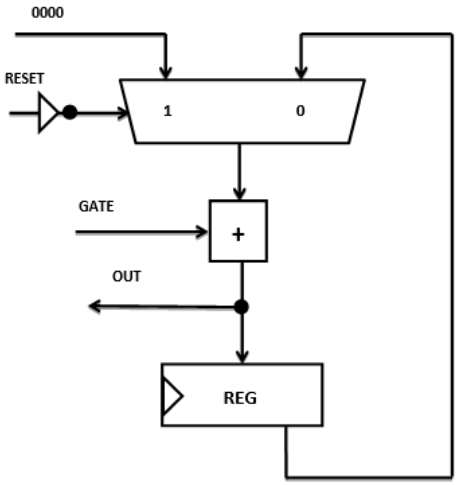
- **controllo.blif**: contiene la FSM. Per poter testare la FSM con il file FSMD.blif dobbiamo prima assegnare gli stati tramite il comando di SIS "state_assign jedi".

- **controlloNew.blif** : è la FSM dopo aver eseguito il comando "state_assign jedi". È necessario avere gli stati codificati affinchè sia possibile testare il circuito completo con centralina.blif.

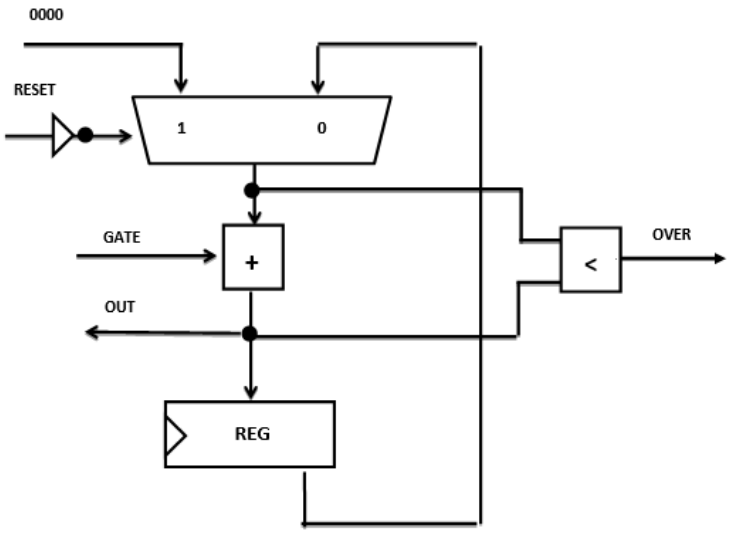
Datapath

Il datapath riceve in ingresso il Gate e il volume iniziale del pezzo e restituisce il numero di oggetti che sono passati per un determinato Gate, ad esempio per il GA incrementerà il contatore NA ogni qualvolta che un pezzo passerà per esso; NA è un valore formato da 4 bit, come tutti gli altri contatori. Per quanto riguarda il volume, è ricevuto in ingresso tramite una sequenza di 6 bit, è lavorato ed esce con una sequenza di altrettanti bit, al suo interno però conta anche la capacità del serbatoio dello scarto, infatti nel caso in cui questo raggiungesse una soglia tale da non permettere il contenimento di altri pezzi (200 nel nostro caso), viene generato un errore e la macchina si spegne.

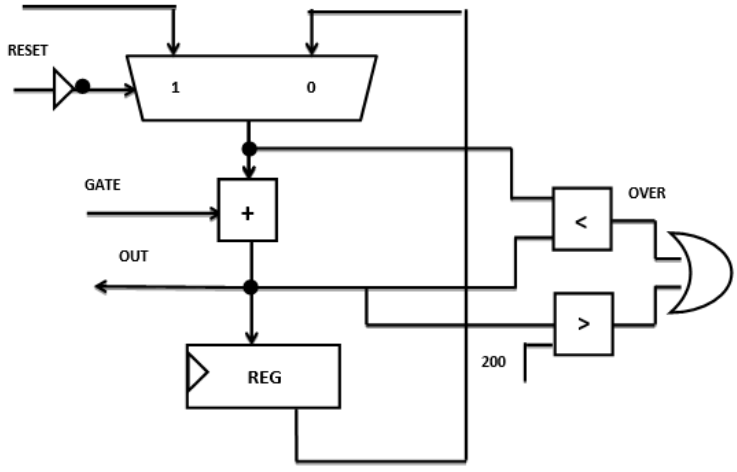
Il datapath è formato da 4 grandi componenti:



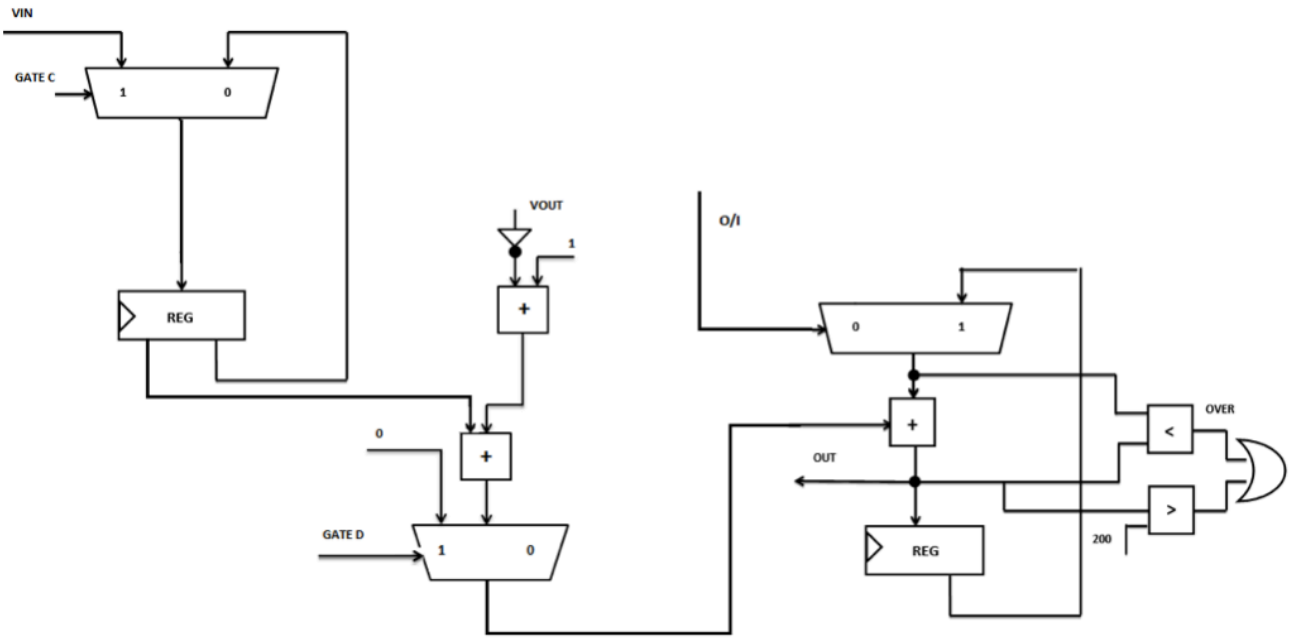
Questo è il contatore base utilizzato per i Gate A, Gate C e Gate F. È composto da 3 moduli: Multiplexer (componente che serve a portare in uscita uno degli ingressi, opportunamente scelto dall'ingresso di selezione) a 4 bit, Sommatore a 4 bit, registro a 4 bit e un negatore.
Il funzionamento è il seguente: riceve il bit dal controllore il quale indica che il Gate è aperto e incrementa di 1 il contatore corrispondente, formato appunto da 4 bit.



Questo è il contatore utilizzato per il Gate B e il Gate E.
Ha lo stesso funzionamento del contatore precedente però ha una parte aggiuntiva: il modulo del minore. Infatti dato che ogni volta incrementa di 1, se il numero incrementato risulta minore del numero precedente, significa che è avvenuto un overflow.
Se questo avviene il segnale di over viene trasmesso al controllore che procederà a terminare con un output di errore.



Questo è il contatore utilizzato per il controllo del serbatoio degli scarti.
Tutti i moduli però non sono più a 4 bit ma a 8 perché il 200 è codificato su 8 bit.
L'idea di funzionamento è lo stesso del contatore precedentemente descritto.



Questo è il componente che calcola il volume in output.
Il datapath riceve in ingresso il volume iniziale (VIN) e calcola il volume finale, questo passerà al Gate D se passa il controllo del contatore degli scarti, presente in figura e precedentemente descritto.

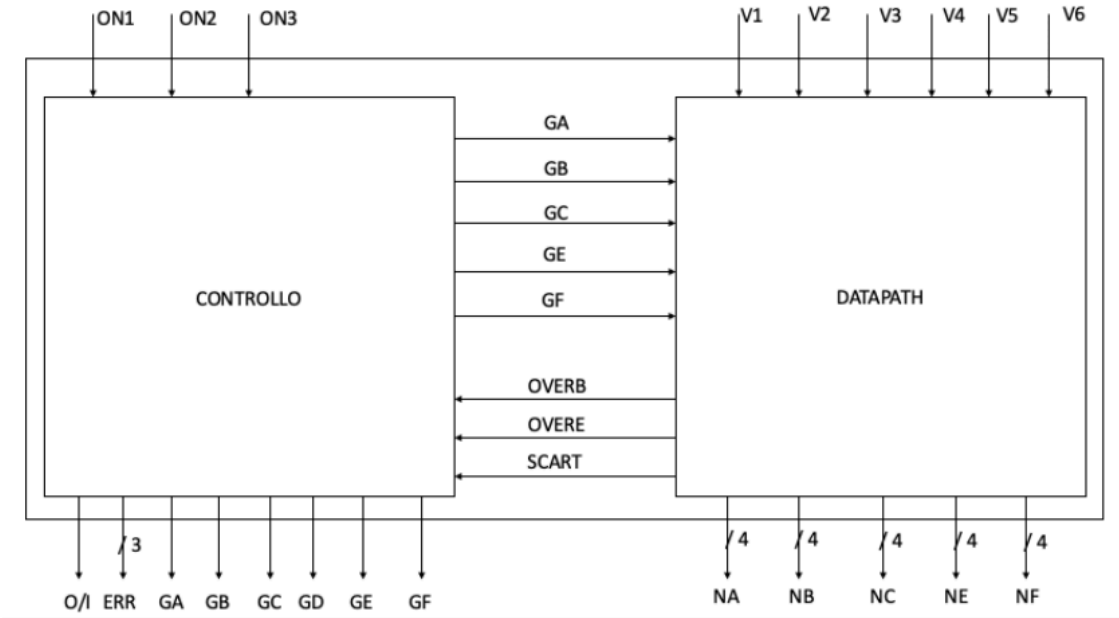
Realizzazione in SIS

Per la realizzazione in SIS sono stati creati 26 file blif:

- datapath.blif : è il file che collega tutti i componenti;
- contatore.blif : è il file contatore base;
- Contrconta.blif : è il file che conta specificatamente Gate per Gate;
- Contrscarti.blif : è il file che controlla gli scarti;
- Duecento.blif : è il file per la costante 200;
- Maggiore8.blif : realizza la funzione “maggiore di” a 8 bit;
- Minore8.blif : realizza la funzione “minore di” a 8 bit;
- Mux4.blif : è un multiplexer a 4 bit;
- Mux6.blif : è un multiplexer a 6 bit;
- Mux8.blif : è un multiplexer a 8 bit;
- Negatore8.blif : realizza la funzione di negazione a 8 bit;
- or.blif : realizza la funzione or;
- Registro4.blif : è un registro a 4 bit;
- Registro6.blif : è un registro a 4 bit;
- Registro8.blif : è un registro a 4 bit;
- Scarto.blif : è il file scarto base;
- Somm1.blif : realizza la funzione somma a 1 bit;
- Somm4.blif : realizza la funzione somma a 4 bit;
- Somm6.blif : realizza la funzione somma a 6 bit;
- Somm8.blif : realizza la funzione somma a 8 bit;
- Sottrazione2.blif : realizza la funzione sottrazione a 2 bit;
- Sottrazione6.blif: realizza la funzione sottrazione a 6 bit;
- Uno.blif : è la costante 1;
- Volumi.blif : è il file che calcola il volume;
- Zero.blif : è la costante 0;

FSMD

La FSMD è il file che contiene il datapath e il controllore, di fatto è ciò che dobbiamo eseguire per il nostro progetto.
Riceve in ingresso 7 bit e da in uscita 30 bit.



In figura in ingresso si vedono 9 bit perché i seguenti si riferiscono allo stesso bit: ON2 è anche V1, mentre ON3 è anche V2

Realizzazione in SIS

Il file FSMD è collega datapath e controllore.
Utilizza anche un registro a un bit per evitare cicli e un not per non azzerare il contatore corrispondente al Gate preso in considerazione

```
.model FSMD
.inputs IP0 IP1 IP2 IP3 IP4 IP5 IP6
.outputs AIO ERR1 ERR2 ERR3 GA GB GC GD GE GF NA3 NA2 NA1 NA0 NB3 NB2 NB1 NB0 NC3 NC2 NC1
NC0 NE3 NE2 NE1 NE0 NF3 NF2 NF1 NF0

.subckt ZERO ZERO=ZERO
.subckt CONTROLLO ON1=IP0 I1=IP1 I2=IP2 E1=ZERO E2=ZERO E3=ZERO ON=AIO ERR1=ERR1 ERR2=ERR2
ERR3=ERR3 GA=GA GB=GB GC=GC GD=GD GE=GE GF=GF

.subckt REGISTRO A=OVERB O=OVERB1

.subckt REGISTRO A=OVERE O=OVERE1

.subckt REGISTRO A=SCART O=SCART1

.subckt NOT A=AIO O=RESET

.subckt DATAPATH VIN5=IP1 VIN4=IP2 VIN3=IP3 VIN2=IP4 VIN1=IP5 VIN0=IP6 GA=GA GB=GB GC=GC
GD=GD GE=GE GF=GF RESET=RESET ERR=ERR NA3=NA3 NA2=NA2 NA1=NA1 NA0=NA0 NB3=NB3 NB2=NB2
NB1=NB1 NB0=NB0 NC3=NC3 NC2=NC2 NC1=NC1 NC0=NC0 NE3=NE3 NE2=NE2 NE1=NE1 NE0=NE0 NF3=NF3
NF2=NF2 NF1=NF1 NF0=NF0


.end

.search controlloNew.blif
.search datapath.blif
.search registro.blif
.search not.blif
```

Ottimizzazione del circuito

Il circuito così ottenuto va ottimizzato, cioè si deve ridurre il numero di componenti e/o il ritardo senza però alterarne il funzionamento . Per ottimizzare i circuiti SIS fornisce vari comandi come sweep, eliminate, resub, fx, simplify e full_simplify: tuttavia nella maggior parte dei casi lo script "script.rugged", il quale utilizza tutte queste funzioni, permette di ottenere i risultati migliori. Verrà utilizzato inoltre il comando "print_stats" per verificare le statistiche del circuito. Prima di ottimizzare il circuito interamente, eseguiremo le operazioni di minimizzazione prima sul datapath e poi sul controllore perché abbiamo constatato che il circuito viene minimizzato ulteriormente.

Datapath

```
sis> rl datapath.blif
Warning: network `SOTTRAZIONE6', node "NO" does not fanout
Warning: network `VOLUMI', node "NO" does not fanout
Warning: network `VOLUMI', node "[218]" does not fanout
Warning: network `DATAPATH', node "NO" does not fanout
Warning: network `DATAPATH', node "[218]" does not fanout
Warning: network `DATAPATH', node "[472]" does not fanout
Warning: network `DATAPATH', node "[473]" does not fanout
Warning: network `DATAPATH', node "[474]" does not fanout
DATAPATH        pi=13   po=21   nodes=188       latches=34
lits(sop)= 788
sis> source script.rugged
DATAPATH        pi=13   po=21   nodes=125       latches=34
lits(sop)= 404
DATAPATH        pi=13   po=21   nodes=111       latches=34
lits(sop)= 404
DATAPATH        pi=13   po=21   nodes=111       latches=34
lits(sop)= 378
DATAPATH        pi=13   po=21   nodes=107       latches=34
lits(sop)= 374
DATAPATH        pi=13   po=21   nodes=107       latches=34
lits(sop)= 374
DATAPATH        pi=13   po=21   nodes= 65       latches=34
lits(sop)= 516
DATAPATH        pi=13   po=21   nodes= 65       latches=34
lits(sop)= 461
DATAPATH        pi=13   po=21   nodes= 65       latches=34
lits(sop)= 461
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
DATAPATH        pi=13   po=21   nodes= 92       latches=34
lits(sop)= 379
```

Utilizzando "script.rugged" il numero di letterali del datapath è passato da 788 prima dell'ottimizzazione a 379 e il numero di nodi da 188 a 92. Ogni altra esecuzione dello script non ha ottimizzato il circuito ulteriormente.

Controllore

```
sis> rl controlloNew.blif
Warning: network `controlloNew.blif', node "I1" does not fanout
Warning: network `controlloNew.blif', node "I2" does not fanout
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)= 168  #states(STG)=   8
sis> state_minimize stamina
Running stamina, written by June Rho, University of Colorado at Boulder
Number of states in original machine : 8
Number of states in minimized machine : 8
CONTROLLO       pi= 6   po=10   nodes= 10       latches= 0
lits(sop)=   0  #states(STG)=   8
sis> state_assign jedi
Running jedi, written by Bill Lin,  UC Berkeley
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)= 168  #states(STG)=   8
sis> source script.rugged
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)= 168  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)= 168  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 13       latches= 3
lits(sop)=  79  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  72  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  72  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  72  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  72  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  72  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  70  #states(STG)=   8
CONTROLLO       pi= 6   po=10   nodes= 16       latches= 3
lits(sop)=  70  #states(STG)=   8
```

Il primo comando per minimizzare la macchina a stati finiti è stato "state_minimize stamina", il quale cerca di minimizzare gli stati utilizzando l'algoritmo di Paull-Unger per fsm completamente specificate. L'esito è stato ininfluente, infatti il numero degli stati è rimasto 8. Il secondo comando è stato "state_assign jedi". Esso non ottimizza la FSM ma serve a convertirla in un circuito sequenziale. L'ultimo comando è la chiamata a script.rugged che ha portato da 10 nodi a 16 nodi.

Mapping tecnologico

In quest'ultima fase del progetto vediamo come è possibile mappare il circuito ottimizzato su una libreria tecnologica. La consegna specifica di usare la "synch.genlib" integrata all'interno di SIS.
Si può scegliere di mappare il circuito ottimizzando l'area oppure il ritardo