UNIVERSITÀ
di VERONA

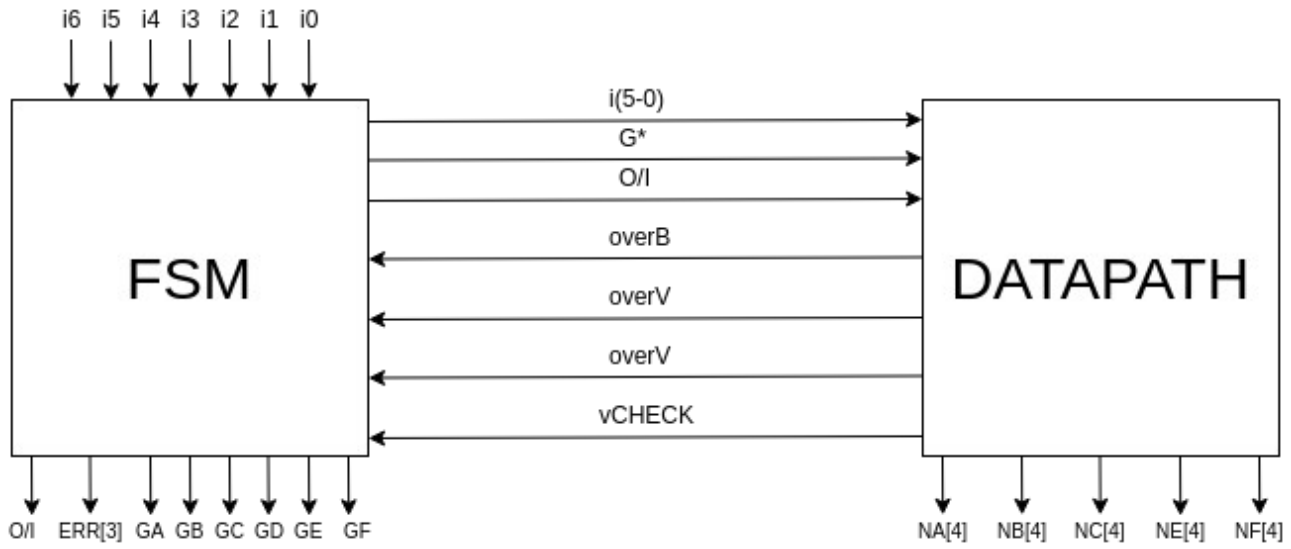
Elaborato SIS

Architettura degli Elaboratori

Timofte Robert Octavian - VR429581
Mirandola Giacomo - *VR429611*
Carra Mattia - VR429609

Indice

Architettura Generale del Circuito



Controllore

Il controllore è una macchina a stati finiti (FSM) di *Mealy.*

Inputs

Come **ingressi** abbiamo 11 bit, 7 dei quali assumono significato diverso in base allo stato nel quale ci si trova (i{6-0}), mentre gli altri quattro (aggiunti per scelta progettuale) sono:

- ***overB*** : Se vale 1 vuol dire che si è verificato un overload del deposito degli scarti del gate B.
  In caso contrario vale 0.

- ***overE*** *:* Se vale 1 vuol dire che si è verificato un overload del deposito degli scarti del gate E.
  In caso contrario vale 0.

- ***overV*** *:* Se vale 1 vuol dire che si è verificato un overload del serbatoio della macchina.
  In caso contrario vale 0.

- ***vCHECK*** *:* Se vale 1 vuol dire che il volume dopo la lavorazione è maggiore o uguale al volume prima della lavorazione.
  In caso contrario vale 0.

Outputs

Come **uscite** abbiamo 10 bit, suddivisi nel seguente modo:

- **O/I [1]** → Indica se la macchina è accesa o spenta (1 se accesa, 0 se spenta)
- **ERR [3]** → Rappresenta un codice di errore
- **GA [1]** → Comando per gestire il gate A (1 se aperto, 0 se chiuso)
- **GB [1]** → Comando per gestire il gate B (1 se aperto, 0 se chiuso)
- **GC [1]** → Comando per gestire il gate C (1 se aperto, 0 se chiuso)
- **GD [1]** → Comando per gestire il gate D (1 se aperto, 0 se chiuso)
- **GE [1]** → Comando per gestire il gate E (1 se aperto, 0 se chiuso)
- **GF [1]** → Comando per gestire il gate F (1 se aperto, 0 se chiuso)

Stati

- **A (*IMPIANTO OFF*)**:

  Stato di RESET nel quale la macchina è spenta.
  In questo stato i bit di ingresso assumono il seguente significato:
  ON2 ON1 ON0 - - - - - - - - -

- **B (*CHECK IN*)**:

  Stato che rappresenta il passaggio di un pezzo nella fase di ispezione iniziale.
  In questo stato i bit di ingresso assumono il seguente significato:
  QI VIN5 VIN4 VIN3 VIN2 VIN1 VIN0 - - - -

- **Bb (*NB++*)**:

  Stato nel quale vado a incrementare il contatore NB.
  In questo stato i bit di ingresso assumono il seguente significato:
  - - - - - - - overB - - -

- **C (*MILLING MACHINE*)**:

  Stato che rappresenta l'entrata del pezzo nella milling machine.
  In questo stato i bit di ingresso assumono il seguente significato:
  EM VOUT5 VOUT4 VOUT3 VOUT2 VOUT1 VOUT0 - - - vCHECK

- **D (*CHECK OUT*)**:

  Stato che rappresenta il passaggio di un pezzo nella fase di ispezione finale.
  In questo stato i bit di ingresso assumono il seguente significato:
  QO - - - - - - - - overV -

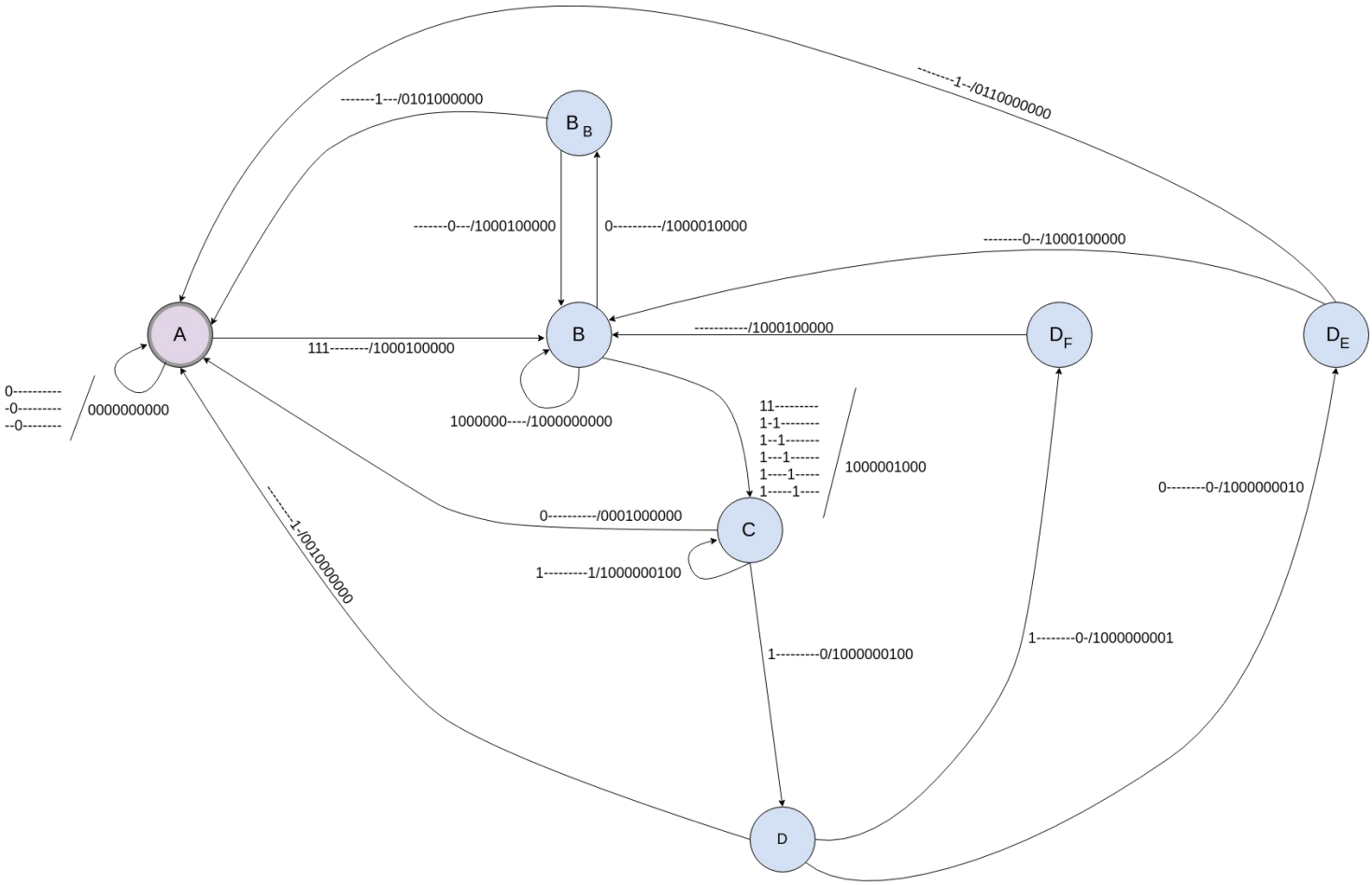
- **De (*NE++*)**:

  Stato nel quale vado ad incrementare il contatore NE.
  In questo stato i bit di ingresso assumono il seguente significato:
  - - - - - - - - overE - -

- **Df (*NF++*):**

  Stato nel quale vado ad incrementare il contatore NF.
  In questo stato i bit di ingresso assumono il seguente significato:
  \- - - - - - - - - - -

Schema FSM



Implementazione in SIS

```
.model FSM
.inputs i6 i5 i4 i3 i2 i1 i0 overB overE overV vCHECK
.outputs oi err2 err1 err0 ga gb gc gd ge gf

.start_kiss

.i 11
.o 10
.s 7
.p 23
.r A

0---------- A A 0000000000
-0--------- A A 0000000000
--0-------- A A 0000000000
111-------- A B 1000100000
1000000---- B B 1000000000
0---------- B BB 1000010000
-------1--- BB A 0101000000
-------0--- BB B 1000100000
11--------- B C 1000001000
1-1-------- B C 1000001000
1--1------- B C 1000001000
1---1------ B C 1000001000
1----1----- B C 1000001000
1-----1---- B C 1000001000
0---------- C A 0001000000
1---------1 C C 1000000100
1---------0 C D 1000000100
---------1- D A 0010000000
1--------0- D DF 1000000001
---------- DF B 1000100000
0--------0- D DE 1000000010
--------1-- DE A 0110000000
--------0-- DE B 1000100000

.end_kiss
```

Datapath

Il DATAPATH COMPLETO è formato da 6 altri datapath principali:

- 3 di questi servono per **incrementare** i contatori **NA, NC ed NF**

- 2 di questi servono per **incrementare** i contatori **NB ed NF**, i quali necessitano anche di un **controllo per l'overload**

- 1 serve per calcolare la **differenza** tra **VIN e VOUT** e controllare se **VOUT >= VIN**.
  Quest'ultimo datapath può essere diviso in quattro sezioni:
  - le prime due servono per ottenere i valori di VIN e VOUT
  - la terza serve per controllare se VOUT >= VIN
  - la quarta calcola la differenza tra VIN e VOUT e controlla se la quantità di scarto totale presente nel serbatoio è maggiore di 200

Inputs

- **i5** : sesto bit di input partendo da destra (viene utilizzato per il volume)
- **i4** : quinto bit di input partendo da destra (viene utilizzato per il volume)
- **i3** : quarto bit di input partendo da destra (viene utilizzato per il volume)
- **i2** : terzo bit di input partendo da destra (viene utilizzato per il volume)
- **i1** : secondo bit di input partendo da destra (viene utilizzato per il volume)
- **i0** : primo bit di input partendo da destra (viene utilizzato per il volume)
- **ga** : uscita GA della macchina
- **gb** : uscita GB della macchina
- **gc** : uscita GC della macchina
- **gd** : uscita GD della macchina
- **ge** : uscita GE della macchina
- **gf** : uscita GF della macchina
- **oi** : uscita O/I della macchina

Outputs

- **NA[4]** : contatore NA
- **NB[4]** : contatore NB
- **NC[4]** : contatore NC
- **NE[4]** : contatore NE
- **NF[4]** : contatore NF
- **overB[1]** : controllore overload NB

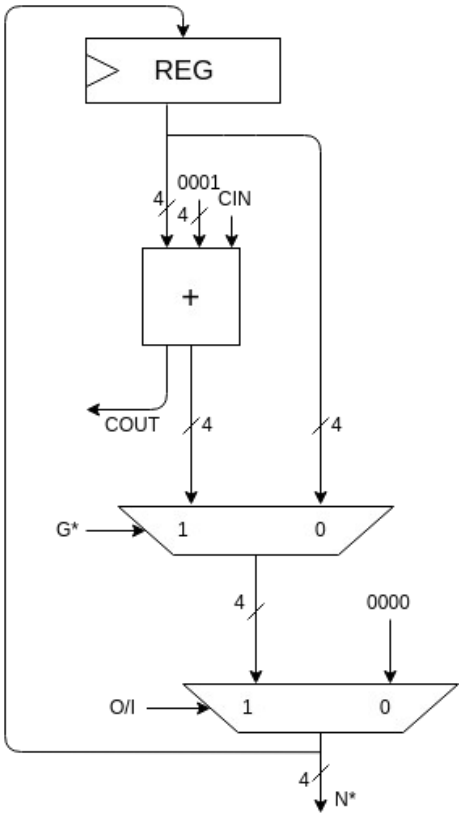
- **overE[1]** : controllore overload NE
- **overV[1]** : controllore overload serbatoio della macchina
- **vCHECK[1]** : controllore VOUT >= VIN

Architettura del Datapath

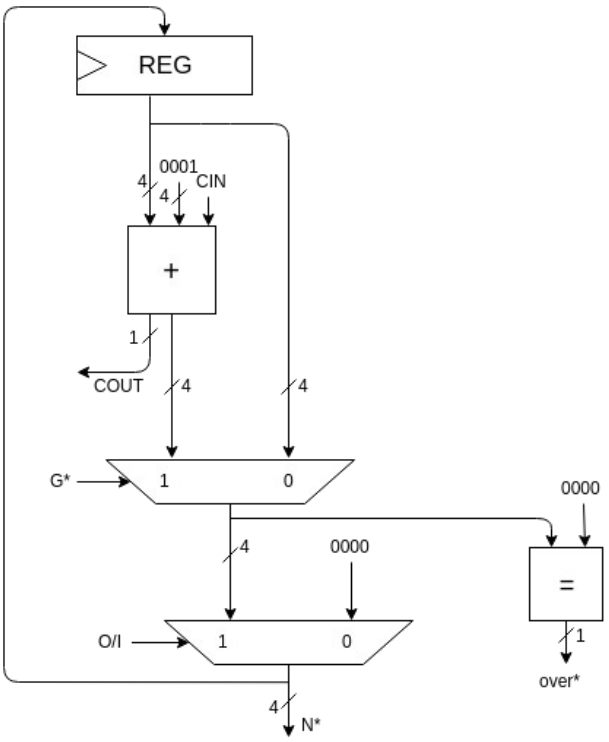
- ***5 registri a 4 bit***
- ***5 sommatori a 2 ingressi a 4 bit***
- ***10 multiplexer a 2 ingressi a 4 bit***
- ***2 comparatori di uguaglianza a 2 ingressi a 4 bit***
- ***2 registri a 6 bit***
- ***2 multiplexer a 2 ingressi a 6 bit***
- ***1 registro a 8 bit***
- ***1 negatore a 8 bit***
- ***2 sommatori a 2 ingressi a 8 bit***
- ***3 multiplexer a 2 ingressi a 8 bit***
- ***1 comparatore di maggioranza a 2 ingressi a 8 bit***
- ***1 comparatore di uguaglianza a 2 ingressi a 6 bit***
- ***1 comparatore di maggioranza a 2 ingressi a 6 bit***
- ***1 multiplexer a 2 ingressi a 1 bit***

Schema del Datapath

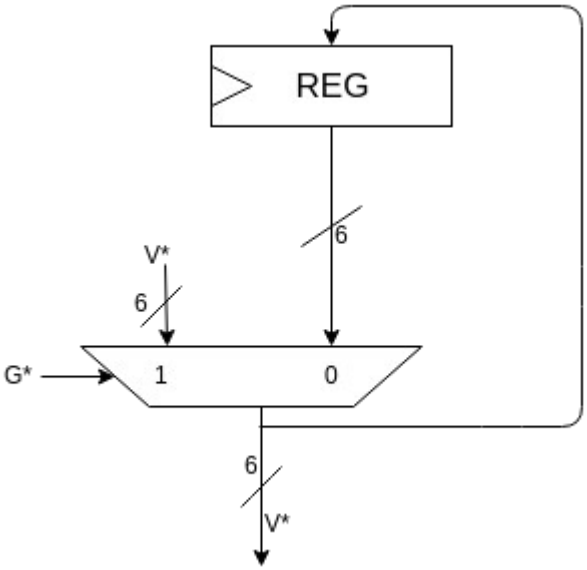
- ***contatore NA, NC, NF***



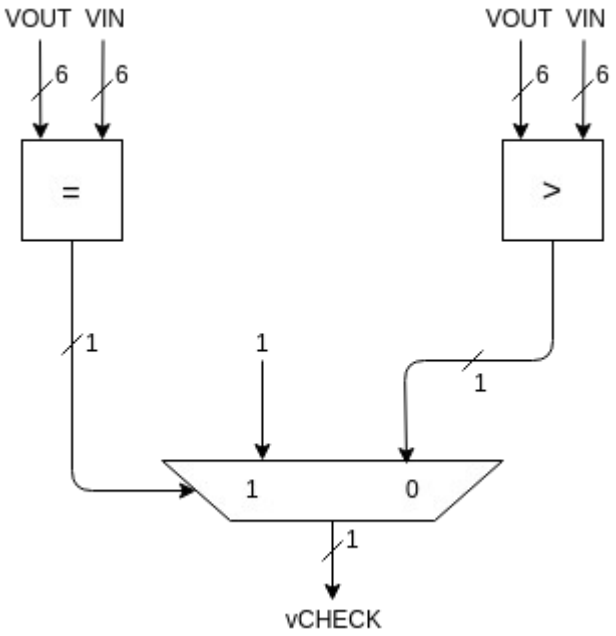
- ***contatore NB, NE***



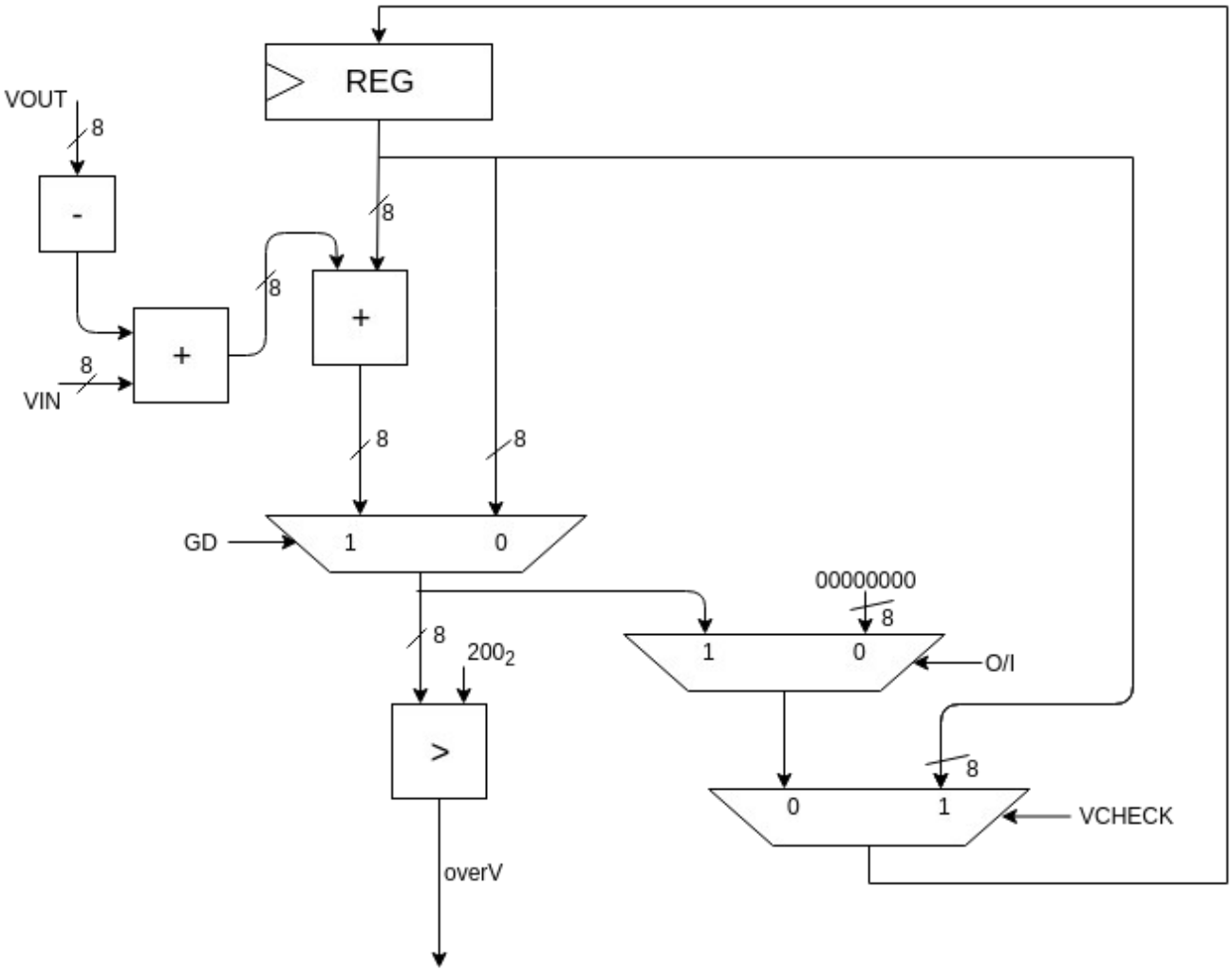
- ***VIN e VOUT (ritorna il valore rispettivo)***



- ***controllore VOUT >= VIN***



- calcolatore quantità scarto serbatoio



Funzionamento del Datapath

- Partiamo ad analizzare il datapath dei contatori **NA**, **NC** ed **NF**:

  Il registro a 4 bit memorizza il valore del contatore, il quale inizialmente vale 0, e lo passa ad un sommatore a due ingressi a 4 bit, il quale aggiunge 1 al contatore. Poi vi è un multiplexer il quale sull'entrata 1 riceve il valore incrementato, mentre su quella a 0 riceve il valore corrente non alterato.
  Ogni tal volta che il valore del gate dei contatori rispettivi vale 1, il multiplexer fornisce in entrata ad un altro multiplexer il valore incrementato. In caso il gate valga 0, viene fornito il valore non modificato del contatore. Questo secondo controllo serve per decidere quale valore fornire in output in base allo stato della macchina, o meglio, se essa è accesa o spenta.
  Se quest'ultima è spenta (O/I vale 0), viene restituito il codice 0000, se invece vale 1, viene fornito il valore del contatore.
  L'output (NA, NC o NF a seconda dei casi) viene poi memorizzato nel registro, andando a sovrascrivere il valore precedente.

- Analizziamo ora il datapath dei contatori **NB** ed **NE**:

  Essenzialmente la struttura è la stessa (le uscite saranno NB o NE), con l'unica differenza che consiste in un controllo, che avviene dopo il primo multiplexer, che serve per verificare se è stato generato un overload (uscite overB ed overE).

- Passiamo ora al datapath che calcola **VIN – VOUT** e controlla se **VOUT >= VIN**:

  Prima di tutto andiamo a calcolare la differenza tra VIN e VOUT.
  Per fare ciò bisogna negare VOUT e poi sommarlo a VIN.
  Il risultato di tale somma viene posto come primo operando in un altro sommatore, il quale va ad aggiungere il valore appena calcolato a quello memorizzato nel registro, che corrisponde alla quantità di scarto già memorizzata.
  Ogni volta che GD vale 1, viene fornito in uscita il nuovo valore della quantità, mentre se vale 0, viene fornito il valore memorizzato nel registro.

  L'output del multiplexer viene posto come primo input di un comparatore di maggioranza, il quale verifica se la quantità di scarto presente nel serbatoio è maggiore a 200.
  Se ciò avviene, overV assumerà il valore 1, in caso contrario sarà uguale a 0.

  L'uscita del multiplexer viene anche posta come ingresso in un altro multiplexer che serve per azzerare il valore nel registro quando la macchina si spegne.
  L'output di quest'ultimo viene posto in un altro multiplexer il quale restituisce il valore non incrementato della quantità di scarto se VOUT >= VIN.

  Per quanto riguarda il controllore *VOUT >= VIN*, come primo passaggio si controlla se VOUT = VIN. L'uscita di tale confronto viene posta come selettore in un multiplexer. Se S (selettore) vale 1, l'output, ovvero vCHECK, sarà uguale a 1, se invece S vale 0, l'uscita sarà uguale al valore restituito dal confronto di maggioranza tra VOUT e VIN.

  L'uscita del multiplexer che ha come selettore vCHECK va infine a sovrascrivere il valore del registro riguardante la quantità di scarto nel serbatoio.

I due datapath che vengono utilizzati per ottenere il valore di VIN e VOUT consistono semplicemente in un registro che memorizza il valore ed un multiplexer, il quale quando viene aperto il gate C (per VIN) o D (per VOUT), sovrascrive tale valore, oppure in caso contrario, lo restituisce.

Il datapath finale rappresenta un'unione dei circuiti appena descritti.

Implementazione in SIS

[(*) = utilizzato per creare componenti dello stesso tipo ma di dimensione maggiore]

- **na.blif** : contatore NA
- **nb.blif** : contatore NB
- **nc.blif** : contatore NC
- **ne.blif** : contatore NE
- **nf.blif** : contatore NF
- **vin.blif** : restituisce il valore di VIN
- **vout.blif** : restituisce il valore di VOUT
- **scarto.blif** : calcola la quantità di scarto
- **vol_control.blif** : controllore volumi (overload serbatoio e VOUT >= VIN)
- **DATAPATH.blif** : datapath completo
- **registro8.blif** : registro a 8 bit
- **registro6.blif** : registro a 6 bit
- **registro4.blif** : registro a 4 bit
- **registro.blif** : registro a 1 bit (*)
- **sommatore8.blif** : sommatore a due ingressi a 8 bit
- **sommatore4.blif** : sommatore a due ingressi a 4 bit
- **sommatore.blif** : sommatore a due ingressi a 1 bit (*)
- **mux8.blif** : multiplexer a due ingressi a 8 bit
- **mux6.blif** : multiplexer a due ingressi a 6 bit
- **mux4.blif** : multiplexer a due ingressi a 4 bit
- **mux.blif** : multiplexer a due ingressi a 1 bit (*)
- **uguale6.blif** : porta AND a due ingressi a 6 bit
- **uguale4.blif** : porta AND a due ingressi a 4 bit
- **xnor.blif** : utilizzato per creare le porte AND
- **maggiore8.blif** : comparatore di maggioranza a due ingressi a 8 bit
- **maggiore6.blif** : comparatore di maggioranza a due ingressi a 6 bit
- **xor.blif** : utilizzato per creare i comparatori di maggioranza

- **negatore8.blif** : negatore a 8 bit
- **negatore.blif** : negatore a 1 bit (*)
- **kzero4.blif** : costante zero a 4 bit
- **kuno4.blif** : costante uno a 4 bit
- **kduecento8** : costante duecento a 8 bit

Scelte progettuali

Oltre ad aver aggiunto 4 bit all'insieme degli input (pag.3), le altre scelte progettuali consistono in:

- non accettare mai un valore nullo per quanto riguarda i volumi.
Questo perché in termini logici, abbiamo pensato che un pezzo non può avere un volume uguale a 0.

- effettuare un controllo per controllare se VOUT è maggiore o uguale di VIN.
Questo perché sempre in termini logici, abbiamo pensato che dopo la lavorazione un pezzo non può avere lo stesso volume che aveva prima o tanto meno maggiore, quindi il suo nuovo volume deve essere minore di quello iniziale.

Statistiche del circuito

Il circuito dev'essere ottimizzato per area quindi limitare il numero di componenti. SIS offre vari comandi per eseguire queste operazioni (fx, full simplify, sweep, ecc.) ma lo script *script.rugged* è il più efficiente in quanto combina i programmi sopracitati.
Per la mappatura abbiamo utilizzato la libreria *synch.genlib* che tramite il comando *map –m 0 –s* mappa il circuito per area (*-m 0*) e visualizza le statistiche di ritardo e area massima (*-s*).

Pre-ottimizzazione:

```
>>> before removing serial inverters <<<
# of outputs:          73
total gate area:       17368.00
maximum arrival time: (82.20,82.20)
maximum po slack:     (-10.20,-10.20)
minimum po slack:     (-82.20,-82.20)
total neg slack:      (-4723.00,-4723.00)
# of failing outputs:  73
>>> before removing parallel inverters <<<
# of outputs:          73
total gate area:       16888.00
maximum arrival time: (81.00,81.00)
maximum po slack:     (-10.20,-10.20)
minimum po slack:     (-81.00,-81.00)
total neg slack:      (-4632.80,-4632.80)
# of failing outputs:  73
# of outputs:          73
total gate area:       15368.00
maximum arrival time: (80.80,80.80)
maximum po slack:     (-10.20,-10.20)
minimum po slack:     (-80.80,-80.80)
total neg slack:      (-4592.60,-4592.60)
# of failing outputs:  73
sis>
```

Ottimizzato:

```
>>> before removing serial inverters <<<
# of outputs:          73
total gate area:       16096.00
maximum arrival time: (79.60,79.60)
maximum po slack:     (-10.00,-10.00)
minimum po slack:     (-79.60,-79.60)
total neg slack:      (-4605.40,-4605.40)
# of failing outputs:  73
>>> before removing parallel inverters <<<
# of outputs:          73
total gate area:       16096.00
maximum arrival time: (79.60,79.60)
maximum po slack:     (-10.00,-10.00)
minimum po slack:     (-79.60,-79.60)
total neg slack:      (-4605.40,-4605.40)
# of failing outputs:  73
# of outputs:          73
total gate area:       15216.00
maximum arrival time: (79.60,79.60)
maximum po slack:     (-10.00,-10.00)
minimum po slack:     (-79.60,-79.60)
total neg slack:      (-4581.40,-4581.40)
# of failing outputs:  73
sis>
```