Comandi SIS

Ottimizzazione

- write_eqn → Stampa a video l'equazione del circuito caricato in sis

  (l'espressione è scritta in somma di prodotti e il simbolo "!" indica la negazione del letterale che lo segue)

- full_simplify → Ottimizza il circuito caricato in sis

  (è bene verificare l'operato di questo comando lanciano il comando write_eqn prima e dopo il full_simplify)

- print_stats → Stampa a video informazioni sul circuito:
  numero di segnali in input (PI),
  numero di segnali in output (PO),
  numero di nodi (nodes),
  numero di elementi di memoria (latches),
  numero di letterali (lits)

Per collegare fsm a fsmd

1. state_minimize stamina → Usa l'algoritmo stamina per minimizzare gli stati della FSM
2. state_assign jedi → Usa l'algoritmo jedi per effettuare automaticamente la codifica degli stati;
   costruisce anche le funzioni di stato prossimo e di uscita
3. stg_to_network → Costruisce le funzioni di stato prossimo e di uscita a partire dalla tabella delle transizioni e dalla codifica degli stati

creare file

4. write_blif → Visualizza la descrizione blif del circuito. (non apporta modifiche al file originale)
   a. write_blif *nome_file*.blif → crea file con le modifiche

altri comandi

- read_blif *nome_file*.blif → Carica la descrizione blif del circuito (compilare)
- simulate i0 i1 i2 … → Simula il circuito in base ai valori forniti per gli ingressi.
  Esecuzioni successive del comando considerano lo stato in cui il circuito si è portato dopo l'ultima esecuzione
  (next state → registri)
- write_kiss → Visualizza la tabella delle transizioni

Minimizzazione di circuiti combinatori multi-livello

- sweep → Esegue l'operazione di sweep

  (eliminazione dei nodi con un'unica linea di ingresso e di nodi con valore costante)

- eliminate *n* → Esegue l'operazione di eliminazione rimuovendo i nodi
  tali che la loro rimozione non aumenti il numero di letterali di una quantità superiore a "n" (numero intero).
  Per eliminare i nodi che sono utilizzati una sola volta utilizzare il valore -1

  (eliminazione di un nodo interno alla rete.
  il nodo N rappresenti la funzione y = (a + b) * c,
  l'eliminazione di N prevede la sostituzione della variabile y in tutti i nodi che la utilizzano con l'espressione booleana (a + b) * c)

- resub *lista* → Esegue l'operazione di scomposizione dei nodi indicati nella lista.
  Se la lista non viene specificata,
  la sostituzione viene eseguita per tutti i nodi della rete.
  I nodi nella lista devono essere specificati con il nome della loro uscita e vanno intervallati tra loro da uno spazio.

  (sostituzione di un nodo interno con un insieme di nodi la cui funzionalità sia equivalente a quella del nodo sostituito.
  L'operazione viene effettuata per diminuire la complessità di un nodo)

- fx → Esegue l'operazione di estrazione

  (estrazione di una sotto espressione comune a più nodi che viene rappresentata con un nuovo nodo)

- extract → estrazione di una sotto espressione comune a più nodi che viene rappresentata con un nuovo nodo

- simplify → riduzione della complessità di ogni singolo nodo con algoritmo di Quine-McCluskey

- full_simplify → Esegue l'operazione di semplificazione su ogni nodo della rete

- source *script* → Carica lo script ed esegue tutti i comandi contenuti al suo interno.
  Lo script che fornisce generalmente i risultati migliori è script.rugged:

script.rugged (file)

```
sweep; eliminate -1
simplify -m nocomp
eliminate -1

sweep; eliminate 5
simplify -m nocomp
resub -a

fx
resub -a; sweep

eliminate -1; sweep
full_simplify -m nocomp
```

- set autoexec *comando* → Stampa automaticamente il risultato del comando specificato dopo l'esecuzione di un qualunque altro comando.

Mapping tecnologico

- read_library *libreria* → Carica la libreria tecnologica di *nome_libreria*.
  Le librerie sono specificate nel formato genlib (estensione .genlib)
  Esempio:
  synch.genlib
  mcnc.genlib

- print_library → Visualizza informazioni inerenti alla libreria caricata;

- map → Esegue l'operazione di mapping.
  - −m 0 → permette di ottenere un circuito minimizzato rispetto all'area
  - −n 1 → permette di ottenere un circuito minimizzato rispetto al ritardo
  - −s → permette di visualizzare alcune informazioni relative ad area e ritardo dopo il mapping
  - total gate area → fornisce il valore dell'area come numero di celle standard della libreria tecnologica
  - maximum arrival time → indica il ritardo.

- write_blif −n → Mostra la rappresentazione del circuito associata alle porte della libreria

- print_delay → Stampa informazioni relative al ritardo del circuito

- reduce_depth → Riduce la lunghezza dei cammini critici

lanciare un test da terminale

scrivere *nome_file*.script

esempio

test.script (file esempio)

```
read_blif fsmd.blif
simulate 0 0 1 0 1
simulate 1 0 1 1 0
simulate 1 1 1 0 0
simulate 1 0 1 1 0
simulate 1 1 1 0 0
simulate 1 0 1 1 0
simulate 1 1 1 0 0
simulate 1 0 1 1 0
simulate 1 1 1 0 0
quit
```

- bash> sis -f *test*.script -x

visualizza output e next state di ogni simulate

Comandi “già pronti”

Sintesi della macchina a stati (fsm):

```
sis> read_blif nome_file.blif
sis> state_minimize stamina
sis> state_assign jedi
sis> stg_to_network
sis> write_blif nome_nuovo_file.blif
```

Sintesi dell’intero sistema (fsmd):

```
sis> read_blif nome_file.blif
sis> print_stats
sis> full_simplify
sis> print_stats
sis> source script.rugged
sis> print_stats
sis> write_blif ../FSMD.blif
```

Mapping del sistema sulla libreria tecnologica synch.genlib:

```
sis> read_library synch.genlib
sis> map -m 0 -s
```

testbench:

filtrare le uscite nel file *output_sis.txt*:

```
bash> sis -f testbench.script -x | grep Outputs: > output_sis.txt
```

verificare le differenze tra l’output del modello verilog e l’output del modello sis:

caso 1: `bash> diff output_sis.txt output_verilog.txt`

caso 2: `bash>`

(se il comando non stampa nulla, i due output sono equivalenti)