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

1. state\_assign jedi → Usa l’algoritmo jedi per

effettuare automaticamente la codifica degli stati;

costruisce anche le funzioni di stato prossimo e di uscita

1. 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

1. write\_blif → Visualizza la descrizione blif del circuito.

(non apporta modifiche al file originale)

* 1. 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)