Strumenti open per la sintesi logica

M. Favalli

Engineering Department in Ferrara

strumenti per la sintesi

Introduzione
BLIF
kiss2

Sommario

- Formati di ingresso BLIF e KISS
- Sistemi aperti per la sintesi e la verifica dei sistemi digitali (in laboratorio)
  - SIS
  - ABC

strumenti per la sintesi

Introduzione
BLIF
kiss2

Sommario

strumenti per la sintesi

Introduzione
BLIF
kiss2

SIS

- Ingresso: FSM, reti combinatorie e reti sincrone
- Reti asincrone (non considerate in questo corso)
- Mette a disposizione algoritmi per la sintesi e la verifica dei sistemi digitali
  - minimizzazione e codifica degli stati di FSMs
  - sintesi al livello logico (technology independent)
  - technology mapping per ASIC e FPGA
  - verifica formale
  - testing (marginale)
- Uscita: rete ottimizzata rispetto ad area o ritardo

ABC

- Simile a SIS
- Non contiene algoritmi per le FSM
- Utilizza algoritmi piú avanzati per la verifica e la sintesi
- Vantaggi notevoli nella verifica, marginali nella sintesi

BLIF (Berkeley Logic Interchange Format

- Linguaggio di descrizione dell'hardware ampiamente utilizzato nei sistemi aperti
- Consente di descrivere:
  - reti combinatorie come insiemi di equazioni logiche o in maniera strutturale utilizzando celle di libreria
  - reti sequenziali al livello strutturale
- Limitazioni rispetto a VHDL e Verilog

Struttura di un modello in BLIF

```
.model <name>
.inputs <list>
.outputs <list>
.clock <list>
# commento \ a capo
<command>
<command>
....
<command>
.end
```

Comandi

```
<logic_gate>
<generic_latch>
<library_gate>
<model_reference>
<subfile_reference>
<fsm_description>
<clock_constraint>
<delay_constraint>
```

Il linguaggio ammette un livello di descrizione gerarchica. I modelli dei componenti devono essere contenuti nello stesso file.

Descrizione di reti combinatorie

- Descrizione technology dependent

```
.names <in_1> <in_2> .... <in_n> <out>
<single output cover>
```

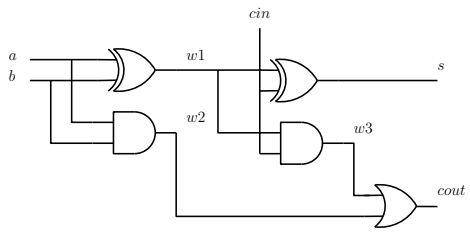
- La single-output cover é un espressione normale descritta tramite i cubi

```
# and gate
.names a b out
11 1
# carry out
.names a b ci co
11- 1
1-1 1
--1 1
```

- Descrizione di macro blocchi

Esempio a livello gate

```
.model full-adder
.inputs a b cin
.outputs s cout
.names a b w1
10 1
01 1
.names a b w2
11 1
.names w1 cin s
10 1
01 1
.names w1 cin w3
11 1
.names w2 w3 cout
1- 1
-1 1
.end
```



Esempio a livello macro

```
.model full-adder
.inputs a b cin
.outputs s cout
.names a b cin s
100 1
010 1
001 1
111 1
.names a b cin cout
11- 1
1-1 1
-11 1
.end
```

In entrambi i casi abbiamo semplicemente un grafo i cui vertici sono espressioni booleane

Descrizioni di tipo gerarchico

Riferimento a sottocircuiti

```
.subckt <model_name> <formal_actual_lits>
```

Sono ammessi diversi livelli gerarchici, anche se il formato interno con cui é rappresentato il circuito é flat

Esempio

Adder 4-bit

```
.model 4bit-adder
.inputs a0 a1 a2 a3 b0 b1 b2 b3 cin
.outputs s0 s1 s2 s3 cout
.subckt full-adder a=a0 b=b0 cin=cin s=s0 cout=c1
.subckt full-adder a=a1 b=b1 cin=c1 s=s1 cout=c2
.subckt full-adder a=a2 b=b2 cin=c2 s=s2 cout=c3
.subckt full-adder a=a3 b=b3 cin=c3 s=s3 cout=cout
.end
.model full-adder
.inputs a b cin
.outputs s cout
.names a b cin s
100 1
010 1
001 1
111 1
.names a b cin cout
11- 1
1-1 1
-11 1
.end
```

Flip-flop e latch

- Costrutto generic latch

```
.latch <input> <output> [<type> <clock>] [<init-val>]
```

- **fe** (falling edge), **re** (rising edge), **ah** (active high), **al** (active low), **as** (asynchronous)
- <init-val>: 0, 1, 2 (don't know 'X' in VHDL)
- Tutte le retroazioni avvengono tramite latch
- Notazione principalmente orientata alla sintesi

strumenti per la sintesi

Introduzione

BLIF

kiss2

SIS/ABC: input e output

- **read_blif** [<file_name>]
- **write_blif** [<file_name>]
- informazioni sulla rete **print_stats**

strumenti per la sintesi

Introduzione

BLIF

kiss2

Clock

strumenti per la sintesi

Introduzione

BLIF

kiss2

Simulazione

- SIS dispone di un semplice simulatore cycle accurate
  - é una tecnica di simulazione per circuiti sincroni in cui il timing non viene considerato esplicitamente, ma si suppone che i ritardi e il clock (non considerato esplicitamente) siano tali da consentire che latch e FF campionino segnali stabili
- **simulate <input values>**
- reti combinatorie e reti sequenziali sincrone
- nelle reti sequenziali mantiene l'informazione sullo stato presente
- si tratta di un utility di debug, per la validazione di un progetto si utilizzano VHDL e Verilog

Formato PLA

strumenti per la sintesi

Introduzione

BLIF

kiss2

BLIF: descrizione di libreria

Celle combinatorie

```
GATE <cell_name> <cell_area> <cell_logic_function> <pin_info>

<pin_info>:==<pin_name> <phase> <input_load> <max_load>
<rise_delay> <rise_fan_out_delay>
<fall_delay> <fall_fan_out_delay>
```

Ritardo di propagazione

$rise_{delay} = rise_{delay} + rise_{fan\_out\_delay} \times fan\_out$

$fall_{delay} = fall_{delay} + fall_{fan\_out\_delay} \times fan\_out$

strumenti per la sintesi

Introduzione

BLIF

kiss2

BLIF: descrizione di reti dopo il technology mapping

- La parte di linguaggio BLIF vista fino ad ora consente di descrivere reti in maniera indipendente dalla tecnologia (livello RTL: rete di blocchi che implementano qualsiasi funzione)
- Dopo il technology mapping la rete deve essere descritta in termini di celle di libreria interconnesse fra loro (livello gate: rete di blocchi che corrispondono a celle fisiche di libreria)

strumenti per la sintesi

Introduzione

BLIF

kiss2

Esempio

```
GATE inv1       1       O=!a;           PIN * INV 1 999 0.9 0.3 0.9 0.3
GATE inv2       2       O=!a;           PIN * INV 2 999 1.0 0.1 1.0 0.1
GATE inv3       3       O=!a;           PIN * INV 3 999 1.1 0.09 1.1 0.09
GATE inv4       4       O=!a;           PIN * INV 4 999 1.2 0.07 1.2 0.07
GATE nand2      2       O=!(a*b);       PIN * INV 1 999 1.0 0.2 1.0 0.2
GATE nand3      3       O=!(a*b*c);     PIN * INV 1 999 1.1 0.3 1.1 0.3
GATE nand4      4       O=!(a*b*c*d);   PIN * INV 1 999 1.4 0.4 1.4 0.4
GATE nor2       2       O=!(a+b);       PIN * INV 1 999 1.4 0.5 1.4 0.5
GATE nor3       3       O=!(a+b+c);     PIN * INV 1 999 2.4 0.7 2.4 0.7
GATE nor4       4       O=!(a+b+c+d);   PIN * INV 1 999 3.8 1.0 3.8 1.0
GATE and2       3       O=a*b;          PIN * NONINV 1 999 1.9 0.3 1.9 0.3
GATE or2        3       O=a+b;          PIN * NONINV 1 999 2.4 0.3 2.4 0.3
GATE xor        5       O=a*!b+!a*b;    PIN * UNKNOWN 2 999 1.9 0.5 1.9 0.5
GATE xor        5       O=!(a*b+!a*!b); PIN * UNKNOWN 2 999 1.9 0.5 1.9 0.5
GATE xnor       5       O=a*b+!a*!b;    PIN * UNKNOWN 2 999 2.1 0.5 2.1 0.5
GATE xnor       5       O=!(!a*b+a*!b); PIN * UNKNOWN 2 999 2.1 0.5 2.1 0.5
GATE aoi21      3       O=!(a*b+c);     PIN * INV 1 999 1.6 0.4 1.6 0.4
GATE aoi22      4       O=!(a*b+c*d);   PIN * INV 1 999 2.0 0.4 2.0 0.4
GATE oai21      3       O=!((a+b)*c);   PIN * INV 1 999 1.6 0.4 1.6 0.4
GATE oai22      4       O=!((a+b)*(c+d));PIN * INV 1 999 2.0 0.4 2.0 0.4
GATE zero       0       O=CONST0;
GATE one        0       O=CONST1;
```

Esempio

- Nel processo di sintesi una libreria puó essere letta con il comando `read_library <file name>`
- Una volta eseguita la sintesi la rete puó essere scritta con riferimento alle celle di libreria con `write_blif -s -n <file_name>`
- Il file puó essere letto con `read_blif` (una volta che sia stata letta la libreria)
- Simile al livello piú semplice di descrizione strutturale in VHDL

strumenti per la sintesi

Introduzione

BLIF

kiss2

Esempio

```
.model top
.inputs a b c d e f g h i j
.outputs k l m n o p q
.gate inv1 a=i O=v13
.gate inv1 a=j O=w13
.gate inv1 a=h O=x13
.gate aoi22 a=w13 b=h c=x13 d=j O=y13
.gate nand2 a=v13 b=y13 O=n
.gate nand2 a=n b=c O=a14
.gate nand3 a=w13 b=i c=h O=m
.gate aoi21 a=m b=h c=v13 O=c14
.gate or2 a=a b=b O=d14
.gate nor3 a=a14 b=c14 c=d14 O=e14
.gate inv1 a=d O=f14
.gate nor3 a=m b=e c=f14 O=g14
.gate nor2 a=i b=j O=h14
.gate nand4 a=x13 b=f c=i d=j O=i14
.gate nand2 a=i14 b=g O=j14
```

```
.gate nor4 a=e14 b=g14 c=h14 d=j14 O=k14
.gate inv1 a=k14 O=k
.gate inv1 a=c O=m14
.gate nand2 a=i b=j O=n14
.gate nand3 a=g b=n14 c=h O=q
.gate nor2 a=a b=b O=p14
.gate nand4 a=m14 b=h c=q d=p14 O=q14
.gate inv1 a=m O=r14
.gate nand3 a=r14 b=d c=e O=s14
.gate inv1 a=j14 O=t14
.gate nand4 a=q14 b=s14 c=n d=t14 O=l
.gate nand3 a=w13 b=x13 c=v13 O=v14
.gate inv1 a=v14 O=o
.gate nor3 a=j b=c c=h O=x14
.gate nand3 a=x14 b=v14 c=p14 O=p
.end
```

strumenti per la sintesi

Introduzione

BLIF

kiss2

Formato kiss2

Formato per la descrizione di FSM

```
.i <number of inputs>
.o <number of outputs>
.p <number of state transitions>
.s <number of states>
<state transition>
....
<state transition>
```

State transition

```
<input cube> <present state> <next state> <output cube>
```

Gli stati possono essere espressi sia in forma simbolica che codificati

strumenti per la sintesi

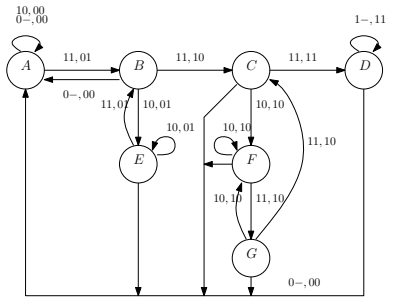
Introduzione

BLIF

kiss2

Esempio

FSM con due ingressi e due uscite che quando $x = 0$ produce 00 in uscita, mentre quando $x = 1$ tiene traccia del massimo numero (fino a 3) di uni consecutivi su y.




```
.i 2
.o 2
.s 8
.p 20
0- A A 00
10 A A 00
11 A B 01
11 B C 10
10 B E 01
0- B A 00
11 C D 11
10 C F 10
0- C A 00
1- D D 11
0- D A 10
11 E B 01
10 E E 01
0- E A 00
11 F G 10
10 F F 10
0- F A 00
11 G C 10
10 G F 10
0- G A 00
```