Technology mapping

M. Favalli

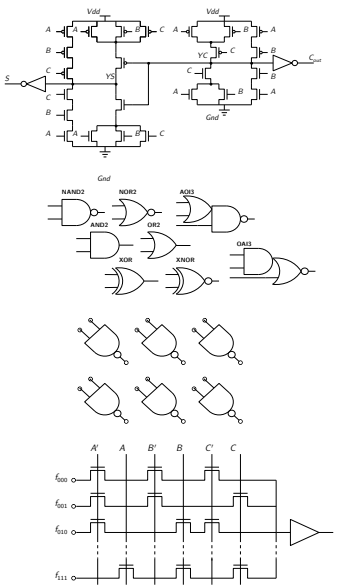
Engineering Department in Ferrara

Technology mapping

- Mappa una descrizione indipendente dalla tecnologia (Boolean network o AIG) di una rete ottimizzata su una libreria di celle
- Caratterizzazione della libreria (ASIC)
  - funzione
  - dimensioni
  - prestazioni
- Eventuali restrizioni
  - fan-in
  - fan-out
- Obbiettivi dell'ottimizzazione
  - delay, area. power, testability
- Problema di tipo NP completo

Librerie di celle

- Elemento base: transistore
- **Custom cell**: le celle possono essere sintetizzate a seconda del bisogno utilizzando anche diversi tipi di famiglie logiche (statiche, dinamiche ....)
- **Standard cell**: libreria fissa di funzioni
- **Gate array**: popolazione predefinita su silicio di gate
- **FPGA**: blocchi funzionali programmabili (look-up table nel caso combinatorio)



Sintesi di celle custom

- In teoria si potrebbe realizzare un blocco MOS per qualsiasi funzione
  - la cella risultante potrebbe essere piuttosto lenta (ad es. 20-input NAND gate)
  - il problema é evitabile limitando le dimensioni delle funzioni durante il processo di sintesi
  - limiti sul fan-in e fan-out delle celle
- Possibilitá di sfruttare appieno il dimensionamento dei transistori per soddisfare le specifiche sulle temporizzazioni
  - logical effort
  - velocizzazione dei cammini piú lunghi del circuito
  - regole specifiche per il tipo di tecnologia utilizzata
- Possibilitá di utilizzare contemporaneamente diversi tipi di logiche

Librerie di celle standard

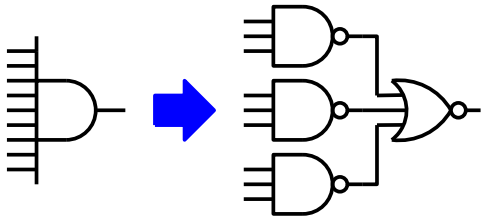
- Tipicamente realizzate in tecnologia CMOS (o FinFET) statica
- Gate standard NOT, BUFF, NAND (2-4), NOR (2-4), AND (2), OR (2), XOR (2), XNOR (2) (il numero tra parentesi indica il fan-in)
  - varie versioni della stessa funzione con diversi dimensionamenti dei transitori $\Rightarrow$ timing
- AOI (POS) e OAI (SOP): gate CMOS basati su reti serie e parallelo
- Elementi di memoria: latch e flip-flop
- Eventualmente celle piú complesse MPX, HA, FA progettate in maniera ottimizzata
- Librerie caratterizzate dal punto di vista di ritardi e consumo di potenza tramite simulazioni al livello elettrico

Graph covering

- Rappresentazione delle equazioni Booleane tramite DAG costituito da NAND a 2 ingressi e invertitori
- Rappresentazione non canonica (correlata ad AIG)
- Rappresentazione delle celle di libreria come DAG costituito da NAND a due ingressi e invertitori
- Si considerano tutte le possibili rappresentazioni di una cella ($n!$, ove n é il numero di ingressi)
- Ogni cella é caratterizzata dal suo costo

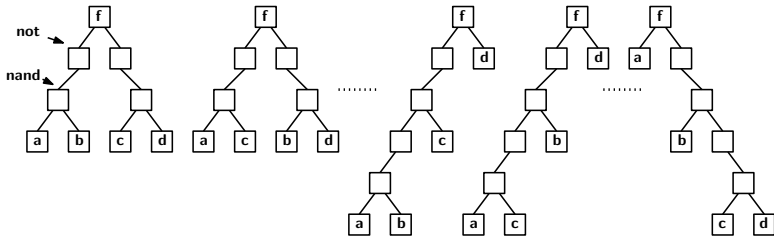
Rule based matching - IBM

- Applica alla rete una serie di trasformazioni basate su regole
  - le regole codificano l'esperienza dei progettisti
  - regole diverse per area e prestazioni
- Le regole non garantiscono un ottimalitá della soluzione (ma questo vale pure per gli euristici) e una sufficiente esplorazione dello spazio delle possibili soluzioni
- Esempio:



Esempio di caratterizzazioni di cella

NAND a 4 ingressi $f = (abcd)'$



Graph covering

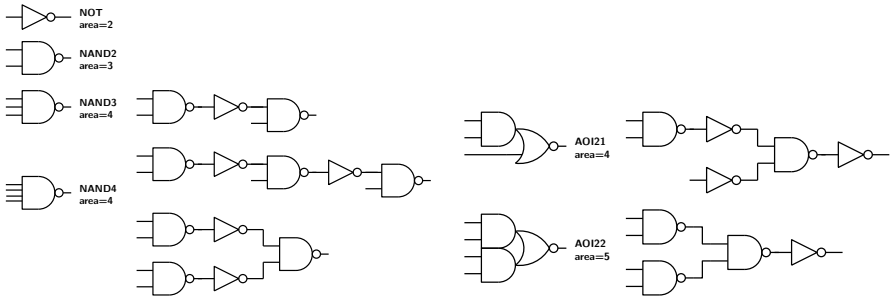
- Copertura di costo minimo del grafo (subject) corrispondente alle equazioni del circuito con i grafi (pattern) che descrivono le equazioni delle porte logiche della libreria
- Problema NP-completo che é simile alla generazione di codice da parte di un compilatore
- Approcci utilizzati
  - ricerca a partire dai primary inputs
  - ricerca a partire dai primary outputs
  - tentare prima con i DAGs delle celle piú grandi (si noti che le celle grandi possono portare a risparmi di area con conseguenti problemi di prestazioni)
- Per le celle si utilizzano rappresentazioni ad albero di tipo pseudo-canonico

Note sul graph covering

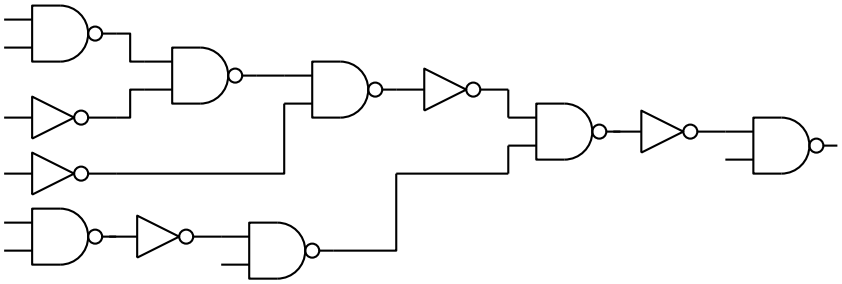
- Ogni nodo del subject graph deve essere coperto da almeno un pattern
- Alcuni nodi possono essere coperti da piú pattern → cambiamenti nel fan-out rispetto al grafo di partenza
- Gli ingressi di un pattern possono venire solo da uscite di un altro pattern e non da archi interni

Esempio di celle di libreria (pattern graph)

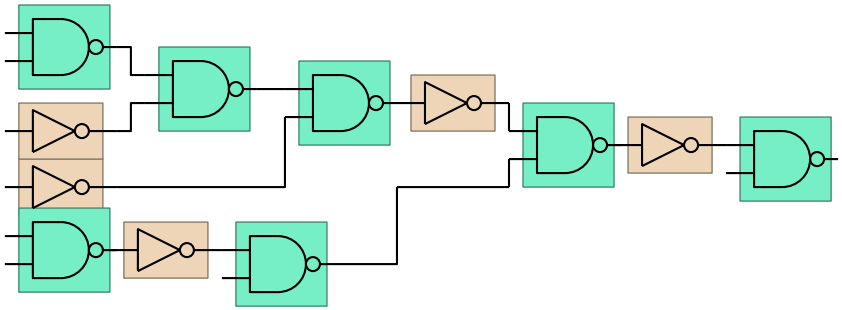
Si osserva che ad alcune celle (NAND4) possono corrispondere piú rappresentazioni



Esempio di grafo da coprire (subject graph)

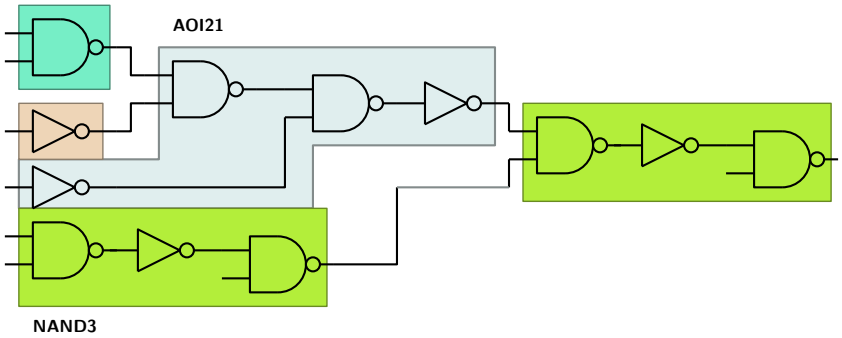


Copertura banale



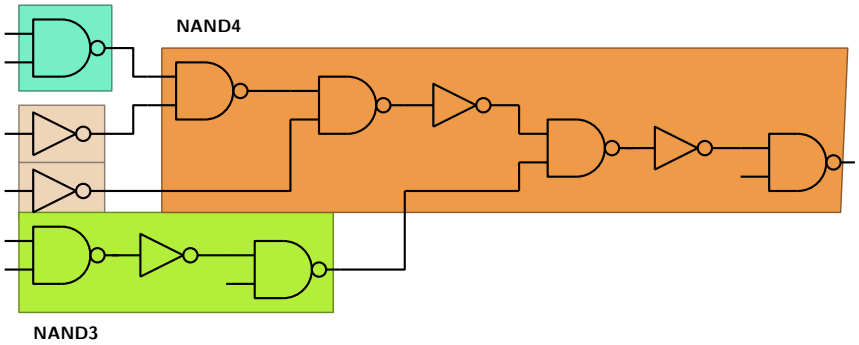
Costo: 7 NAND2 + 5 INV = 31

Secondo tentativo



Costo: 1 NAND2 + 2 NAND3 + 1 INV + 1 AOI21 = 3 + 8 + 2 + 4 = 17

Primo tentativo



Costo: 1 NAND2 + 1 NAND3 + 2 INV + 1 NAND4 = 3 + 4 + 4 + 5 = 16

Tree pattern matching

- Il graph covering puó avere dei costi esponenziali
- Si utilizzano allora degli euristici che garantiscono una maggior efficienza computazionale
- Chiaramente, danno luogo a soluzioni parzialmente ottimali
- Se il subject graph e i pattern graph sono alberi, si puó trovare una soluzione ottima

Problema

- Forma normale: rete di NAND a 2 ingressi e invertitori con gli alberi orientati verso sinistra (il primo ingresso vede il minimo numero di livelli logici, o viceversa)
- Rappresentazione della rete in forma normale sia per il subject DAG che per il pattern DAG
- Ciascun DAG ha un costo
- Obbiettivo: trovare una copertura (matching) di costo minimo (minimizzazione)

Copertura ottimale di alberi

Proprietá

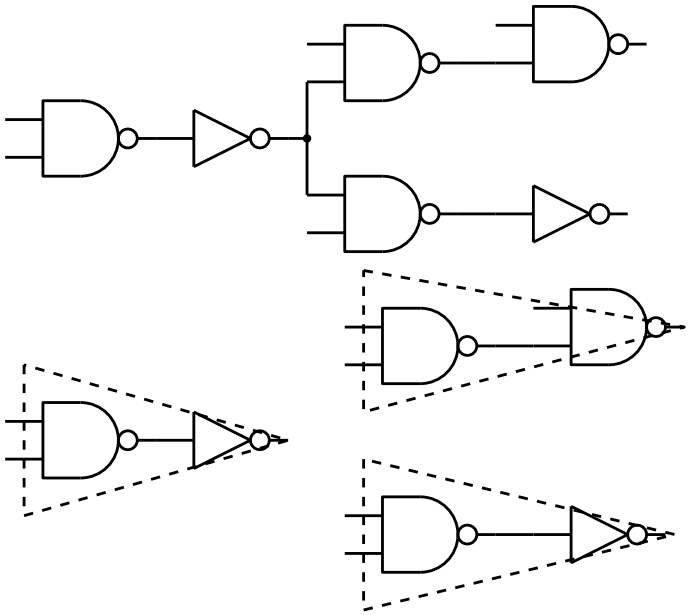
La copertura di una rete ad albero é ottima se consiste nella migliore copertura per la radice dell'albero e nelle coperture ottime dei sottoalberi che partono dagli ingressi della copertura della radice

- Questo consente di utilizzare tecniche di dynamic programming
- Le reti di partenza peró non sono alberi

Tree pattern matching

- Partiziona il grafo diretto aciclico del circuito in un albero tramite lo split dei nodi di fan-out
- Algoritmo per la ricerca di una copertura ottimale per un albero (subject tree)
  1. per ogni nodo dell'albero si cercano tutti matching fra i sottoalberi che partono dai nodi del subject tree e gli alberi della libreria (pattern matching)
  2. viene selezionato un matching ottimale per ciascun nodo (tree covering)
- Queste operazioni vengono eseguite in maniera recursiva
  1. cerca un matching per l'uscita
  2. cerca un mapping ottimale per le ciascun sottoalbero che parte dagli ingressi delle cella giá mappata
  3. il costo si ottiene sommando quello della cella di uscita piú il costo delle celle di ingresso
- Tempo esponenziale

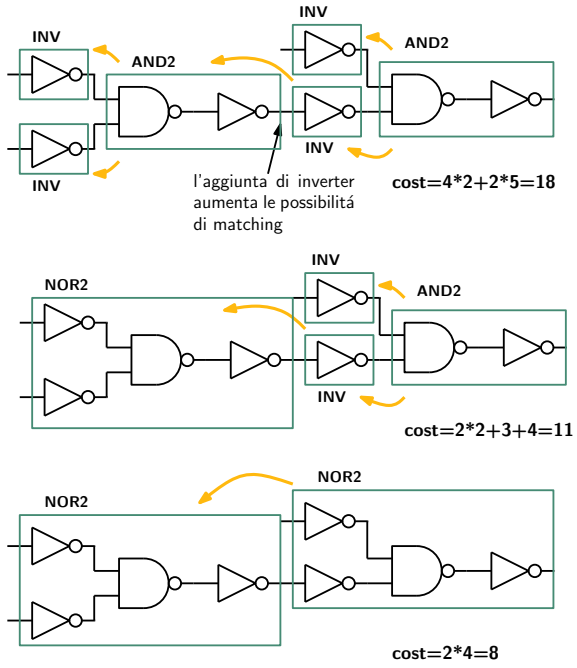
Fan-out splitting



Tree pattern matching

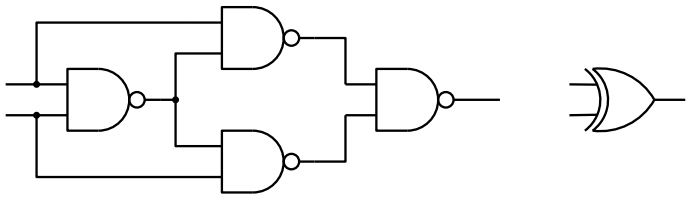
```
OptimalTree(tree)
{
  mincost = INF;
  for all cells
    if (cell matches at tree.root) {
      cost = cell.cost;
      for all cell inputs
        cost += OptimalTree(cell.input[i]);
      if (cost < mincost) {
        mincost = cost;
        keep tree mapping;
      }
    }
  return(mincost);
}
```


Esempio



Graph matching

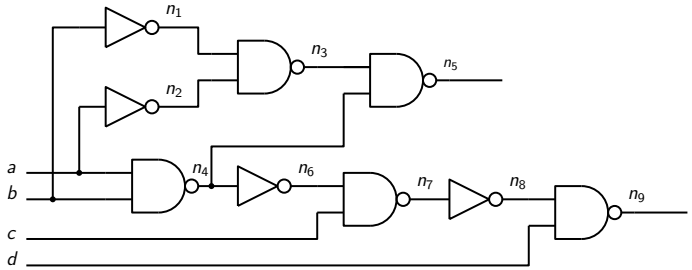
- Il tree matching ha limitazioni nel caso di sotto-espressioni comuni o di celle di libreria che non possono essere descritte come alberi
  - XOR e MUX
  - gate con piú uscite
- Nel graph matching si evita di decomporre il circuito in foreste di alberi
- Il match avviene con sotto-grafi e non con alberi



Algoritmo

- Per ogni nodo del subject graph n_i, $i = 1..N$, trova tutte le possibili celle di libreria il cui grafo ha un matching con un sottografo avente tale nodo come radice
- Si produce l'insieme di tutti i possibili matching con i pattern m_j, $j = 1..M$
- Complessitá $O(C \times N)$ (dove C é il numero di celle di libreria)
- Problema di copertura dei nodi del grafo del circuito con grafi in modo da minimizzare il costo delle celle utilizzate (non si puó utilizzare il dynamic programming)
  - matrice di copertura X con una riga per ogni n_i e una colonna per ogni m_j: $x_{ij} = 1$ se m_j copre il nodo n_i e 0 altrimenti
  - tabella Y che descrive i matching m_j dal punto di vista del nodo radice, delle foglie (ovvero l'uscita e gli ingressi della cella di libreria corrispondente), del nome della cella e del costo

Esempio



$X =$

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}
n_1	1	0	0	0	0	0	0	0	0	0	0	1	0	1
n_2	0	1	0	0	0	0	0	0	0	0	0	1	0	1
n_3	0	0	1	0	0	0	0	0	0	0	0	1	0	1
n_4	0	0	0	1	0	0	0	0	0	0	1	1	1	0
n_5	0	0	0	0	1	0	0	0	0	0	0	1	0	1
n_6	0	0	0	0	0	1	0	0	0	0	1	0	1	0
n_7	0	0	0	0	0	0	1	0	0	1	1	0	1	0
n_8	0	0	0	0	0	0	0	1	0	1	0	0	1	0
n_9	0	0	0	0	0	0	0	0	1	1	0	0	1	0

match	output	inputs	cell	cost
m_1	n_1	b	inv	1
m_2	n_2	a	inv	1
m_3	n_3	n_1, n_2	nand2	2
m_4	n_4	a, b	nand2	2
m_5	n_5	n_3, n_4	nand2	2
m_6	n_6	n_4	inv	1
m_7	n_7	n_6, c	nand2	2
m_8	n_8	n_7	inv	1
m_9	n_9	n_8, d	nand2	2
m_{10}	n_9	n_6, c, d	nand3	3
m_{11}	n_7	a, b, c	nand3	3
m_{12}	n_5	a, b	xnor	5
m_{13}	n_9	a, b, c, d	nand4	4
m_{14}	n_5	a, b, n_4	aoi21	3

Algoritmo

- Si cerca un insieme di colonne indipendenti della tabella di copertura che sia di costo minimo
- Strategia di branch and bound sulla matrice o formulazione del problema come Pseudo Boolean Satisfiability
- Apparentemente sembrerebbe un problema di copertura simile a quello incontrato nella sintesi di reti a 2 livelli
- Sono presenti altri vincoli che rendono la rete consistente: **se utilizzo una cella, gli ingressi di quella cella devono essere generati da altre celle nella copertura utilizzata**
- Il problema di copertura rimane *NP*-completo, ma é piú generale di quello incontrato nella sintesi di reti a 2 livelli

Formulazione Pseudo-booleana

- Si associa una variabile Booleana a ciascun match m_i, per comoditá utilizzeremo lo stesso simbolo
- Se $m_i = 1$ la cella corrispondente al match viene utilizzata nella copertura se $m_i = 0$ non viene utilizzata
- Si costruisce un espressione CNF esplicitando le condizioni di copertura per ogni nodo
  - nell'esempio il nodo n_7 é coperto da m_7, m_{10}, m_{11} e m_{13}, e quindi la condizione di copertura é data da questa clausola:

$$(m_7 + m_{10} + m_{11} + m_{13})$$

- Nell'esempio si ha:

$$\Phi_{cov} = (m_1 + m_{12} + m_{14})(m_2 + m_{12} + m_{14})(m_3 + m_{12} + m_{14})$$
$$(m_4 + m_{11} + m_{12} + m_{13})(m_5 + m_{12} + m_{14})(m_6 + m_{11} + m_{13})$$
$$(m_7 + m_{10} + m_{11} + m_{13})(m_8 + m_{10} + m_{13})(m_9 + m_{10} + m_{11} + m_{13})$$

Vincoli di consistenza

- Ad esempio, se utilizzo il match m_3, allora devo utilizzare celle che producano m_1 e m_2 e quindi ho la condizione
  $(m_3 \rightarrow m_1 m_2) = (m_3 \rightarrow m_1)(m_3 \rightarrow m_2) = (m_3' + m_1)(m_3' + m_2)$
- Complessivamente i vincoli di consistenza nell'esempio sono

$$\Phi_{cons} = (m_3' + m_1)(m_3' + m_2)(m_5' + m_3)(m_5' + m_4)$$
$$(m_7' + n_6)(m_8' + m_7 + m_{11})(m_9' + m_8)(m_{10}' + m_6)(m_{14}' + m_4)$$

- Questa espressione va in AND con Φ_{cov}
- La presenza di variabili negate rende il problema di copertura binato (nella sintesi a due livelli era unato)

Soluzione mediante PB optimizer

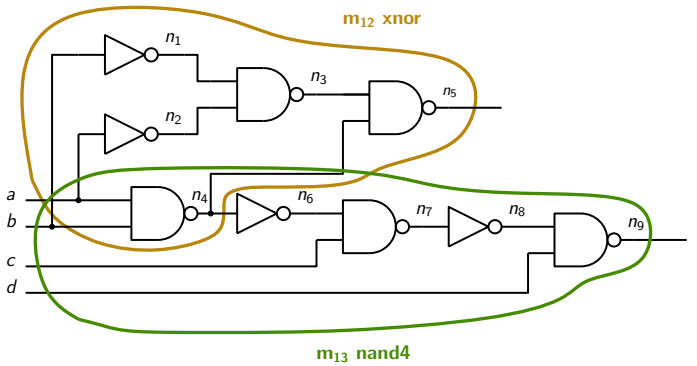
- Il problema é descritto da:
  - una CNF $\Phi = \Phi_{cov}\Phi_{cons}$ che deve essere soddisfatta da un opportuno assegnamento delle variabili m_i
  - una funzione obbiettivo da minimizzare $W = \sum_{\forall i} w_i m_i$ (si interpreti la somma come algebrica e w_i come il costo della cella corrispondente a m_i)
- Tale problema si dice Pseudo-Booleano e puó essere risolto con tecniche relativamente efficienti
- Un esempio di solver/optimizer per questi problemi é dato da `minisat+`

Miglioramenti

- Il matching basato su grafi e sottografi presenta rilevanti limitazioni dovute alla necssitá di ricondurre sia il grafo subject che i grafi pattern a qualche forma pseudo-canonica
- Il matching Booleano cerca di eliminare tali problemi determinando una corrispondenza fra le funzioni di sotto-reti e quelle delle celle
- Sembrerebbe un problema di verifica, ma in realtá é piú complesso perché non conosciamo l'ordinamento degli ingressi e vogliamo utilizzare la presenza di invertitori in ingresso e in uscita come gradi di libertá
- Utilizzo di classi di funzioni equivalenti NPN
- Utilizzo di signature delle funzioni per eliminare soluzioni sicuramente non fattibili
  - esempio di signature (limitatamente alla classe P): il numero di 1 in uscita

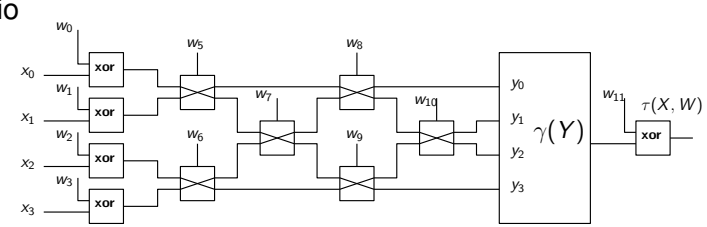
Soluzione

La soluzione fornita dal solver é
$m'_1 m'_2 m'_3 m'_4 m'_5 m'_6 m'_7 m'_8 m'_9 m'_{10} m'_{11} m_{12} m_{13} m'_{14}$ con costo $W = 9$



Boolean matching

- Si vuole determinare se $\varphi(X)$ é NPN equivalente a $\gamma(Y)$
- In pratica si vuole vedere se la formula $\varphi(X) = N_{out}(\gamma(P(N_{in}(X)))$ puó essere vera per qualche trasformazione N_{out}, P e N_{in}
- L'idea é semplificata in questo schema dove per $|X| = 4$ si vedono le reti che realizzano tali trasformazioni in funzione di un insieme di parametri $W = \{w_0, ..., w_{11}\}$ che le codificano in binario



- Quindi si tratta di verificare la QBF $\exists W,\ \forall X,\ \varphi(X) = \tau(X, W)$

Esercitazioni

- technology mapping con SIS e ABC
- dipendenza della qualitá dei risultati ottenuti dalle dimensioni della libreria utilizzata

Esempi di strutture di FPGA

- Xilinx
  - blocchi logici configurabili da RAM (CLB)
    - logica combinatoria (LUT): 2 funzioni di 4 variabili o 1 di 5 variabili
    - elementi di memoria
  - switch programmabili tramite RAM che realizzano il wiring
- Actel
  - elementi logici programmabili da fusibili
  - interconnessioni programmabili da fusibili
  - tutte le funzioni di 2 o 3 variabili
  - alcune funzioni di 4 variabili

FPGA technology mapping

- Blocchi logici programmabili
  - multiplexer based (ACTEL)
  - lookup table based (XILINX)
- Problema
  - non si possono utilizzare gli approcci visti per gli ASIC perché una LUT a k ingressi realizza 2^{2^k} possibili ingressi
  - valori tipici per k, 4 o 5
  - il numero di celle di libreria da considerare rsiulterebbe non pratico
- Soluzioni
  - bin packing
  - OBDD matching

Bin packing

- É un problema di ottimizzazione ben noto
- Dati n oggetti ciascuno dei quali i con dimensione a_i e dei contenitori ciascuno dei quali puó contenere una dimensione k, si tratta di determinare il numero di contenitori necessari a contenere tutti gli oggetti
- Problema NP completo

Bin packing

Si vede ogni blocco logico configurabile come un bin

- i nodi della rete booleana vengono visti come espressioni SP
- si cerca di inserire i prodotti nel minor numero possibile di scaffali
- inserisce il termine somma in uno o piú CLB che lo possa contenere

Euristico

- ordina i termini prodotto in ordine di dimensione (numero di letterali) decrescente
- li inserisce nel primo CLB in cui c'é spazio sufficiente
- complessitá $O(NlogN)$ (dove N é il numero di letterali dell'espressione SP) con risultati peggiori del 22% rispetto all'ottimo

Esempio