Synthesis and optimization of domino logic

Min Zhao and Sachin Sapatnekar
Department of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

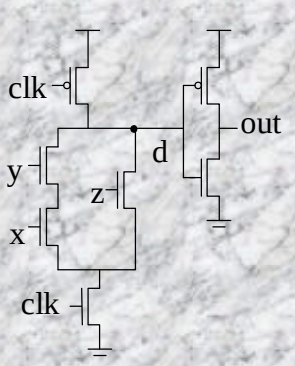
1

Outline

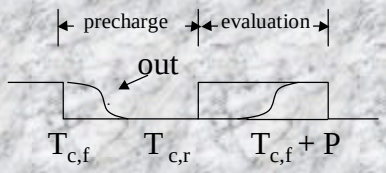
- **Introduction to domino logic**
- **Domino logic synthesis flow**
- **Technology mapping of domino logic**
- **Timing-driven static-domino partitioning**

2

Basics of domino logic



d: *dynamic node*



3

Advantages of domino logic

- **Speed advantages**
  - ***Reduced fighting during transitions***
  - ***Fewer transistors per gate, lower capacitive load***
- **Area advantages**
  - ***Mainly consists of NMOS***
  - ***N+4 transistors instead of 2N transistor per gate***
- **Therefore, domino logic is widely used in high-performance circuit design.**

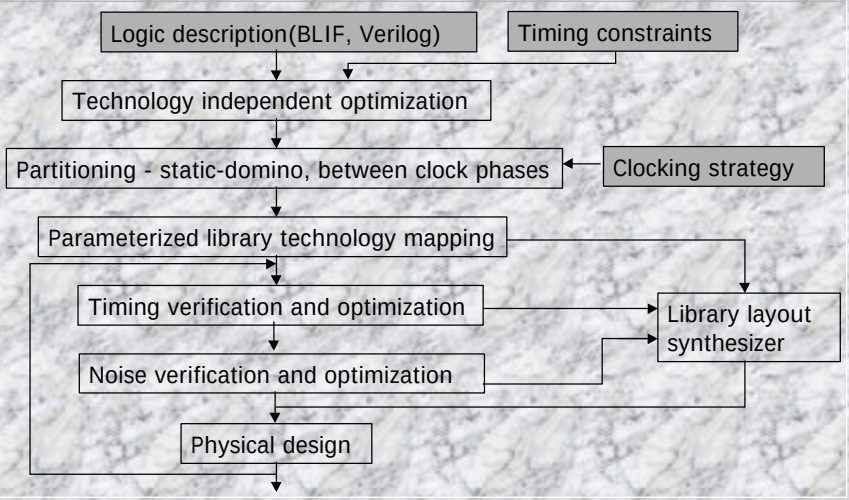
4

Disadvantages of domino logic

- **Disadvantages**
  - ***Non-inverting nature may require logic duplication***
  - ***Strict timing constraints***
  - ***Charge sharing, noise susceptibility***
  - ***High clock routing overhead***
- **Need automated techniques considering these issues for domino circuit design**

5

Domino logic synthesis flow

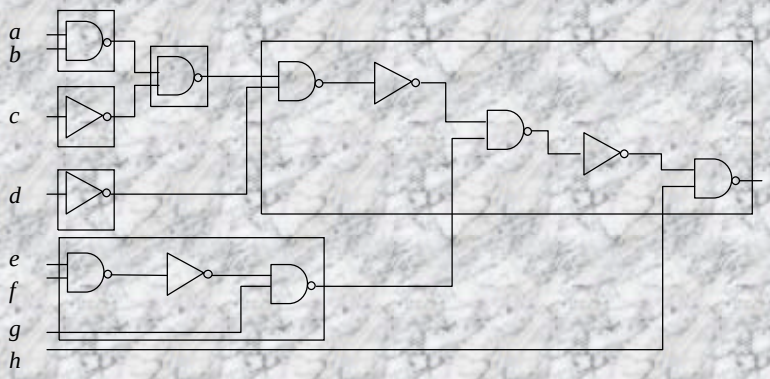


6

Technology mapping of domino logic

What is technology mapping?

- **Implement input network with gates in a library.**



Parameterized library

- **Large NMOS pull-down network of domino gate.**
  - ***Small short circuit current and small driven load.***
  - ***No complementary part.***
  - ***The delay overhead of inverter may offset the advantage of fast switch speeds in small gates.***
- **Dramatical increase of library number with the increase of length(s) and width(p) of gate.**
  - ***(s,p): (3,6): 6877; (4,4): 3503; (4,6): 222943***
- **A parameterized library is applied for technology mapping of domino logic.**

9

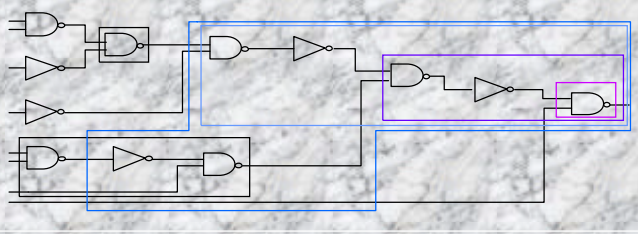
Problem definition

- **A parameterized library**
  - **A collection of gates that satisfy the constraints on the width and height of the pull-down(pull-up) implementation of a gate.**
  - **Cell layout produced on the fly**
- **Technology mapping of domino logic**
  - ***Given***
    - **An optimized Boolean network**
    - **A constraint on the width and height of domino gates**
  - ***Find***
    - **Minimum cost solution to the problem that nodes in the network are implemented in domino logic**

10

General technology mapping algorithm

- Dynamic programming algorithm is applied.
- At each network node
  - *pattern matching*
  - *cost calculation for each possible matching*
- The cost will be large if the library is large.



11

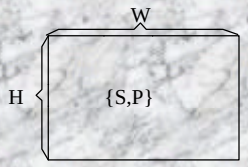
Parameterized library mapping algorithm

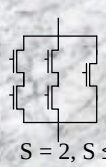
- Starting point
  - Given an arbitrarily optimized network
  - It is first unated
  - Then mapped into a two input AND-OR DAG
  - Then the DAG is decomposed into trees.
- Complexity
  - *space complexity: O(WHN)*
  - *time complexity: $O(W^2H^2N)$*
    - W: maximum number of parallel chains
    - H: maximum number of series transistors
    - N: number of nodes in the tree

12

Subsolutions

- **Subsolution space at each node.**



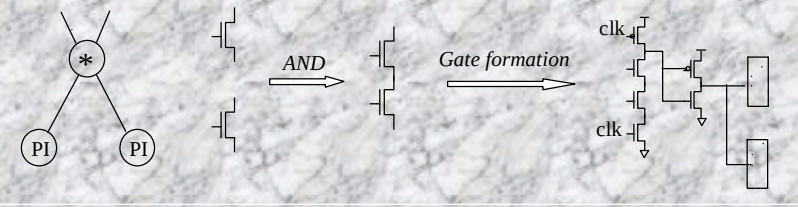


S = 2, S ≤ H
P = 3, P ≤ W

- **Each stored subsolution is optimal for its subtree under specified constraints**
- **Physically,**
  - ***{S,P}(S³1 & P ³ 1) represents a segment of a domino pull-down whose height and width are S and P***
  - ***{1,1} represents a complete domino gate or a PI.***

Basic Operations

- **OR operation: $S=\max(S_l, S_r)$, $P=P_l+P_r$**
- **AND operation: $S=S_l + S_r$, $P=\max(P_l, P_r)$**
- **PI / Gate formation operation: S=1, P=1**
  - ***A gate formation operation corresponds to a situation where the structure collected so far is converted to a domino gate with an output at that network node.***



Node data structure

- Store the optimal subsolutions for all possible [height, width] combinations from [1,1] to [H,W].
- Each optimal subsolution can be represented as {S, P, C, {S_l, P_l}, {S_r, P_r}}
  - S (1 £ S £ H) is the maximum height of the current solution.
  - P (1 £ P £ W) is the maximum width of the current solution.
  - C is the cost.
  - {S_l, P_l}, {S_r, P_r} is the subsolutions of left and right child whose combination provides the minimal cost of subsolution {S,P}

15

Node data calculations

- {S, P} (S ³ 1 & P ³ 1) subsolution at a parent node is obtained by combining optimal subsolutions at child nodes.
- {1, 1} subsolution at a node is obtained from the subsolution of the same node whose cost is minimal.
- The procedure consists of
  - *Node constraint functions*
  - *Node cost functions*

16

Node cost functions

- **Here, cost is area -- the number of transistors.**
- **Literal operation: C=C+1**
  - ***Literal operation corresponds to a primary input or a situation where a new domino structure is started after gate formation operation.***
- **OR/AND operation: C=Literal(C_l) + Literal(C_r)**
- **Gate formation operation: C=C_{min} +4**
  - ***The minimal cost solution, C_{min} is the minimal value out of all H*W optimal subsolutions***
  - ***'4' includes two clock control transistors + an inverter***

17

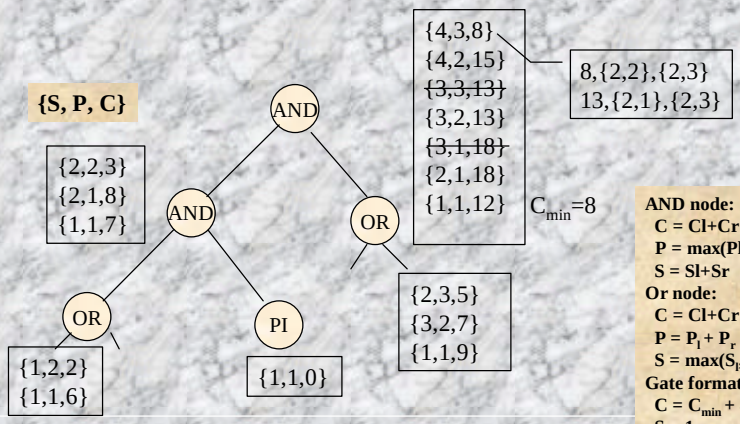
Node mapping algorithm

For each valid [height width] subsolution of the left child {
for each valid [height width] subsolution of the right child{
{S,P}= *Node constraint functions* ({S_l, P_l}, {S_r, P_r});
if {S, P} was within the constraints (H, W)
{
C = *Node cost functions* (C_l, C_r)
if (C<C[S,P]$_{min}$) then C[S, P]$_{min}$ = C.
if (C<C_{min}) then C_{min} =C.
}
}
}
C[1,1] = *Gate formation* (C_{min})

18

An example

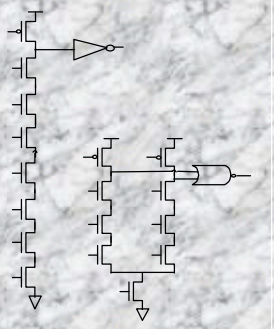
- Of all (S,P) mapping subsolutions for the children only those with minimal cost are stored



AND node:
C = Cl+Cr
P = max(Pl,Pr)
S = Sl+Sr
Or node:
C = Cl+Cr
$P = P_l + P_r$
$S = \max(S_l, S_r)$
Gate formation:
$C = C_{min} + 4$
S = 1
P = 1

Wide domino gate

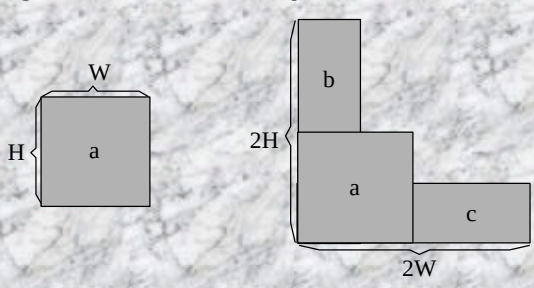
- **NAND, NOR gate can be used to replace inverter.**
  - ***Break up large stacks of series transistors into parallel chains***



20

Wide AND/OR domino gate mapping

- Enlarged subsolution space is used.

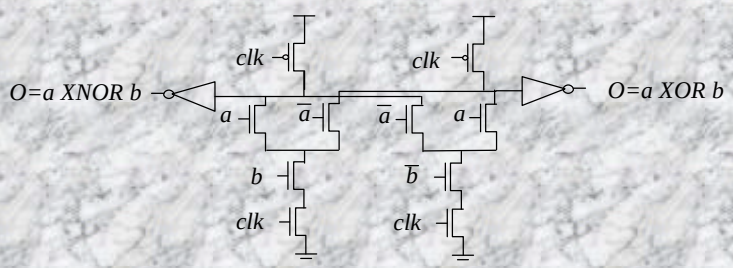


  - Region a: standard domino gate mapping
  - Region b: wide AND domino gate mapping
  - Region c: wide OR domino gate mapping

21

Dual-monotonic gate

- A common dual-monotonic XOR gate.

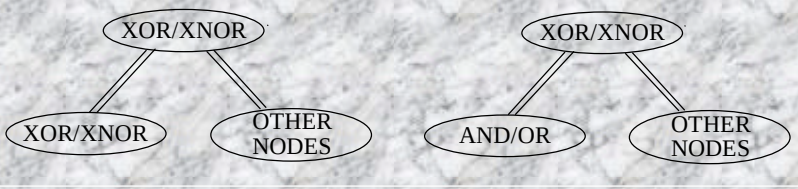


- The presence of an XOR/XNOR function decomposes the input network into small mapping trees, which causes a larger area and delay cost.

22

Dual-monotonic gate mapping

- Recognize the XOR/XNOR logic of the network by pattern matching.
- Perform the technology mapping on the AND/OR/XOR/ XNOR subject network, mapping AND/OR nodes to the standard domino gate and XOR/XNOR nodes to dual-monotonic gate.
- Permitted mapping scheme.



23

Implementation and results(1)

- Execution time: < 10 seconds
- Comparison with another domino mapper

Circuits	*Our approach #trans/#level*	*Prasad et al. #trans/#level*	*Reduction %*
c8	289/6	328/7	13.5%
I6	890/2	890/3	0%
C880	1056/9	1499/7	42.0%

- Comparison of various mapping methods

Circuits	*Basic mapping #trans/#level*	*Wide AND/OR gate #trans/#level*	*Dual-mono gate #trans/#level*
C1355	1824/9	1824/9	1360/7
C1908	1978/18	1965/18	1588/14
k2	2884/16	2738/15	2884/16

24

Experimental results

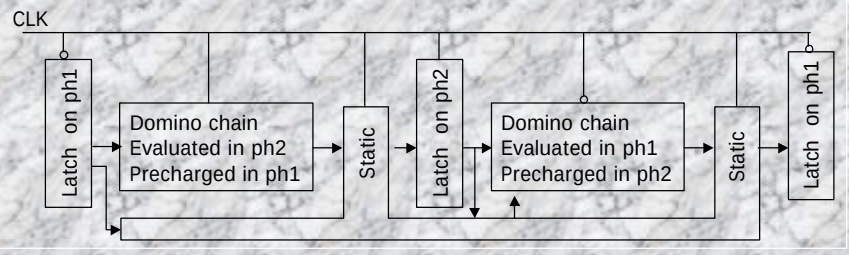
- Domino mapping vs. static mapping

Circuits	*Domino #trans/#levels*	*SIS: 44-3.genlib #trans/#levels*	*Reduction %*	*Dup-ratio %*
i6	761/3	1194/5	36.3%	13%
C1355	1360/7	1378/20	1.3%	77%
C3540	4002/20	3140/34	-27.5%	92%

25

Partitioning: Motivation

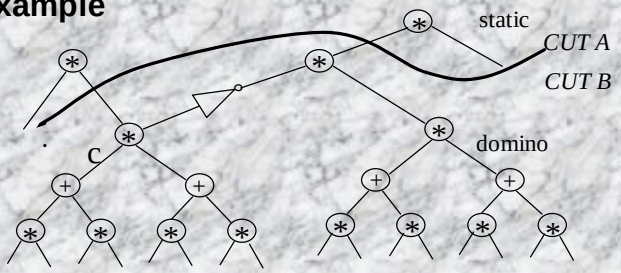
- Use domino gates to speed up parts of the circuit; remainder is implemented in static CMOS
- Domino logic is typically multiphase
- General clocking strategy



26

Another consideration

- Observation: duplication cost can be reduced by proper partitioning
- An example



- In addition to the partitioning cost, implementation cost varies with partitions.

27

Problem definition

- Static-domino partitioning problem
  - *Given*
    - An optimized combinational circuit
    - The delay specification on the output of the network
  - *Implement the nodes with domino+static logic*
    - Minimize the cost while meeting delay specs
    - Satisfy the precedence constraints that no static logic gate is permitted to fan out a domino gate
- Two-way domino partitioning
  - Partition the domino implementation into two phases, with inverters permitted between the phases.

28

The timing-driven static-domino partitioning algorithm

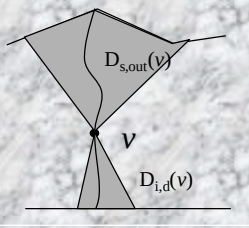
- **Cost: area or power.**
- **Outline of the algorithm**
  - **Perform fast static and domino mapping on the entire logic network.**
  - **Apply a PERT based timing analysis method to find the candidate cut nodes in the network.**
  - **Build the flow network from the candidate cut nodes. The edge capacities are determined from the cost difference of static and domino implementations.**

29

Static-domino mapping algorithm

Determining candidate cut nodes

- **From the static mapping, get**
  - **$D_{i,d}(v)$ ($D_{i,s}(v)$): the delay from the inputs to node *v* using a domino(static) implementation**
  - **Find the maximum delay from output to node *v***
- **If maximum delay from input to output through node *v***
  - **$D_{i,d}(v) + D_{s,out}(v) < T_{spec}$**
  - **Þ *v* is a candidate cut node**

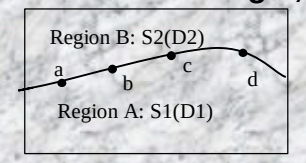


30

Static-domino partitioning algorithm

Finding the minimum cut

- **Notation:**
  - **S1 (D1): the static(domino) implementation cost of region A**
  - **S2 (D2): the static(domino) implementation cost of region B**
- **If regions A and B are implemented in static logic,**
  - ***Cost(s) = S1 + S2;***



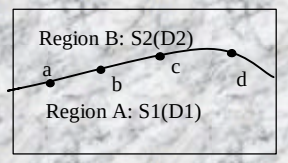
- **If A is domino and B is static:**
  - ***Cost(d-s) = D1 + S2 = D1 - S1 + Cost(s)***

31

Static-domino partitioning algorithm

Finding the minimum cut (Contd.)

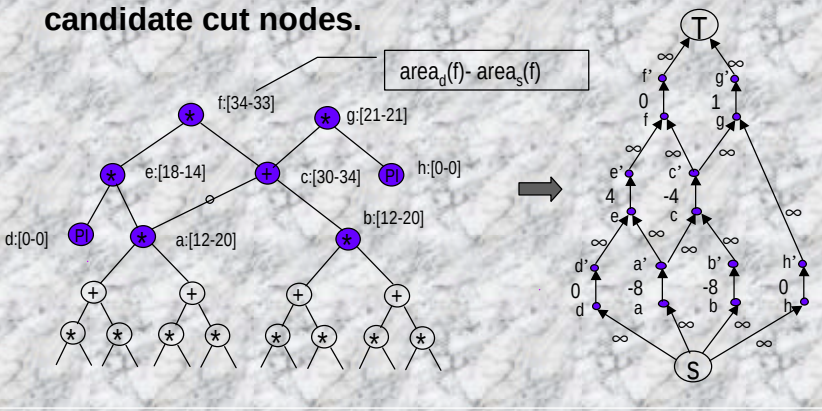
- **Cost(s) is constant.**
- **Therefore, minimizing the partitioning cost is to find the region A whose (D1-S1) is minimized.**
- **(D1-S1) value of a partitioning**
  - ***S [d(i)-s(i)] " i Î cutset between Region A and B***
- **Build the flow network**
  - ***Edge capacities are [d(i)-s(i)] for each node i***
  - ***Standard technique used to maintain precedence constraints***



32

Building the maximum flow graph

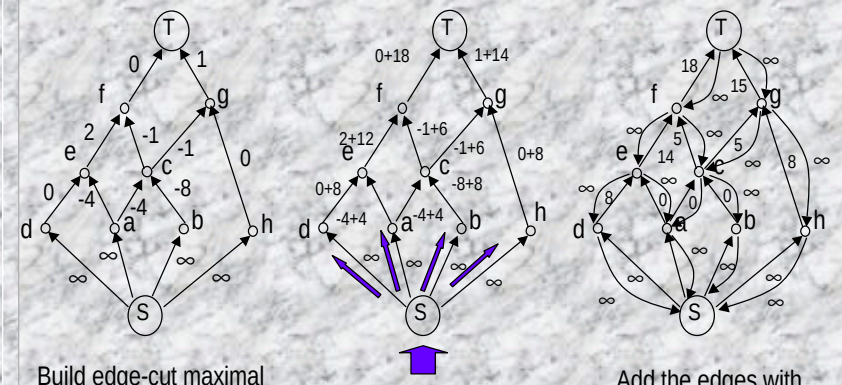
- Build the vertex-cut maximum flow graph from candidate cut nodes.



Maximum flow graph (contd.)

- Constraints to max-flow min-cut algorithm
  - *Maintain the predecessor constraints*
  - *Handle edges with negative capacities.*
- To solve the problem,
  - Heuristically transform the vertex-cut maximum flow network into an edge-cut maximum flow network
  - A positive initial flow is injected into the source node and distributed into the whole network.
  - Edges with capacities of ¥ are introduced into the graph to force the precedessor constraint.

Maximum flow graph: Example



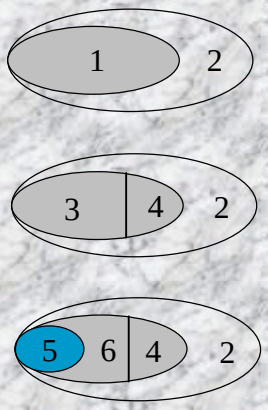
Build edge-cut maximal flow graph

Initial flow=32

Add the edges with capacities of ∞

A partitioning flow for a general two-phase clocking strategy

- **Perform static-domino partitioning on the entire network into domino region(1) and static region(2)**
- **Perform two-way domino partitioning on region 1 to obtain phase I region(3) and phase II region(4)**
- **Perform static-domino partitioning on region 3 into domino region(5) and static region(6)**



Experimental results

- Results of static-domino partitioning (one phase)

Circuits	Domino #trans	Static #trans/delay	No spec #trans	Spec=(*1.25) #trans	Spec=(*1.05) #trans	CPU (s)
C3540	4527	2850/1.43	2748	3312	3987	10.9
des	9945	8134/4.25	7527	7536	7536	60.2
C7552	7919	5464/2.35	5370	5987	6198	30.9

37

Experimental results (Contd.)

- Partitioning flow for two-phase clocking scheme

Circuits	Domino #trans	Static #trans/delay	Spec=(*1.25) #trans/#latches	Spec=(*1.05) #trans/#latches
c2670	1992	1754/1.75	1538/52	1538/52
K2	2884	2896/1.54	2691/157	2795/115
C3540	4527	2850/1.43	3063/60	3235/68
des	9945	8134/4.25	7510/118	7513/119
C7552	7919	5464/2.35	5754/164	5772/164

38

Conclusion

- Synthesis procedure for domino logic discussed
- Technology mapper: fast, good solutions
- Partitioning between static and domino to gain advantages of both
- Placed into a flow including transistor sizing and noise fixes for charge sharing

39