

Power, Delay, and Area Constrained Synthesis for Mixed Domino/Static Logic Optimization

Hui-Yuan Song and R. Iris Bahar
Brown University, Division of Engineering, Providence, RI 02912

Paper Category

Design and Synthesis

Contact Person

Iris Bahar
Brown University
Division of Engineering
Providence, RI 02912
Phone: (401) 863-1430
Fax: (401) 863-1157
E-mail: Iris@lems.brown.edu

Abstract

Domino logic has proved to be a powerful alternative to conventional CMOS in high-performance IC design. Domino logic has many advantages, including fewer transistors, faster switching speeds, and no short-circuit or “glitching” power consumption. This paper addresses the synthesis for mixed domino/static logic to avoid large logic duplication due to the monotonic property of domino logic. We use an accurate timing and power estimation, which accounts for glitching and short-circuit power consumption. Our method gives the designer more flexibility to optimize the circuits with a mix of domino and static logic, and make the tradeoff among several design constraints, including area, delay, and power consumption. Experimental results show the advantages of the mixed domino/static logic. On average for circuits mapped for area optimization, power is reduced by 26% and delay is reduced by 23% without paying a large area overhead. For circuits original mapped for speed, area and delay, on average, remain the same, but power dissipation improves by 35%.

1 Introduction

With the increasingly tighter constraints found in today's VLSI design, domino logic has proved to be a powerful design style. Domino logic has several advantages over conventional CMOS structures, including fewer transistors, faster switching speeds, no short-circuit power consumption and reducing glitching. On the other hand, it has an increased susceptibility to charge redistribution, increased sensitivity to noise and coupling problems, and is limited to non-inverting logic implementations. To invert, the signal must be implemented as the complement of its inputs. Therefore the logic duplication becomes a significant penalty for the use of domino logic. The work proposed in [7] tries to minimize this duplication by optimally assigning primary outputs to true or complemented values. Alternatively, we can implement the circuits with a dominant domino logic followed by a smaller conventional CMOS logic [3] [12]. The motivation is to avoid the logic duplications which are close to the primary output of the circuits.

Since circuits will be synthesized for mixed circuit structures, it is desirable to extract a particular segment or cluster within a circuit and optimize it separately to target a particular logic structure. This task is particularly challenging since choices made on where to partition a circuit will have a large impact on the final optimized and mapped circuit. Using accurate delay and power information, we propose different ways to partition the circuits, optimizing for area, delay, and/or power. This paper is organized as follows. Section 2 presents the background in domino logic operation. Section 3 discusses how to create the parameterized library and characterize the domino gates. Section 4 gives an accurate power model which accounts for both the dynamic transition and short-circuit power dissipation. Section 5 addresses the issue of reducing logic duplication from domino circuits. In Section 6 we propose two ways to partition a network into domino and static parts. Section 7 is dedicated to experimental results, and finally, Section 8 gives conclusions and directions for future work.

2 Domino Logic

Domino logic allows us to significantly reduce the number of transistors used to implement any logic function. The circuit operation is based on first precharging the output capacitance and subsequently, evaluating the output level according to the applied inputs. Both of these operations are scheduled by a single clock signal, which drives one NMOS and one PMOS transistor in each dynamic stage. A domino gate and its equivalent static CMOS gate are shown in Figure 1. The integers near the transistors represent their sizes in units of λ . We can see that the domino gate implementing the function $z = (a(b + c) + de)'$ takes 56λ (not including the inverter), but the equivalent static gate

takes 152λ . In general, for complex functions, static gates are larger than the corresponding domino implementation.

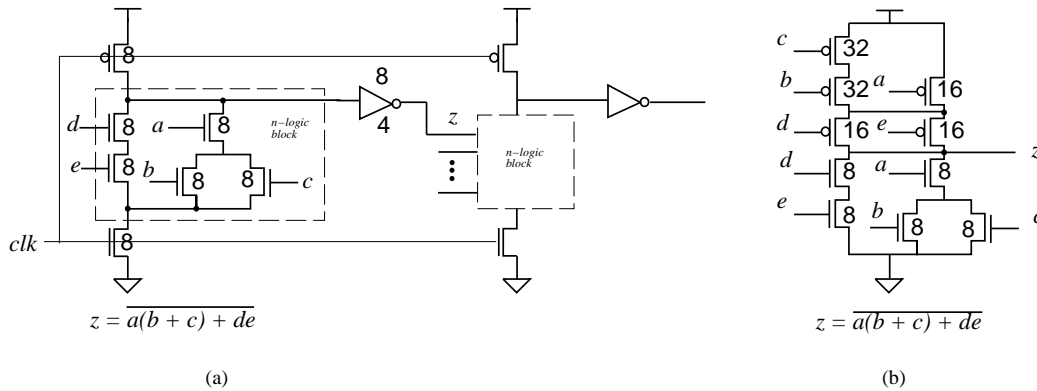


Figure 1: Examples of (a) Domino CMOS Logic, (b) Equivalent Static CMOS Logic.

Due to the monotonic property of domino logic, a static inverting buffer must be incorporated into each dynamic gate. In our implementation, this buffer can be any inverting function (not only an inverter). For example, the output of two NMOS blocks may feed into a single 2-input NAND gate. During precharge ($clk = 0$), the output node of the dynamic gate is precharged high and the output of the buffer is low. All output nodes of subsequent logic stages fed from this buffer will also be precharged high during this phase. When the gate is evaluated, the output will conditionally discharge, allowing the output of the buffer to conditionally go high. Thus each gate in sequence can make at most one transition (1 to 0). Hence, the buffer can only make a transition from 0 to 1. One evident limitation of this structure is only non-inverting structures are possible. To invert a function, we must find a unate representation of the original network and introduce some logic duplication into the circuit.

The results from SPICE simulations show that for NAND2 and NOR2, domino gates are not faster than the equivalent static gates. The reason is that we add two extra transistors in a domino gate even for the gates with simple functions, so there is no advantage in implementing these gates with domino logic. In our experiments, after mapping the circuits with domino gates, we optionally recombine domino NAND2 or NOR2 gates with their inverting buffers and substitute them with a static equivalent (following DeMorgan's law). Therefore we can remove two levels of logic as shown in Figure 2. Note that inputs a and b may be coming from other domino gates (before buffering). Care must be taken during this step since a static gate substitution may add significant load to inputs a and b .

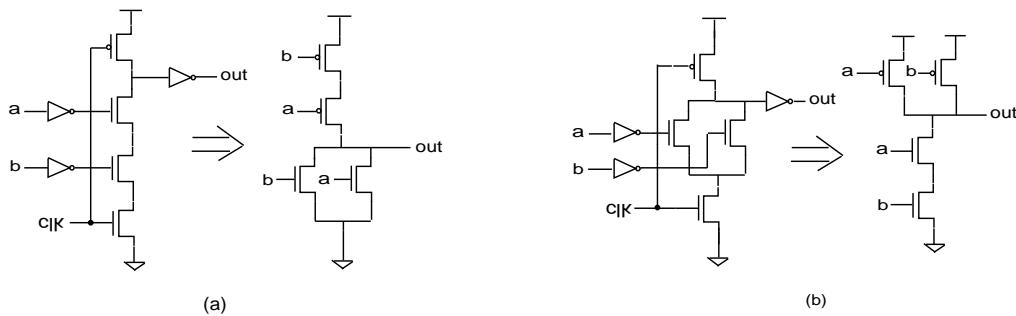


Figure 2: Using DeMorgan to eliminate domino NAND2 (a), and NOR2 gates (b).

3 Combined Standard Cell and Parameterized Libraries

One advantage of implementing circuits using conventional CMOS structures is their ease of design making them particularly well suited for use in a standard cell library. This conventional CMOS library can be defined and characterized early on in the design process and used to implement a wide variety of circuits. The number of complex functions that may be implemented in conventional static CMOS is somewhat limited due to the large number of transistors that would be required to implement these complex functions. On the other hand, this limitation keeps the number of gates available in a given library down to a manageable size, which can be an important consideration since too many choices may severely hamper the speed (and effectiveness) of the technology mapper.

For domino CMOS logic, due to the reduced number of transistors per gate and the single transistor load per fanin, the load capacitance of the gate is substantially lower than for conventional static CMOS. This enables the designer to use rather large NMOS-logic blocks, thus allowing fast-switching complex functions to be implemented by a single gate. Although methods for domino logic standard cell design have been presented [6], it would be difficult to adequately represent the functional flexibility offered by domino logic within a standard cell library without requiring an extremely large number of cells in the library. Indeed, in the case of domino logic, instead of mapping functions to a pre-defined standard cell library, it may be better to generate cells *on-the-fly* using a parameterized library, as suggested in [5]. In addition, the library should include a rich set of standard inverting CMOS gates, as was shown to be beneficial in [9]. To exploit the benefits of each library, we use a standard-cell design for conventional CMOS, and a parameterized library for the domino gates.

We performed timing analysis on each cell within the library using SPICE. From these results, we found that the delay of a domino gate depends greatly on the maximum number of transistors in series and in parallel (and much less on the number of inputs). Therefore, we decided to use these two parameters to characterize the domino gates. In our library, the domino gates are labeled `DOMINO x _ y` , where x denotes the maximum number of transistors in series, and y denotes that in parallel. Note that domino gates with different functions may share the same timing characteristics. For instance, the domino implementation of $f_1 = (ab + c + d)'$, $f_2 = (ab + cd + e)'$ and $f_3 = (ab + cd + ef)'$, are all mapped using the `DOMINO2_3` parameterized gate. Although the delay of the gates are considered the same, their area is calculated according to their functions.

We simulated several domino and static CMOS gates using SPICE to get the corresponding relation between them. Based on a simplified version of `lib2.genlib` (i.e., all pin-to-pin delays are the same), we created a corresponding domino library. This way the static and domino libraries are consistent with each other, since we use the same technology parameters to characterize them. Some results are shown in Tables 1 and 2, where delays are given in *nsec*, assuming an output load of $100pF$. In our experiments, the number of transistors in series is limited to 4, and the number in parallel limited to 6. We found that, for domino gates with the same maximum number of transistors in series, delay increases near linearly as the number of transistors in parallel increases. This is not the case for static gates, as shown in Table 2.

Table 1: Timing characteristics of domino gates.

	1 in	2 in	3 in	4 in	5 in	6 in
1 in series	1.005	1.050	1.100	1.145	1.190	1.229
2 in series	0.642	0.679	0.712	0.744	0.776	0.805
3 in series	0.679	0.716	0.752	0.786	0.822	0.856
4 in series	0.909	0.958	1.008	1.058	1.106	1.153

Table 2: Comparison between domino gates and static gates.

	<i>inverter</i>	<i>NOR2</i>	<i>NOR3</i>	<i>NOR4</i>
static	0.780	1.066	1.645	2.260
domino	1.005	1.050	1.100	1.145

4 Power Model

Several factors contribute to the power dissipation of a circuit, including dynamic, short-circuit, and leakage current. Dynamic power dissipation comes from the source-drain current, which flows when the output of a gate switches values. This component is usually considered the dominant factor in power dissipation and can be estimated as

$$P_{dyn} = V_{dd}^2 * \sum_{i=1}^N (P_i * C_i) * f / 2$$

where P_i is the expected number of transitions of node i in one clock cycle, C_i is the capacitive load of node i , f is the clock frequency, and N is the total number of gates.

The second current contributing to the power dissipation is due to the direct-path short-circuit current. During transition of a node, both NMOS and PMOS transistors are on for a short period of time. This results in a short circuit current pulse from V_{dd} to V_{ss} [10]. Our power estimation due to this current is taken from [11] and is given by

$$P_{sc} = \beta * (V_{dd} - 2V_t)^3 * t_{rf} * f / 12$$

where β is the transistor gain factor, V_t is the threshold voltage, and t_{rf} is the rise or fall time of the input waveform. We assume that $V_{tn} = -V_{tp} (= V_t)$, $\beta_n = \beta_p (= \beta)$, $t_r = t_f (= t_{rf})$. The above formula is for an inverter without load, assuming the rise and fall times are equal. As the load capacitance is increased, the fraction of total power dissipation due to short-circuit effects is reduced and the total power will be dominated by dynamic transition power dissipation. We use the equation above to estimate the short-circuit power dissipation on each gate of the circuit. For complex gates, we calculate the equivalent β according to their maximum number of transistors in series.

The final term is the leakage current which is primarily determined by fabrication technology considerations; we assume it is negligible.

5 Logic Duplication

As mentioned earlier, domino gates require fewer transistors to implement than their equivalent conventional CMOS gates, however, since only non-inverting logic is possible, complements of internal signals must be realized through separate cones of logic using complements of primary inputs. To minimize the logic duplication, the method of output phase assignments may be used to reduce the number of inverters in the circuits [7]. However, when some inverters or inverting gates are trapped in reconvergent fanout nets, logic duplication is still necessary.

Reconvergent fanout is the presence of two or more distinct paths with a common input gate, leading to a common output gate, and with no other gates in common. The gate where the paths reconnect is called the *reconvergence gate*, or common output such as gate G5 in Figure 3(a), in which two distinct paths (highlighted in bold) from gate G1 reconverge at gate G5. All XOR and XNOR gates contain reconvergent fanout by definition, since they require both the inputs and their complements be available.

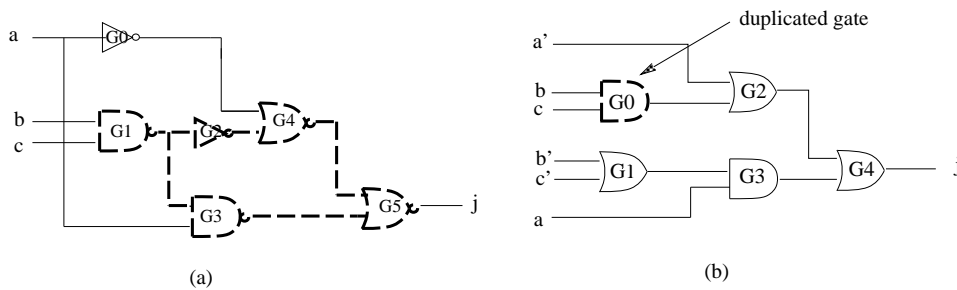


Figure 3: Example of reconvergent fanout nets. (a) Circuit mapped for static gates, (b) Circuit mapped for domino gates with logic duplications.

We find the reconvergent fanout nets with a method similar to [7], and account for all the gates with inverting functions on the different paths. Let n_1 denote the number of inverting gates on one path of the reconvergent fanout net, and n_2 denote the number of inverting gates on another path. We have two cases:

$|n_1 - n_2|$ is even Without any logic duplication we can implement the function with domino gates such that every fanin of the domino gates is a static inverting gate and every fanin of the static gates is a domino gate.

$|n_1 - n_2|$ is odd This implies that one path has even inversions and the other has odd inversions. Because an even number of inversions can be eliminated, we will always have one inversion remaining in one path of the reconvergent fanout net [7]. The inverted function have to be implemented with some logic duplication.

For the circuit in Figure 3, we have two reconvergent paths with common input G1, and common output G5 ($G1 \rightarrow G2 \rightarrow G4 \rightarrow G5$ and $G1 \rightarrow G3 \rightarrow G5$) Since $|n_1 - n_2| = 1$, we must duplicate logic to implement true and complement of the common input gate G1, as shown in Figure 3(b). Note that there are some other reconvergent fanout nets in this circuit, such as the one coming from primary input a and reconverging at gate G5. Because we assume the complements of primary inputs are available, we don't consider these reconvergent subnetworks.

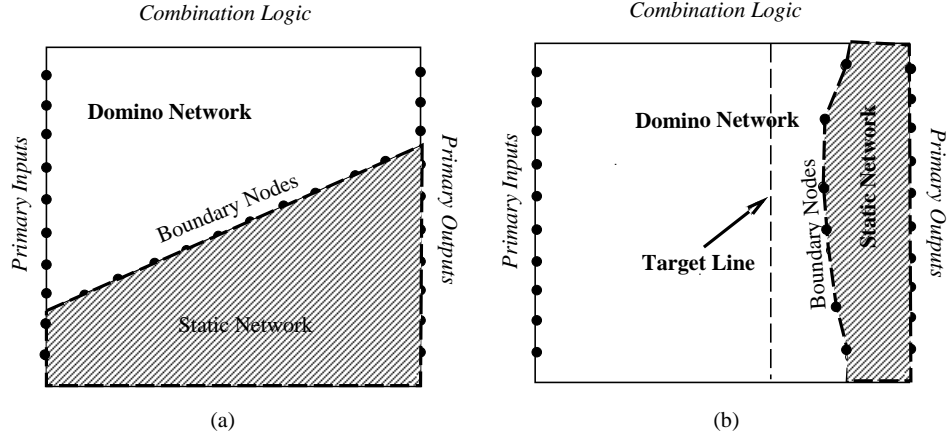


Figure 4: Partitioning the network from primary outputs (a) or internal nodes (b).

6 Partition of the Networks

The motivation to partition the networks is that we want to identify regions of a chip that may be best suited for one circuit design style or another depending on the specific design constraints of the region and select a mix of appropriate implementations to get the optimal design. In other words, if the implementation of some gate with domino logic requires too much logic duplication, we may want to retain its implementation in conventional CMOS logic. Due to the monotonic property of domino logic, we can only implement the circuits with a domino subnetwork followed by a static one. Retaining some static CMOS logic in the circuit may be helpful to avoid significant area penalty due to duplication.

We choose two ways to partition the circuits. The first one is cutting from the primary outputs. First we compute the critical delay of the circuit. Then we obtain a critical set of primary outputs such that their delay is within a user-defined threshold of the critical delay. Next, starting from these primary outputs, we use a depth first search (DFS) to obtain all the gates within the transitive fanin cones of the critical set. These gates make up a subnetwork that is to be implemented with domino logic. The remaining subnetwork will be implemented with conventional CMOS logic. The main idea is shown in Figure 4(a). By adjusting the threshold, we can control the number of primary outputs included in the critical set.

Another way to partition the network is by cutting from internal nodes. We use another user-defined threshold for the logic level of the gates. According to this threshold, a target boundary is defined as shown in Figure 4(b) such that all gates with logic level less than the threshold are on the left of the target boundary. For each of the gates between the target boundary and the primary outputs, we use breadth first search (BFS) to search for the reconvergent fanout nets with trapped inverting gates. The key idea is that we don't want to duplicate any logic beyond the target boundary. If the common inputs are before the target boundary, their duplications are allowed, otherwise, they are prohibited and logic must be implemented in static CMOS. First we put all the primary outputs into the boundary set, then check the searching results for other gates. If a subnetwork described above is found and its common input is beyond the target boundary, we put this common input into the boundary set. Otherwise, either no subnetworks is found or the common input is before the target boundary so duplication will be allowed.

Once the boundary set is created, there may be some fanin and fanout relationship between the gates contained in the set. We picked out its left frontier as the boundary nodes to cut the network. In other words, we prune the boundary set such that any fanin of the boundary nodes should not be included in the boundary set. All gates in the transitive fanin of the boundary nodes are implemented in domino logic. All others are static CMOS.

7 Experimental Results

Table 3: Initial results mapped for area and speed.

Circuit	Mapping for Area					Mapping for Speed				
	Gates	Area	Delay	Power	Level	Gates	Area	Delay	Power	Level
rd53	25	48256	9.87	298	6	52	77024	6.66	387	8
rd73	83	161008	14.33	1028	7	175	251024	9.59	1182	10
rd84	93	178176	17.13	1182	9	198	278400	10.53	1416	11
duke2	319	664448	31.02	4482	11	666	993888	11.71	4623	13
clip	112	214832	14.15	1353	8	192	291392	9.28	1488	10
5xp1	74	135024	10.89	826	6	130	186992	7.88	904	8
misex1	38	75168	9.76	462	5	77	115072	5.77	540	6
sao2	84	167040	16.62	937	9	154	234784	9.92	1088	10
squar5	37	73312	10.04	459	5	70	108112	5.99	530	6
c499	287	458432	29.16	4840	19	486	665376	20.84	7142	21
c880	225	369344	30.95	2310	19	344	481168	19.12	2284	21
c1355	510	713632	30.73	5239	26	618	821280	24.62	5735	27
c1908	349	610160	42.68	7920	25	660	907120	29.33	7920*	30

*Due to memory problems, we use the same power cost as that of the circuit mapped for area.

Table 4: Mapping the whole circuit with domino logic.

Circuit	Mapping for Area					Mapping for Speed				
	G_d/G_s	ΔA	ΔD	ΔP	ΔL	G_d/G_s	ΔA	ΔD	ΔP	ΔL
rd53	18/0	1.29	0.82	0.71	1.83	20/0	0.87	0.86	0.52	0.88
rd73	59/0	1.24	0.67	0.71	1.57	49/0	0.76	0.99	0.50	1.10
rd84	64/0	1.28	0.65	0.71	1.44	63/0	0.85	1.08	0.58	1.00
duke2	257/0	1.41	0.58	0.52	1.55	281/0	1.05	1.16	0.68	0.85
clip	87/0	1.24	0.71	0.67	1.38	84/0	0.96	0.98	0.70	1.00
5xp1	54/0	1.25	0.70	0.65	1.50	45/0	0.92	0.97	0.67	0.88
misex1	24/0	1.27	0.65	0.66	1.40	23/0	0.89	1.02	0.58	0.83
sao2	68/0	1.33	0.71	0.70	1.67	68/0	1.03	1.07	0.63	1.00
squar5	29/0	1.29	0.65	0.68	1.60	26/0	0.93	0.92	0.59	0.83
c499	288/0	1.86	0.67	0.80	1.32	210/0	1.10	0.73	0.48	0.81
c880	206/0	1.67	0.61	1.01	1.32	226/0	1.35	0.97	1.05	0.95
c1355	461/0	1.74	0.92	0.90	1.23	436/0	1.47	0.97	0.83	1.00
c1908	433/0	2.07	0.63	0.70	1.24	453/0	1.44	0.83	0.66	0.90
Average		1.46	0.69	0.72	1.47		1.05	0.97	0.65	0.93

The circuits are initially mapped with SIS [8] for either area (using **map**) or for delay (using **map -n 1 -AFG**). The statistic for the mapped circuits are shown in Table 3. We report the number of gates, the area (in μm^2), the worst

case delay (in *nsec*), the power consumption (in μW), and the maximum gate levels of logic. The power consumption is estimated by using a modified version of the power simulation tool in SIS which accounts for short circuit power.

First, we remapped the whole circuit with domino logic. The results are shown in Table 4. Next, we used two ways to partition the network as described in Section 6. The first way cut the network from the primary outputs (as shown in Figure 4(a).) We obtain the results using two thresholds, 0.01 and 0.10. Threshold 0.01 means that all primary outputs within 1% of the critical delay will be mapped with domino logic. The best results are shown in Table 5. The relative changes in area, delay, power and levels are shown in the columns labeled ΔA , ΔD , ΔP , and ΔL (e.g., a 0.80 in the ΔA column indicates a 20% reduction in area compared to that given in Table 3). In the column G_d/G_s , the number of gates in the domino subnetwork G_d and in the static subnetwork G_s are shown. We use the command **demorgan** as described in Section 2 to recombine small domino gates with inverters back to a single inverting static gate. Therefore we can remove two levels of logic for the inverters on the input and output pins of the domino gates.

Table 5: Cutting the Network from the Primary Outputs.

Circuit	Mapping for Area						Mapping for Speed					
	Th	G_d/G_s	ΔA	ΔD	ΔP	ΔL	Th	G_d/G_s	ΔA	ΔD	ΔP	ΔL
rd53	0.01	9/ 10	1.17	0.88	0.88	1.83	0.01	20/ 0	0.87	0.86	0.52	0.88
rd73	0.01	31/ 29	1.14	0.86	0.78	1.57	0.01	40/ 59	0.96	0.99	0.74	1.10
rd84	0.01	48/ 12	1.15	0.66	0.71	1.44	0.10	60/ 6	0.85	1.07	0.59	1.00
c499	0.01	217/ 115	1.73	0.77	0.60	1.42	0.01	210/ 0	1.10	0.73	0.48	0.81
c880	0.01	112/ 135	1.52	0.69	1.10	1.47	0.10	139/ 149	1.27	1.04	1.11	1.00
5xp1	0.01	30/ 26	1.14	0.81	0.77	1.50	0.10	40/ 12	0.93	0.99	0.73	0.88
duke2	0.01	17/ 281	1.04	0.78	0.89	1.36	0.10	217/ 145	1.05	1.16	0.74	0.85
clip	0.01	23/ 80	1.09	0.88	0.89	1.38	0.01	65/ 33	0.94	1.09	0.73	0.90
misex1	0.01	5/ 28	1.11	0.84	0.93	1.40	0.01	17/ 20	0.93	1.12	0.71	1.00
sao2	0.01	20/ 53	1.14	0.85	0.93	1.67	0.01	38/ 57	1.05	1.05	0.84	1.00
c1355	0.01	587/ 164	2.14	1.12	0.84	1.58	0.01	436/ 0	1.47	0.97	0.83	1.00
c1908	0.01	230/ 173	1.69	0.71	0.65	1.32	0.01	319/ 185	1.28	0.83	0.61	0.97
squar5	0.01	3/ 28	1.04	0.85	0.94	1.00	0.10	16/ 25	0.98	1.00	0.78	1.00
Average			1.32	0.82	0.84	1.46			1.05	0.99	0.72	0.95

In counting the logic levels of the mixed domino/static circuits, we consider the static inverting gate following the domino gate as a separate logic level, so generally we obtain mixed domino/static circuits with increased logic levels. From the results, we can see that the power consumption is reduced greatly by using domino logic. The main reason is that there is no glitching or short circuit power consumption for domino gates. Due to the great reduction in power cost, the small percentage increase in area is a reasonable tradeoff. Compared with the results in Table 4, fewer domino gates were used, and since there was less logic duplication, the circuits required less area. However, delay and power improvements were not as good. One reason for this is that some non-critical paths may now become critical if they are not implement with domino logic.

The second way we partitioned the network is by cutting it along internal nodes. The optimal results obtained for different user-defined thresholds, (0.90, 0.80, 0.70 and 0.60) are shown in Table 6. A threshold of 0.90 means that all the gates within 10% of the maximum logic level are targeted to search for reconvergent subnets. For all the benchmarks, we search backwards for up to 10 levels. Compared with the results in Table 4, we obtained a similar tradeoff as with Table 5 (i.e., less area, larger delay and power consumption.)

Table 6: Results of cutting from the internal nodes.

Circuit	Mapping for Area						Mapping for Speed					
	Th	G_a/G_s	ΔA	ΔD	ΔP	ΔL	Th	G_a/G_s	ΔA	ΔD	ΔP	ΔL
rd53	0.60	15/ 5	1.14	0.82	0.55	1.83	0.90	20/ 3	0.87	0.86	0.48	0.88
rd73	0.90	59/ 3	1.24	0.67	0.69	1.57	0.90	49/ 3	0.76	0.99	0.49	1.10
rd84	0.60	50/ 30	1.24	0.73	0.72	1.44	0.90	63/ 4	0.85	1.08	0.56	1.00
duke2	0.70	258/ 36	1.40	0.58	0.52	1.64	0.70*	272/ 58	1.12	1.01	0.64	1.00
clip	0.60	62/ 35	1.17	0.79	0.72	1.38	0.90	84/ 5	0.96	0.98	0.68	1.00
5xp1	0.90	3/ 71	1.04	0.94	1.01	1.00	0.70	38/ 24	0.94	1.05	0.69	0.88
misex1	0.90	24/ 7	1.27	0.65	0.61	1.40	0.90	23/ 7	0.89	1.02	0.52	0.83
sao2	0.90	68/ 3	1.33	0.71	0.69	1.67	0.90	68/ 4	1.03	1.07	0.62	1.00
squar5	0.90	29/ 8	1.29	0.65	0.61	1.60	0.90	26/ 8	0.93	0.92	0.53	0.83
c499	0.60	189/ 125	1.61	0.77	0.57	1.32	0.60	48/ 422	1.11	0.86	0.42	0.90
c880	0.60	151/ 94	1.52	0.73	1.06	1.32	0.60	138/ 165	1.25	1.01	0.96	1.05
c1355	0.60	317/ 509	1.91	1.08	0.78	1.42	0.60	432/ 266	1.69	1.12	0.61	1.26
c1908	0.60	254/ 190	1.63	0.77	0.48	1.48	0.60	199/ 393	1.16	0.88	0.45	1.00
Average			1.37	0.76	0.69	1.47			1.04	0.99	0.59	0.98

*We build some inverter trees on its primary inputs because they are heavily loaded.

Finally, we combined these two partitioning algorithms together. First we identified the fanin cone of the critical primary outputs using the first method, then we searched for reconvergent fanout nets within this critical cone. Thus we could avoid some logic duplications for the non-critical part of the circuits. The results are shown in Table 7. From these results we can see that the area is greatly reduced, particularly for circuits mapped for area, since logic duplication requirements are reduced. However, in general, the circuits also have larger delay and power consumption. These methods are not so uniformly effective for speed mapped circuits. We found that even small amounts of logic duplication can increase critical paths (due to more heavily loaded gates). Furthermore, the area saved by not duplicating logic, may be in regions of the circuit that do not effect the critical path. In practice, we can make the tradeoff among the area, delay, and power consumption of the circuits by the different partitioning methods.

8 Conclusions

In this paper, we addressed the problem of synthesis for mixed domino/static logic within the constraints of area, delay, and power. To obtain the best tradeoff of the three, we tried to selectively implement only a portion of the circuit in domino logic, and thus avoid the area penalty due to large logic duplication. Three methods for partitioning the circuits were presented. Experimental results for the different partition techniques were compared. Results indicate that the power dissipation can be reduced significantly when even a small portion of the circuit is implemented in domino logic (on average 35% for circuits mapped for speed). Delay improvements are most pronounced for circuits originally mapped for area minimization (average 26% improvement), however, even circuits originally mapped for delay optimization have shown a speed improvement of up to 27%. An all-domino implementation may require a significant area overhead, especially for area mapped circuits; by judiciously partitioning the circuit, the area overhead can be reduced from 46% to 29% while still maintaining delay and power improvements. For future work, we are modifying our partitioning algorithms to better target regions of the circuits with high power dissipation .

Table 7: Results from Combining the Two Methods.

Circuit	Mapping for Area						Mapping for Speed					
	Th	G_d/G_s	ΔA	ΔD	ΔP	ΔL	Th	G_d/G_s	ΔA	ΔD	ΔP	ΔL
rd53	0.60	13/9	1.21	0.86	0.71	1.83	0.90	20/3	0.87	0.86	0.48	0.88
rd73	0.90	31/30	1.14	0.86	0.78	1.57	0.90	49/3	0.76	0.99	0.49	1.10
rd84	0.60	34/40	1.12	0.74	0.72	1.44	0.90	60/8	0.84	1.07	0.58	1.00
duke2	0.70	158/131	1.24	0.66	0.66	1.64	0.70*	203/184	1.11	0.99	0.73	1.00
5xp1	0.90	38/19	1.16	0.77	0.68	1.50	0.90	40/20	0.94	0.97	0.68	0.88
clip	0.60	52/47	1.16	0.83	0.75	1.38	0.90	84/5	0.96	0.98	0.68	1.00
misex1	0.90	14/18	1.19	0.78	0.75	1.40	0.90	23/7	0.89	1.02	0.52	0.83
sao2	0.90	49/20	1.21	0.73	0.73	1.67	0.90	68/4	1.03	1.07	0.62	1.00
squar5	0.90	23/11	1.23	0.67	0.71	1.60	0.70	16/26	0.96	1.00	0.74	1.00
c499	0.60	189/125	1.61	0.77	0.56	1.32	0.80	202/216	1.25	0.77	0.40	0.86
c880	0.70	43/164	1.08	0.63	0.70	1.05	0.60	21/270	0.97	0.97	1.00	1.00
c1355	0.60	317/509	1.91	1.08	0.75	1.42	0.60	432/266	1.69	1.12	0.64	1.26
c1908	0.60	190/219	1.50	0.83	0.48	1.48	0.60	199/394	1.16	0.88	0.46	1.00
Average			1.29	0.79	0.69	1.48			1.03	0.98	0.62	0.99

*We build some inverter trees on its primary inputs because they are heavily loaded.

References

- [1] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. DeMicheli, "Decision Diagrams and Pass Transistor Logic Synthesis," *International Workshop on Logic Synthesis*, Lake Tahoe, CA, May 1997.
- [2] P. Buch, A. Narayan, A. R. Newton, and A. Sangiovanni-Vincentelli, "On Synthesizing Pass Transistor Networks," *International Workshop on Logic Synthesis*, Lake Tahoe, CA, May 1997.
- [3] K. Kim, C. L. Liu and S. M. Kang, "Implication Graph based Domino Logic Synthesis." *International Conference on Computer Aided Design*, pp. 111-114, 1999.
- [4] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic Decomposition During Technology Mapping," *International Conference on Computer-Aided Design*, pp. 264–271, November 1995.
- [5] M. R. Prasad, D. Kirkpatrick, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Domino Logic Synthesis and Technology Mapping," *International Workshop on Logic Synthesis*, Lake Tahoe, CA, May 1997.
- [6] J. A. Pretorius, A. S. Shubat, and C. A. T. Salama, "Optimization of Domino CMOS Logic and its Applications to Standard Cells," *Custom Integrated Circuits Conference*, pp. 150–153, 1984.
- [7] R. Puri, A. Bjorksten, and T. E. Rosser, "Logic Optimization by Output Phase Assignment in Dynamic Logic Synthesis," *International Conference on Computer Aided Design*, pp. 2–8, 1996.
- [8] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoj, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization," *ICCD-92: IEEE International Conference on Computer Design*, pp. 328-333, Cambridge, MA, October 1992.
- [9] T. Thorp, G. Yee, and C. Sechen, "Domino Logic Synthesis Using Complex Static Gates," *International Conference on Computer Aided Design*, pp. 242–247, November 1998.
- [10] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design- A Systems Perspective*, second edition, Reading, MA: Addison-Wesley, 1993.
- [11] H. J. M. Veendrick, "Short-circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, pp. 468-473, August 1984.
- [12] M. Zhao, S. S. Sapatnekar, "Timing-Driven Partitioning For Two-Phase Domino and Mixed Static/Domino Implementations," *International Conference on Computer Aided Design*, pp. 242–247, November 1999.