Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool

Luciano Lavagno, Patrick C. McGeer,
Alexander Saldanha, Alberto L. Sangiovanni-Vincentelli
Cadence Berkeley Laboratories
1919 Addison Street, Suite 301
Berkeley-CA 94704

Abstract A method of synthesizing low-power combinational logic circuits from Shannon Graphs is proposed such that an *n* input, *m* output circuit realization using 2-input gates with unbounded fanout has *O(nm)* transitions per input vector. Under a bounded fanout model, the transition activity is increased at most by a factor of *n*. Moreover, the power consumption is independent of circuit delays.

1 Introduction

This paper proposes a new method of synthesis for combinational logic circuits that can yield up to 5X reduction in the power dissipation over circuits optimized for some other criteria, *e.g.* area or performance. The power reduction is achieved by reducing the transition activity in a combinational logic circuit while paying close attention to capacitance effects due to the fanout loads of gates.

A change in the value at the primary inputs of the circuit may cause wasted power due to two reasons: (1) There may be multiple paths that propagate a transition to an output; (2) There may be glitches on gates caused by transitions to **0** and **1** arriving at different times at the gate inputs. The technique described in this paper attempts to minimize such wasted computation with the goal to *minimize the power consumption due to transition activity under a bounded fanout model. Additionally, unlike all previous synthesis methods, the power consumed is independent of the delays of the circuit elements.*

Techniques for reducing the power consumption in electronic circuits may be applied at all levels of design. See [1] for a survey of applicable techniques. Here we will only be concerned with logic synthesis techniques for low-power circuits. An objection to all of the existing approaches is that it is very difficult to estimate either the peak or average power consumption which are complex functions of both the logical and timing properties of circuits. Further none of the published approaches has shown any significant power reduction, *i.e.* greater than 50%.

2 Power Consumption Model

We focus on reducing the total power due to switching activity at each node in a circuit. The dynamic power consumed by a CMOS gate *i* with a load capacitor C_i is given by $P = 0.5 p_i C_i V^2 f$ where p_i is the probability of a transition at *i*, *V* is the supply voltage and *f* is the clock frequency. The approach in this paper targets the reduction of C_i and p_i over all the gates in the circuit.

32nd ACM/IEEE Design Automation Conference ®

From the power equation, accurate estimation of the dynamic power dissipation requires that both C_i and p_i be known for each gate *i*. While p_i can be determined exactly in the proposed design style, an accurate estimate of C_i for each *i* must be provided to the algorithm. There are three sources of capacitance at a gate: (1) the capacitance due to the fanin gates, (2) the capacitance due to the fanout gates, and (3) the interconnection wire capacitance. While any user-provided load capacitance model can be utilized in the algorithm, in this paper the load capacitance at a gate is abstracted by an estimation based on the fanout of the gate. Since gates at this level of design may be of arbitrary size and complexity, it is difficult to easily estimate the load capacitance contributed by each complex gate. Thus, a two-input simple gate realization is utilized for estimating the power consumption in the synthesis algorithm. Let *P* denotes the power consumed by a single transition on a gate with fanout one. Under this model a transition on a gate with fanout of *n* consumes *nP* power. Although the effect of the technology mapping process is not accounted for in the proposed synthesis algorithm, *all the experimental results in this paper are reported after mapping to the gates in a given technology library.*

3 Shannon Circuits

Definition 1 *A Shannon Graph of a function is defined as follows:*

1. *Nodes labelled 1 and 0 are the Shannon Graphs of the constant functions 1 and 0, respectively.*
2. *A Shannon Graph for a nonconstant function is a rooted, labelled, binary, directed acyclic graph with the following properties:*
   - *i. The outdegree of the root of f is two;*
   - *ii. One edge of the root of f is labelled x, and the other labelled $\bar{x}$, where x is any variable such that $f_x \neq f_{\bar{x}}$;*
   - *iii. The edge of the root of f labelled x is the root of a Shannon Graph of f_x;*
   - *iv. The edge of the root of f labelled $\bar{x}$ is the root of a Shannon Graph of $f_{\bar{x}}$.*

If no two nodes in a Shannon Graph of f are roots of Shannon Graphs of identical functions, the Shannon Graph of f is said to be reduced.

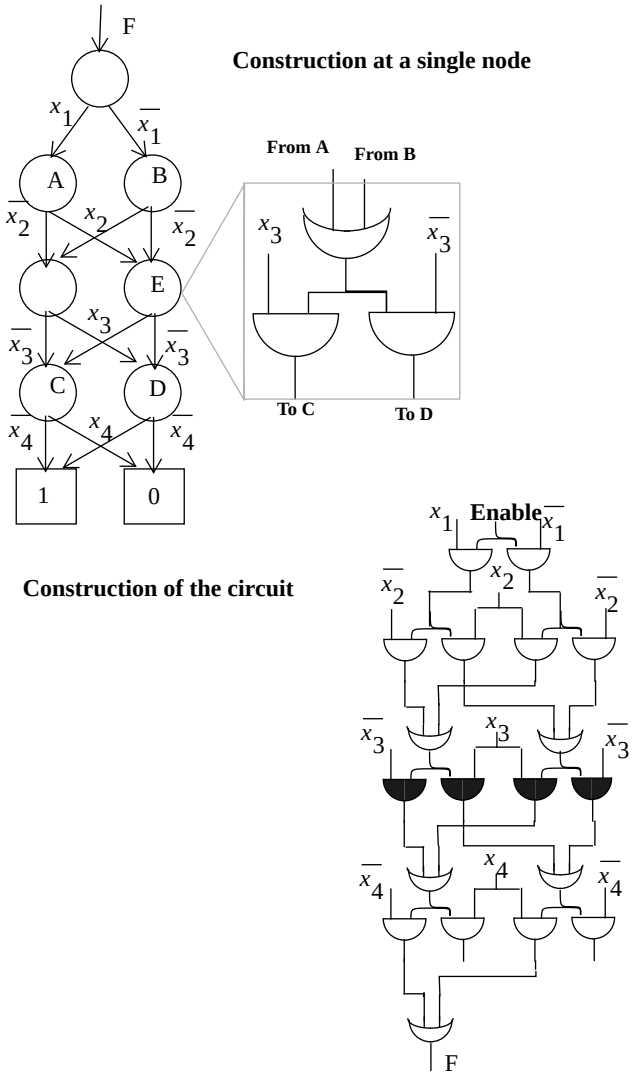
A **Binary Decision Diagram (BDD)** is a Reduced Shannon Graph with the property that no path from the root to the terminals in the Reduced Shannon Graph contains two distinct edges labelled with the same variable.

A Shannon Circuit is derived by starting computation at the root of the Shannon Graph and performing the evaluation towards the terminals. This preserves the desirable one-path-only evaluation property of the Shannon Graph. Signals flows from the top of the Shannon Circuit to the bottom - just as in the Shannon Graph. There is one wire per edge of the Shannon Graph and each node of the Shannon Graph gives rise

to two gates if the in-degree of the node is one, and three if it is greater than one.

In Figure 1, the construction at the node labeled E is shown. The incoming wires from A and B are input to an **OR** gate, and the output of this gate in input to two **AND** gates; the first **AND** gate has x_3 as its second input, the other has $\overline{x_3}$ as its second input. The output of these **AND** gates feed the subnetworks constructed from nodes C and D, respectively.

FIGURE 1. Construction of Timed Shannon Circuit



The output of the circuit is the **OR** of all the edges leading to the **1** terminal in the Shannon Graph. A single wire connected to an enable signal, denoted **Enable**, is connected to the two **AND** gates created at the root node. The two **AND** gates which correspond to the edges which are incident upon the **0** terminal are unused, and are deleted. Note, however, that a dual rail output circuit can be obtained by creating an **OR** gate from the edges leading into the **0** terminal. The dual rail circuit can be used to achieve self-clocking and asynchronous implementations of a given circuit. Alternatively, by simply adding an **OR** gate with inputs connected to the **OR** gates for the **0** and **1** terminals of the Shannon Graph, a completion signal is obtained which can be used for asynchronous operation.

The Shannon Circuit is designed to be operated in a clocked mode, using the **Enable** signal to operate the timing scheme (hence the name Timed Shannon Circuit). The **Enable** signal is also necessary for hazard-free operation. **Enable** is first set to 0 so that all gates in the circuit evaluate to 0. Next the circuit inputs are changed. Since all gates are at 0 and each input is connected to an **AND** gate, whose other input is at 0, there are no transitions within the circuit. After the circuit inputs settle to their new values, **Enable** is set to 1 and precisely those nodes on the single selected path from the root to a terminal node are set to 1. The value is then read from the output of the circuit. Finally **Enable** is set to 0 again, so that all internal gates evaluate to 0. The cycle is repeated for the next input vector.

4 Transition and Power Analysis

The power consumed by a Timed Shannon Circuit with n inputs and m outputs is analyzed. The model assumes two-input **AND** and **OR** gates. The case of unbounded fanout is considered first followed by an analysis for the bounded fanout case.

An **AND** gate in the Timed Shannon Circuit is 1 if and only if the corresponding edge in the Shannon Graph is traversed on evaluation. Since only one edge in the Shannon Graph may be traversed per input variable, it follows that exactly one **AND** gate per input variable per output is set to 1. Further, since each **OR** gate feeds two **AND** gates, and since the **OR** gate rising must force one of the two successor **AND** gates to rise, it follows that at most one **OR** gate rises for each rising **AND** gate.

The **OR** gate denoting a node in the Shannon Graph may have fanin equal to the in-degree of the node, which can be extremely high, particularly near the **1** terminal of the graph. Let M denote the largest fanin of a node in the Shannon Graph. An **OR** gate with M inputs can be decomposed into a balanced binary tree of two input **OR** gates, which yields a tree of depth at most $\lceil \log M \rceil$. It follows that for an n input, m output circuit, at most nm **AND** gates and $nm\lceil \log M \rceil$ **OR** gates rise each time **Enable** is set to 1; these same gates will fall when **Enable** is set to 0. This implies that on each full cycle there are at most $2nm\,(1 + \lceil \log M \rceil)$ gate transitions in the circuit.

This bound can be improved by employing a (non-reduced) Shannon graph where each internal node in the graph has in-degree at most two. This is achieved by duplicating nodes in the Shannon Graph while traversing from the root towards the terminals. In this case there may be up to 2^n edges leading into the **1** terminal of the Shannon Graph for each output. Using a balanced tree decomposition into two input **OR** gates at gate for the **1** terminal yields an upper bound of $4nm$ transitions, which may be tighter than the upper bound derived earlier.

However, the power expended when the primary inputs change must be counted. In the worst case, each input changes, and a transistor is charged or discharged on every **AND** gate in the circuit. Since there is one **AND** gate per edge in the Shannon Graph, the worst-case power is: $P_{\text{total}} = 2nm\,(1 + \lceil \log M \rceil)\,P + 2|E|P$ where $|E|$ is the number of edges in the BDD's for all outputs. Using node duplication with balanced tree decomposition for the **OR** gate at the **1** terminal yields a second upper bound: $P_{\text{total}} = 4nmP + 2|E|P$

4.1 Estimating Mean Transition Activity

The maximum power consumed under any input vector by the Timed Shannon Circuit is directly calculated since the circuit is delay-insensitive. The average power consumed by an input vector, on the other hand, requires knowledge of the internal probabilities of the various nodes of the network. In particular, two sets of probabilities are of interest:

1. The probability that a primary input x_i will *switch*: this probability, which we denote p_i, we find as $p_i = 2p(x_i = 1)(1 - p(x_i = 1))$. Observe that $0 \le p_i \le 0.5$, and the maximum occurs at $p(x_i = 1) = 0.5$; this case is assumed for the remainder of the paper.
2. The probability that an internal gate g is 1, or the *1-controllability* of g is denoted p_g. Given the probabilities that each primary input is 1, p_g is computed *exactly* for each node and edge as follows:
   a. The 1-controllability of the **Enable** input is set to 1.0;
   b. The 1-controllability of the output of an AND gate is equal to the product of the 1-controllability of its inputs.
   c. The 1-controllability of the output of an OR gate is equal to the sum of the 1-controllability of its inputs.

5 Optimizations

5.1 Area Recovery

The circuit derived from a Shannon Graph may have redundant stuck-at-1 faults on the input of AND gates corresponding to edges in the Shannon Graph. This occurs if the function computed by the left child in the Shannon graph is contained by the function computed by the right child of the Shannon graph, or vice versa; *i.e.* the function is *unate* in the variable at the node. Removal of the redundant stuck-at-1 fault does not give rise to any glitches during operation of the Timed Shannon Circuit. At worst, there may be more than one path that propagates a transition; hence some wasted power may be expected. However, in practice it is observed that the power reduction due to decrease in fanout on the primary inputs more than offsets the power increase due to possible multiple transitions. Hence, redundancy removal is invoked on the circuit derived from the BDD. The circuit at this stage is also irredundant for all single stuck faults.

The second operation performed to recover area is the merging of gates with identical logic functions in the Shannon Circuit. Although this step is a conceptually simple optimization, it has proven critical in achieving Timed Shannon Circuits which have area and delay competitive with the original optimized circuits. Note that power is saved since a single transition occurs instead of two or more transitions whenever merging of identical logic is performed.

5.2 Decomposition of High Fanin Gates

The Shannon Circuit, as described above, can have gates with high fanin. These gates must be decomposed into trees of gates with a bounded number of fanin.

A naive decomposition of an m-input gate yields a tree with m gates and $\log m$ transitions. More generally, if the inputs to the **OR** gate have 1-controllability $\{p_1, \ldots, p_m\}$, the total *mean* power consumption is: $2P\sum_{i=1}^{m} p_i$. Consider the problem of minimizing the power consumed by the decomposition. This requires that the expected number of gates in the tree which transition on an input vector change must be minimized. Murgai, *et al.* analyze this problem in [3], and demonstrate that it is equivalent to the problem of finding a minimum weighted binary tree, where the weight of each node is its 1-controllability. An elegant solution to this problem is by Huffman [2]. Though Murgai *et al.* claim an exact solution only in the zero-delay case for general circuits, this solution is exact for Timed Shannon Circuits since the transition activity is delay-insensitive.

The Huffman algorithm, in brief, is as follows. Place all the elements in a list, sorted in ascending order by weight. Choose the two elements of minimum weight, delete them from the list, combine them in a node, and then insert the node in the list; the weight of the node is the sum of the two elements. Repeat this procedure until only a single element remains in the list.

6 Reducing Charging Power: Conditional Selection

The second term of P_{total} will dominate total power consumption when $|E| \gg n$. As a result, the amount of power consumed when the inputs change must be reduced. Consider a primary input x, which fans out to a set of **AND** gates $A = \{A_1, A_2, \ldots, A_k\}$.

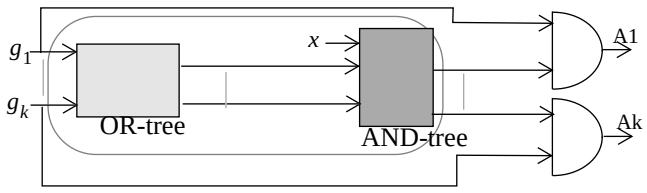
The problem is that on each change in the primary input x, inputs on k **AND** gates must be charged. This is *wasted* power, since a 1 will appear on the input of at most one of the k **AND** gates. For example, in the circuit of Figure 1 exactly one of the four shaded gates has a transition under any input vector. The information on which **AND** gate is actually needed is available in the Shannon Circuit and can be utilized to disable those primary input leads not required for the current input vector.

There are two approaches based on this idea: conditional selection tree and dominator selection.

6.1 The Conditional Selection Tree

The first approach is schematically depicted in Figure 2. In this case, there is a set of **AND** gates $\{A_1, \ldots, A_k\}$ in the original circuit. One input of each gate A_i is connected to g_i and the other to x. The circuit contained in the oval drawn with a dotted line in Figure 2 is inserted between the leads from x and g_i to A_i. The new circuit consists of two subcircuits. The shaded subcircuit chooses minimal-weight subsets $I \subseteq \{1, \ldots, k\}$ and computes functions $(g_{i(1)} \vee \ldots \vee g_{i(j)})$ for $I = \{i(1), \ldots, i(j)\}$; the crosshatched subcircuit takes in these functions and the primary input lead x and computes those functions which replace the input lead x to each gate in $\{A_1, \ldots, A_k\}$. Thus, the fanout capacitance on x has been reduced in return for increased activity in the depicted subcircuits.

FIGURE 2. Conditional Selection Tree Schematic



The **OR**-tree is a minimum weighted binary tree of two-input **OR** gates; the output of each **OR** gate is the function $G_I = g_{i(1)} \vee \ldots \vee g_{i(j)}$. The weight of each node of the **OR**-tree is its 1-controllability, which is directly calculated from the (known) 1-controllability of the input leads $\{g_1, \ldots, g_k\}$. The **OR**-tree is composed using the Huffman algorithm [2][3].

The **AND**-tree is an inverted tree of **AND** gates in one-to-one correspondence with the **OR** gates of the shaded circuit. If G_I is a gate in the **OR**-tree, there is a corresponding gate in the **AND**-tree, H_I. Further, the *function* of H_I is $G_I \wedge x_i$. In this manner, the depicted functions at the outputs of the **AND**-tree are obtained: the outputs of the **AND**-tree correspond to the inputs to the **OR**-tree.

The **AND**-tree is obtained by reversing the wires of the **OR**-tree. The first **AND** gate has two inputs, x and the final output of the **OR**-tree. If G_J is the fanout of gate G_I in the **OR**-tree, form $H_I = G_I \wedge H_J$. It is easy to see by induction from the root (the depicted **AND** gate in Figure 2 is actually the first gate of the **AND**-tree) that this yields the desired function $H_I = x \wedge G_I$.

An example appears in Figure 3. In the initial circuit on the left, the input x_i has fanout to five **AND** gates A1, A2, A3, A4, A5 for each of which the 1-controllability of each of the other inputs {g1, g2, g3, g4, g5}is denoted.

6.1.1 Power Analysis of the Conditional Selection Tree

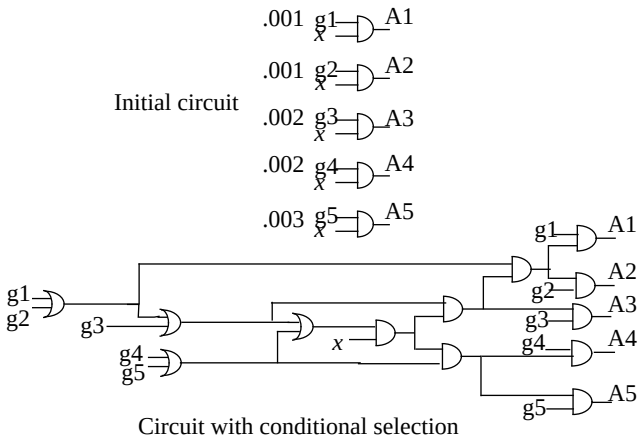
The application of the conditional selection algorithm may not always reduce the power. Hence, it must be applied with care during the algorithm. Fortunately, the determination of whether or not the power is reduced can be done when building the **OR**-tree using the Huffman algorithm. Assume that the input x_i has 1-controllability of p_i and that the next two candidates (each may be a leaf or an **OR** gate in the **OR**-tree constructed up to now) to be combined to form an **OR**-gate have weights w_1 and w_2, with combined weight $w = w_1 + w_2$.

Consider the power consumed without conditional selection applied to these two candidates. The input x_i has a fanout of two to these candidates which consumes power $2p_iP$. The power consumed by the **AND** gates is $2wp_iP$ since the gates charge and discharge once per input vector with probability wp_i. Hence the power consumed due to the two candidates is: $(w + 1)\,2p_iP$.

Now consider the power consumed after conditional selection. The fanout of the input x_i is now one accounting for power equal to p_iP. The **OR** gate has switching probability of w and has fanout one, the associated **AND** gate has switching probability wp_i and has fanout two. Noting that exactly one of the final two **AND** gates switch and that the fanout of each candidate is increased by one due to the construction, the total power is $\left(2w + 2w + 4p_iw + 2p_iw + p_i\right)P$. Conditional selection is only applied on the two candidates when

$$\left(4w + 6p_iw + p_i\right)P < (w + 1)\,2p_iP \quad \text{or} \quad w < \frac{p_i}{4 + 4p_i} .$$

FIGURE 3. Conditional Selection Tree Example



Since the right-hand side of the equation is monotonic over the interval [0,1], solving at $p_i = 1.0$ (*i.e.*, x_i always switches) shows that conditional tree selection can only pay when $w < \frac{1}{8}$; for $p_i = 0.5$, conditional tree selection is advantageous only when $w < \frac{1}{6}$.

Note that the power analysis of this section is on a level-by-level basis through the conditional-selection circuit. Hence it is possible to build a partial conditional selection tree, terminating the Huffman algorithm when the inequality above is not satisfied. In this case x_i will have number of fanout equal to the number of roots of the **OR**-forest.

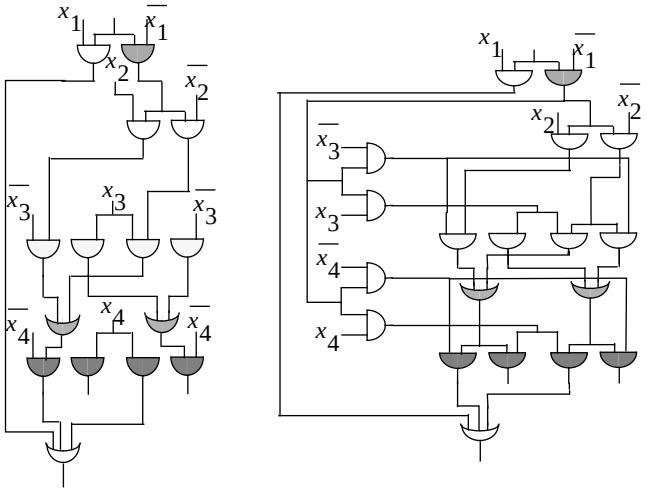
6.2 Dominator Selection

An edge e in a Shannon Circuit is said to *dominat*e a gate g if e precedes g on every path that starts from the **Enable** signal and contains g. An example of a dominator appears in Figure 4. Each of the shaded gates g in the circuit will not rise unless e has first risen: thus, the input pins on these gates need not be set to their correct values unless e rises. This observation is utilized in reducing the load capacitance of primary inputs that are connected to **AND** gates.

Given a set of AND gates that have one input connected to primary input x, one can **AND** the output of e with x and replace the input x of g with the output of this AND gate. This has demonstrable effects on power consumption when e dominates several **AND** gates in the Shannon Circuit that have a common primary input as one of their inputs.

In the example circuit on the left in Figure 4, the crosshatched gate dominates the shaded gates. The key point is that the input transistors on the shaded gates need not be charged unless the crosshatched gate evaluates to 1. The result of applying the dominator selection technique for the primary inputs x_3 and x_4 is shown on the right. Note that the positive and negative phases of the primary inputs are considered separately, hence four new **AND** gates are created. The new circuit computes the same function as the old; in fact, the connections from the cross-hatched gate to the new **AND** gates are redundant for stuck-at-one value. Although the fanout of the four new **AND** gates has not decreased, the probability of these gates switching (or 1-controllability) is much less than the probability of the primary inputs x_3 and x_4 switching. Hence the overall power can be expected to be reduced. This analysis is performed below.

FIGURE 4. Dominator Selection in Shannon Circuit



The algorithm to find all the dominators of a gate is linear in the size of the Shannon Circuit and is obtained by performing a traversal in reverse levelized topological order from the gate towards the **Enable** signal.

There may be more than one dominator for some gates. In this case, the dominator with the lowest 1-controllability is considered. In addition, a dominator is only considered for the above transformation if at least two AND gates connected to the same primary input are dominated by the dominator.

6.2.1 Power Reductions Due to Dominator Selection

The power saving due to dominator selection is due to the fact that dominators, in general, are off most of the time. If the probability of switching for an input x_i is denoted p_i, then the expected unbuffered power consumption to charge k gates is $p_i kP$.

If the probability of a dominator g of those k gates evaluating to 1 is denoted p_g, then the buffered power is derived as follows. The input x must now charge only one gate with power consumption of $p_i P$. There is one new **AND** gate with k fanout that undergoes a transition with probability $p_i p_g$, giving a total transition power consumption of $p_i p_g kP$. There is an increase in fanout of one on the dominator which accounts for additional power of $p_g P$. Recalling that **Enable** is set to high and low for each input vector yields the total buffered power consumption $p_i P + 2p_g\left(p_i k + 1\right)P$. A saving in power consumption is achieved if $\frac{p_i(k-1)}{2\left(p_i k + 1\right)} > p_g$

This expression is evaluated for each dominator to determine whether it is worth buffering. While the exact value of p_g at which it is worth buffering is dependent upon the constants, asymptotically the limit as k tends to infinity is 0.5.

6.2.2 Multiple Dominators: Dominating Sets

It is often the case that a node in a Shannon Graph has only one dominator, the root, which is useless for power reduction. In this case dominator selection can still be performed using a *dominating set*. A dominating set is simply a set of edges in the Shannon Graph such that each path to the node must pass through one edge in the set. In the case where a dominating set is chosen, the edges of the set are OR'ed together to form the input; the probability p_g of the buffering input is the sum of the 1-controllability of the edges in the dominating set.

7 Multiple-Output Circuits

A difficulty with the Shannon Circuit approach is that, as written, each output is a separate circuit. Thus, sharing of gates between circuits, as in the case of most logic design styles, is not done *a priori*. Hence special consideration is made for multiple-output circuits.

The most straightforward approach is to perform post-processing to replace two or more identical sub-functions by a single equivalent sub-function. In addition, we consider two other strategies to attempt to increase sharing across output functions.

7.1 Multiple-Terminal Shannon Graphs

In order to use a single Shannon Graph to compute more than one output simultaneously, the Shannon Graph can be modified to have more than two terminals, 0 and 1. This technique has already been proposed as a method of representing large, structured matrices. We use a Shannon Graph with 2^m terminals to represent an *m*-output function. The function is formed exactly as before; the only difference is that instead of a single output generated from the 1 terminal, there are now m outputs generated from the 2^m terminals. The i^{th} bit of each *m*-bit terminal is connected to the input of the **OR** gate denoting the i^{th} output, if this bit is 1.

7.2 Explicit Sharing

Shannon Graphs for multiple outputs are regarded as Shannon Graphs with multiple roots. An example appears on the left in Figure 6.

The shaded nodes in this figure represent shared nodes between the two Shannon Graphs. If this sharing in the Shannon Graph is to be exploited in the Shannon Circuit, the problem is obvious just from considering a traversal on the Shannon Graph. Assume we get to the 1 terminal: how do we know which of the two outputs is 1, or whether both are?

There are two methods that may be used to resolve this difficulty: Time-Division Multiplexing and Disambiguation Lines.

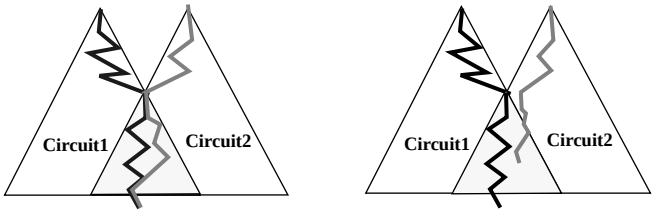
7.2.1 Time-Division Multiplexing

Time-Division Multiplexing requires one enable line per output and uses time to disambiguate which output is being computed. The Enable line for the first output is set to 1 and all others to 0; the value of this output is then read. The Enable line for the first output is then set to 0 and the Enable for the second line is set to 1 and the value of the second output is read. The process is repeated for each output, in turn, until all the output values have been computed. All the Enable lines are then set to 0, the inputs are changed, and the cycle is repeated.

7.2.2 Disambiguation Lines

A second approach to sharing computes all the outputs at the same time, but uses a small amount of additional logic to remove the ambiguity associated with the shared logic.

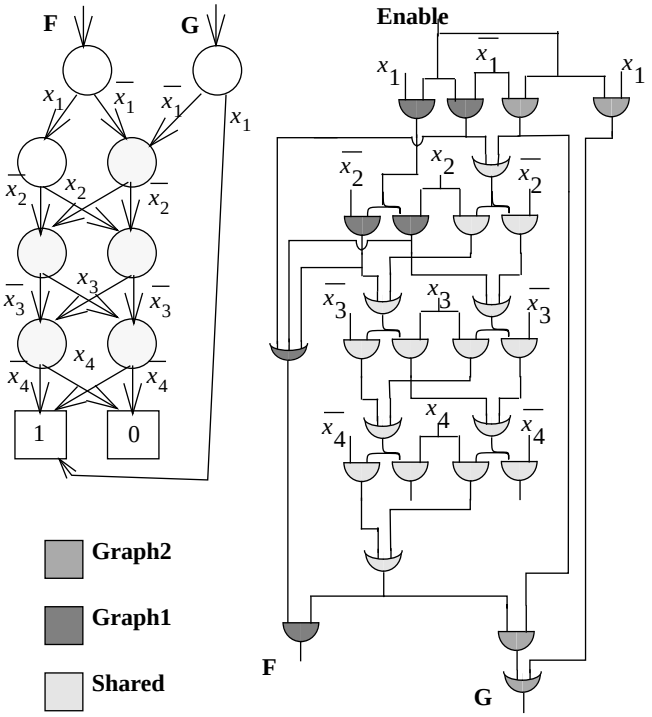
FIGURE 5. Schematic of Shared Output Disambiguation



This is shown schematically on the left in Figure 5. Assume there is a rising signal on the output of the shared logic. For which output did this rising signal emanate? Did this signal emanate from the shaded path (in which case output 1 should rise); or from the striped path (in which case output 2 should rise); or from some third, shared path (in which case both should rise)?

If a 1 propagates to the shared logic from Circuit1, then there must be a 1 on one of the wires crossing from the un-shared part of Circuit1 into the shared section of the circuit, and similarly for Circuit2. Hence, the approach is simply to **OR** together the wires from the unshared section to the shared section, **AND** the result with the shared output and **OR** the result of that **AND** to the output of the Shannon Graph in question. The result of this on the example circuit is shown in Figure 6.

FIGURE 6. Disambiguation of Two-Output Circuit



The disambiguation logic described up to now is enough to resolve the ambiguity in the example circuit, but could lead to errors if it is applied carelessly. This is shown schematically on the right in Figure 5. The shaded path in Circuit2 and the bold path in Circuit1 are active. The shaded path does not terminate at an output. The bold path, however, goes all the way to the output and the correct answer in this case is Output1 = 1, Output2 = 0. However, if the disambiguation logic given above is blindly implemented, the reported Output2 is 1! Note that an edge crossing the boundary from the unshared part of Circuit2 is 1, and the output of the shared part is 1. This is all that is required, according to the discussion above, to set Output2 to 1.

The problem is that there are two distinct entry points to the shared section (this only occurs if the shaded and bold paths enter the shared circuit at separate points, since there can only be one active path from any point in the circuit), and both of these are set to 1. The solution is to bar this possibility by enforcing that separate entry edges to a shared section, starting from different roots, have disjoint characteristic functions (the characteristic function of an edge is simply the set of input vectors which set that edge to 1). This rule is easily enforced (logic is duplicated in this case, rather than shared), and easily checked on Shannon Graphs. Note that the amount of logic to implement disambiguation is small: generally two **OR** gates and an **AND** gate per output. In addition, there is a maximum of three additional transitions per output.

8 Pipelining

Occasionally, it is uneconomic to form the Shannon Graph for a circuit, e.g. a combinational multiplier. It is possible to use the proposed technique even in this case by partitioning the circuit into a set of subcircuits. The outputs of each sub-circuit form inputs to the next. Latches are used to separate the outputs of one pipeline stage that are inputs to the succeeding pipeline stage. The Shannon Graphs for each sub-circuit are then created and evaluated in a pipeline fashion.

Pipelining may also be used in a multiple-output circuit when an output function can be compactly expressed as a function of some other output functions. By computing the latter outputs first, the former may be evaluated with potentially lower power consumption.

9 Results

Experimental results comparing the power consumption of Timed Shannon Circuits against area optimized circuits are summarized in this section. The complete set of two-level examples from the MCNC benchmark suite is considered. A multi-level circuit optimized for area is created for each example using the standard script, called *script.rugged* in the logic optimization program SIS available from U.C. Berkeley. Next the circuit is decomposed into a network of two-input simple gates (*i.e.* **AND**, **OR** gates with possible inversions on the inputs and outputs). This circuit is used for estimating the power consumption at the technology independent level. This circuit is mapped to the gates in a given technology library, *lib2.genlib* in this case, to obtain an area-minimal implementation in a specific technology. This circuit is denoted the optimized circuit.

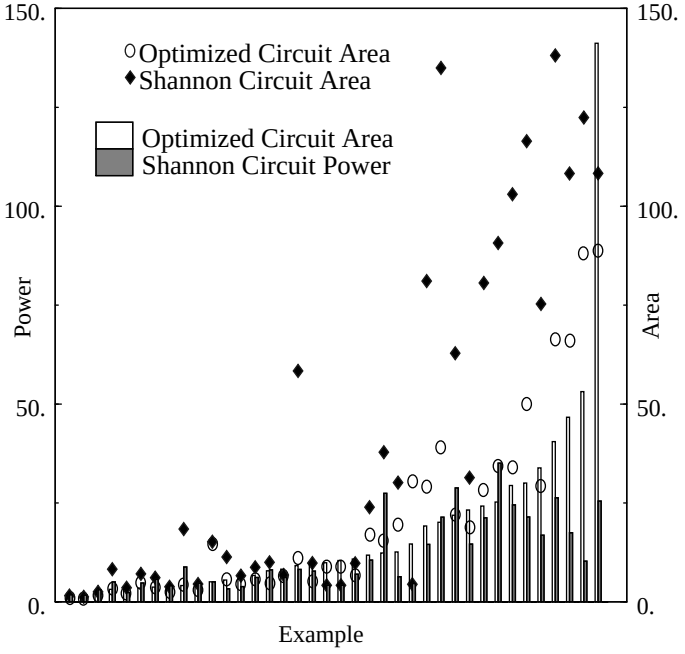
Power consumption is estimated using a timing simulator to determine the transitions at each gate in a circuit. In this experiment the power consumption is estimated by obtaining the power consumption using 1000 randomly generated input vectors. The power consumed at a gate is measured using the formula described in Section 2. Since the quantities V and f are constants for a circuit, the total power is given as $\sum_i p_i C_i$. The load capacitance C_i on each gate i in the mapped circuit is obtained from the characterization provided by the library.

The Timed Shannon Circuit is created for each circuit using the BDD package available in SIS. The area and power optimization techniques outlined in Section 5 and Section 6 are employed in obtaining a circuit that is minimal with respect to power consumption. During the conditional selection process a two-input simple gate model, with load capacitance equal to the number of fanout, is used to estimate the power consumed before and after each transformation. Technology mapping is performed using the map command in SIS to realize an

area minimal circuit with no regard to either the delay or power consumption of the resulting circuit.

The Shannon Circuits consume less power than the optimized circuits in 23 of the 38 examples, with the average power ratio between the optimized and Shannon circuits being 1.87 (using the technology dependent power consumption). This corresponds to an average reduction in power of 47% for these 23 examples. The area of the Shannon Circuit is less than that of the optimized circuit for 3 of these examples. Excluding these from consideration, the average area ration between the optimized and Shannon circuits is 0.56, which corresponds to an average area increase of 44% for the 20 examples where the Shannon Circuit consumes less power but occupies more area.

FIGURE 7. Power and Area Comparison between Optimized and Timed Shannon Circuits



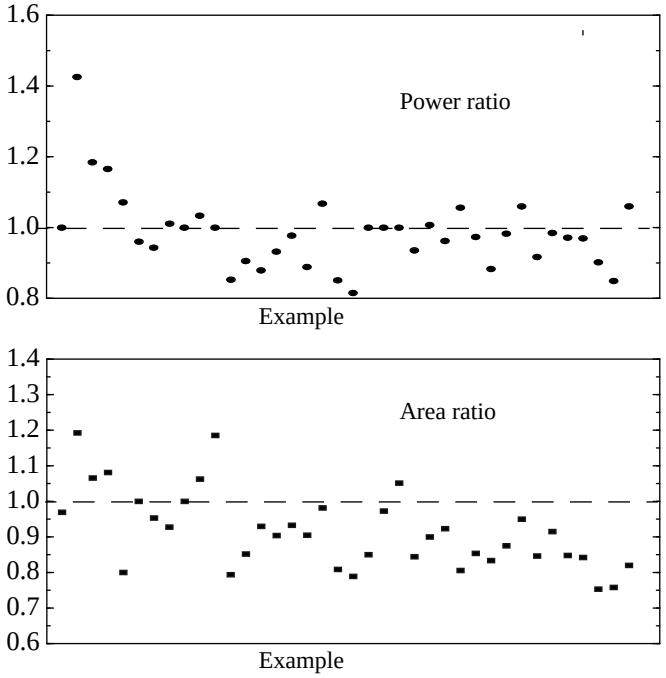
Although the power consumed by the Shannon Circuit is larger than the optimized circuit for 14 of the 38 examples, note that 9 of these 14 examples are very small circuits. The difference in the power consumption on 2 of the remaining 5 circuits is very small. However, the power consumed by the Shannon Circuits for three examples is substantially higher than the power consumed by the optimized circuit. The principal reason for this appears to be the lack of logic sharing between the outputs in the Shannon Circuit implementation of these circuits versus the corresponding area-minimal optimized circuit. In fact, for all of these circuits the average number of transitions per input vector in the area-minimal circuits is substantially smaller than in the Shannon Circuit. The investigation of alternative techniques to achieve better logic sharing among outputs remains for the future.

Figure 7 illustrates the comparison between the total power consumed in the optimized and Shannon Circuits. The examples are shown in increasing order of power consumption and the power reported is normalized to the power consumed by the smallest circuit. The normalized area for the optimized and Shannon circuits is also shown in the figure. Notice that while the optimized circuits have a wide range of power consumption, the Shannon Circuits appear to have bounded power consumption. Moreover, the Shannon Circuit power is dramatically less than the optimized circuit for the circuits with higher power consumption.

Figure 8 illustrates the correlation between the area and power ratios between the area optimized circuits and the Shannon circuits at the technology indepemdent and technology dependent levels. Observe the good correlation between the power and area estimates at the technology dependent and technology, which indicates the validity of the model used in the algorithms.

No conclusions are apparent about the trade-off in the delay between the Shannon Circuit and optimized circuits. On about half of the circuits where the Shannon Circuits consume less power, the delay of the technology mapped Shannon Circuits is also less than that of the optimized circuit. However, note that the technology mapping algorithm has been invoked to realize an area-minimal circuit irrespective of the resulting delay. A more comprehensive experiment to explore the area-delay-power trade-off remains for the future.

FIGURE 8. Ratio of Technology Independent andTechnology Dependent Power and Area Ratios between Optimized and Timed Shannon Circuits



References

[1] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Broderson. Hyper-LP: A system for power minimization using architectural transformations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 300-303, November 1992.

[2] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proceedings of the IRE*, volume 40, pages 1098-1101, September, 1952.

[3] R. Murgai, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Decomposition for minimum transition activity. In *Proceedings of the Low-Power Workshop*, Napa, April 1994.

Acknowledgments

Useful discussions on various aspects of low-power circuits with Jerry Burch, Ken McMillan, Patrick Scaglia, Lou Scheffer, Robert Brayton, Roberto Guerrieri, Ricky Ho and Luca Carloni are gratefully acknowledged. Philip Blair provided critical feedback on an early draft of this paper.