Data Driven Power Optimization of Sequential Circuits [*]

Qi Wang, Sarma B.K. Vrudhula
Center for Low Power Electronics
ECE Dept., University of Arizona
Tucson, AZ 85721
email: qi,sarma@ece.arizona.edu

Abstract

In this paper we present an efficient technique to reduce the power dissipation in a technology mapped CMOS sequential circuit based on logic and structural transformations. The power reduction is achieved by adding sequential redundancies from low switching activity gates to high switching activity gates (targets) such that the switching activities at the output of the targets are significantly reduced. We show that the power reducing transformations result in a circuit that is a valid *replacement of the original. The notion of validity used here is that of a* delay safe replacement *[11, 12]. The potential transformations are found by direct logic implications applied to the circuit netlist. Therefore the complexity of the proposed transformation is polynomial in the size of the circuit, allowing the processing of large designs.*

1 Introduction

Reduction of power consumption in digital circuits has become an increasingly important design optimization goal. At the stage of logic optimization, dynamic power dissipation can be reduced by minimizing the switching activities of the circuit.

Several different approaches to reducing switching activity in combinational logic circuits have been proposed ([2, 4, 7, 9, 10, 13, 14]). The more recent of these methods [2, 9, 10, 14] are based on applying structural and logical transformations directly to the circuit netlist. These are generally faster than their algebraic counterparts, and can be applied to much larger circuits. Obviously, one way to modify the structure of a circuit without modifying its function is to add or remove (or both) redundant connections and gates. Systematic techniques to do this to improve some cost function (e.g., area, delay or power) are called *rewiring*, and these are based on ATPG (automatic test pattern generation) techniques [10], or implication analysis [2, 9, 14]. Rewiring of combinational circuits is based on identifying untestable logic faults, since they are guaranteed to be redundant.

Power reduction in sequential logic circuits has also received some attention recently, albeit to a much lesser extent. In [7], the reduction is attempted by *re-timing* the latches, without modifying the combinational logic that is bounded by the latches. The objective was to reduce the amount of glitching in the circuit - a quantity that can sometimes account for as much as 30% to 70% of the dynamic power dissipation. This approach does not exploit the *sequential dependencies* in the signals, which can potentially result in significant reductions in switching activity. Furthermore, restricting the power reducing transformations to only the combinational logic between latches may not be very effective for high performance systems that are typically highly pipelined, and have latch-intensive circuits, with only small subcircuits of combinational logic between latches.

In this paper we present a new technique for reducing the power dissipation in synchronous sequential logic circuits that are implemented in static CMOS technology. In our model of sequential circuits, no global reset line is assumed. The method to be described is based on extending the implication based rewiring techniques of [14] to be applicable to sequential circuits. We will refer to this as *Sequential Rewiring* (SR).

It is important to note that SR is considerably more difficult than its combinational counterpart since in a sequential circuit, an untestable fault is not necessarily redundant [1, 6]. Consequently, redundancy identification in sequential circuits is a much more difficult problem than for combinational circuits, especially when some latches do not have global reset lines. When a sequential circuit C is rewired, the new circuit C' must be a *valid* replacement of C. The notion of *validity* used here is that of a *safe replacement* [11]. Simply stated, a replacement of a circuit is said to be *safe*, if the external environment cannot distinguish between the original and the replacement by only observing the input and out-

[*]Copyright 1998 EDAA. Published in the Proceedings of DATE'98, February 23-25, 1997 in Paris, France. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this 5work in other works, must be obtained from EDAA.

put behavior [11]. This property is clearly important to satisfy especially when there are no global reset lines. Furthermore, it is general enough without the stringent requirement of having each state in C be equivalent to some state in C'.

The work of [3] was perhaps the first attempt to apply SR techniques for reducing the area of sequential circuits. However it was shown [6] that their transformations would not result in a *safe replacement* of the original circuit.

The important contribution of the SR method presented here is that it will result in a circuit that is a *delay safe replacement* [11] of the original, and which has less dynamic power consumption. Our method is different from the work of [7] since it involves logic transformations that cross the latch boundaries. The proposed method is based on implication analysis carried out on the circuit netlist, and is different from some previously published work on sequential optimization for area, e.g. [8], which are based on the manipulation of state transition graphs (STG) using BDDs. Note that the size of the STG is exponential to the number of latches in the circuit. If only direct implications are used, the time complexity and space requirement of the proposed method is polynomial in the size of the circuit. Therefore, the proposed approach can be very efficient and suitable for very large designs.

The rest of this paper is organized as the follows. Section 2 provides a brief background to the notion of safe replacement and combinational rewiring techniques. In Section 3, the proposed Sequential Rewiring method is described. Experimental results are presented in Section 4.

2 Preliminaries

In this section we present a very brief introduction to the notion of *safe replacement* of sequential circuits and related concepts. Details are available in [11, 12]. Following this, the power and delay models used in the calculations are described. Finally, a summary of combinational rewiring methods for reducing power are given.

2.1 Design Replacement

Definition 1 *A* **deterministic Finite State Machine** *(DFSM) M is a 5-tuple, (Q,I,O,λ,δ), where Q is the set of states, I is the set of input values, O is the set of output values. The output function λ : (Q $\times$ I) $\rightarrow$ O and the next state function δ : (Q $\times$ I) $\rightarrow$ Q are completely-specified functions.*

Definition 2 *[11] A design D_1 is a* **safe replacement** *of a design D_0 (denoted by $D_1 \preceq D_0$) if given any state $s_1 \in D_1$ and any finite input sequence $\pi \in I^*$, there exists some state $s_0 \in D_0$ such that the output behavior $\lambda_{D_1}(s_1,\pi) = \lambda_{D_0}(s_0,\pi)$.*

The notion of safe replacement is the least stringent and sufficiently general in the sense that that no prior knowledge or assumptions are made regarding the design's environment [11]. This is important for design optimization since we almost always prefer to obtain an optimized design which can replace the old design without any constraints or restrictions. In practice, a design will be used only after some number of clock cycles have elapsed after power-up to ensure that all the voltages and currents have settled before any useful computation can proceed. Further optimization is possible by taking advantage of this delay.

Definition 3 *[12] Given a design D, the* **n-cycle delayed design** *(denoted by D^n) is the restriction of D to the set of states $\{s| \exists\pi \in I^n,\ s' \in D : \delta_D(s',\pi){=}s\}$, i.e. a state s belongs to D^n if and only if there exists a power-up state s' in D and an input sequence of length n which drives s' to s. A new design C is an* **n-delay safe replacement** *for D if $C^n \preceq D$.*

The following two lemmas show some of the important and useful properties of the of n-delay replacement.

Lemma 1 *[12] If $C^n \preceq D$, and $m \geq n$, then $C^m \preceq D$.*

Lemma 2 *[12] If $C^n \preceq D$ and $B^m \preceq C$, then $B^{m+n} \preceq D$.*

The property of n-delay safe replacement is compositional [12]. This is important because a sequence of transformations will be an n-delay safe replacement if each of the individual transformations is a n-delay safe replacement. The total delay will be the sum of the individual delays.

2.2 Power and Delay Model

The total switching power is computed using Equation 1.

$$P = \frac{1}{2} \times \frac{{V_{dd}}^2}{T_{cycle}} \times \sum_{\forall gates\ i} C_{load}(i) \times E(i). \qquad (1)$$

$C_{load}(i)$ and $E(i)$ denote the load capacitance and expected number of logic transitions on the output of gate i. T_{cycle} denote the clock cycle time. With V_{dd} and T_{cycle} being fixed, the power consumption of a CMOS circuit can be estimated by summing the average switching activity of each node weighted by its load capacitance. For brevity, we will refer the term $C_{load}(i) \times E(i)$ as the *switching capacitance* of gate i.

It is assumed that technology mapping has already been done, and the load capacitance of each gate is computed from the technology library. An event driven simulator is used to obtain the average switching activity at the output of each gate given the input binary waveforms. The delay model used in the simulator is a simple linear model, given by Equation 2. Again, all the parameters in the equation can be obtained from the technology library.

$$gate\_delay = transport\_delay + unit\_fanout\_delay \times C_L. \quad (2)$$

2.3 Related Work: Combinational Rewiring for Low Power

The proposed method is an extension of the implication based transformations for low power that were developed for combinational circuits [14]. As these transformations form the basis of the present work, we present a brief summary of the methods. Details can be found in [14].

In [14], a power reducing transformation is found by logic implications. The gist of the method is to add redundant connections from *low activity* gates (called *sources*) to *high activity* gates (called *targets*) so as to reduce the switching activity of the latter. The key step is to identify such source and target gates so that connections between them will be redundant. The identification is done by logic implication. The methods presented in [14] guarantee reduction of the total switching capacitance.

The implication starts by setting the output of a *source* (low activity) gate to a logic value v. The value of v is determined by the 0-1 or 1-0 signal transition probabilities. Using logic simulation, v is propagated forward and justified backward. At the end of this implication step, a subset of gates will have logic values assigned to them. The next step is called $u - propagation$, which is is conducted as follows. An input of a gate is marked by u if (1) another input of the gate has been assigned the controlling value of the gate; or (2) the output of the gate is marked by u. A fanout stem is marked by u if and only if all the fanout branches are marked by u. Finally, a gate is marked by u if its fanout stem is marked by u. While the logic value v can be propagated backward and forward, the marking of u can only be propagated backward. It was shown in [14] that if a gate (called *target*) is only marked by u (as opposed to being marked by u and v or $\overline{v}$) at the end of implication procedure, and the target is not in the fanin cone of the source, then connection from the source to the target is redundant, and the total switching capacitance will be reduced.

3 Sequential Rewiring for Low Power

We now describe the method for rewiring a sequential circuit with the objective of reducing its power consumption. The key problem here is to guarantee that connections that are added or removed are *sequentially redundant* and the transformed circuit will be a safe replacement of the original. Without loss of generality, we assume the only memory elements in a sequential circuit are DFFs. During the implication process, the linear iterative array model of a sequential circuit is used.

3.1 Sequential Implication

The implication analysis on a sequential circuit requires two propagation rules (similar to those in [5]) in addition to those given in Section 2.3. If a DFF's output is marked by u, then its input is also marked by u. Whenever a logic value or a u mark is propagated through a DFF, the adjacent time frame is entered. Let R (L) to denote the rightmost (leftmost) time frame of the implication, where $R \in Z^+$ and $L \in Z^-$. Since it is possible that the implication may never stop, we assign a maximum number of time frames (backward and forward) that the implication may proceed. These are denoted by R_{max} and L_{max} respectively.

3.2 Sequential Redundancy Addition

We now describe, by the way of an example, the method used to find a sequential redundancy, which will be added to circuit so that the switching capacitance of the circuit is reduced.

Consider a design D_0 shown in Figure 1(a). Switching activities of some of the gates are shown diagramaticaly. The notation 0^{-1} indicates a logic 0 in time frame -1. It can be seen from the figure that the outputs of the NAND gate e and the DFF D_2 have very low switching activity, and the output of the AND gate d has very high switching activity. Gate d also fans out to two other gates. Furthermore, the waveforms of e and D_2 show that the two signals have long runs of 0s. Now if a logic 0 is implied at the output of D_2 then $Q_2 = 0^0$, $e = 0^{-1}$, $b = 1^{-1}$, $c = 1^{-1}$, and $Q_1 = 1^0$. After logic implication, u-propagation is done. This results in connections $\{d \rightarrow g,\ d \rightarrow f\}$ being marked u^0, and therefore d is marked u^0. Now if we add the connection from Q_2 to d we obtain a new design.

The switching activity at the output of d will be reduced due to the fact that the signal added as input to d has, on the average, long runs of $0's$, and 0 is the controlling value of an AND gate. Furthermore, this reduction may have equally positive impact on the nodes in the fanout cone of d. Now, to see if the new design D_1 is a delay safe replacement of the original design D_0, consider the STG's of these two designs, which are shown in Figure 2. It can be seen from the two STGs that D_1 is not a safe replacement of D_0. For example if the input is (abc)=(101) and the two designs both start from state 00, the output of D_0 is (10) while the output of D_1 is (00). However D_1 is a 1-delay safe replacement of D_0. This is because if D_1 is clocked once before applying any useful inputs, then the only reachable states by D_1 are 11, 01 and 10. For each of the three reachable states there is a state in D_0 that under any input sequence, the outputs of the two designs are the same. The observations can be generalized as follows.

Theorem 1 *Given a node s in a circuit C, let t be an AND gate marked only by u in the time frame $T \geq 0$ after performing a sequential implication starting from $s = 0$ in time frame 0. Let C_r be another circuit obtained by adding the connection from s to t with T DFFs inserted from s to t. Then $C_r^n \preceq C$ if:*

1. *The connection from s to t is valid (see Section 3.3)*
2. *$t \notin TFI(s)$, where $TFI(s)$ is the combinational transitive fanin cone of s,*

where $n = |L| + T$ and L is the leftmost time frame at which the sequential implication process terminated.

In the previous example, $T = 0$ and $L = -1$. This means that no DFFs were added and the circuit in Figure 1(a) is a 1-delay replacement of the circuit in Figure 1(b). Similar theorems can be derived for other implication cases. In general if $s = v$ and v is the controlling value of t, a connection can be added from s to t; otherwise an inverted connection is needed. According to Lemma 2, a sequence of transformations T_1, T_2, $\cdots$ which are n_1 , $n_2, \cdots$ safe delay replacements respectively, and which reduce the switching capacitance in the manner described above, will also result in a n-delay safe replacement, where $n = n_1 + n_2 + \cdots$.

3.3 Validity of Sequential Implication

In the rewiring procedure described above, the logic implication is performed on the original circuit. It is possible that performing the same implication on the transformed circuit we may produce different results. In particular, if a target gate is marked by u in the original circuit but cannot be marked u in the transformed circuit, then the connection for the source to the target cannot be added.

Consider the example shown in Figure 3. We represent the sequential circuit as an iterative array which expands two time frames. The signals x_1 and x_2 are the present state lines and y_1 and y_2 are the next state lines. Suppose the implication starts at the output of c with logic value 0. Then this results in $f = 1^0$, $e = 1^0$, and $x_2 = 0^1$, $a = 1^1$, $b = 1^1$, $c = 0^1$, $f = 1^1$, $e = 1^1$. After u propagation, the stem at the output of b is marked by u. Now consider the circuit after the connection from c to b is added. This is shown in Figure 4. If the same implication is performed on the transformed circuit, $b = 0^1$, instead of 1^1 as in Figure 3. Performing u propagation on the transformed circuits results in b not being marked by u. In this case the connection from c to b is *invalid*. An examination of the STGs of the two circuits will reveal that they are not equivalent, i.e., adding the connection will modify the function of the circuit.

To determine whether a connection to be added is *valid* or not, another complete implication step, followed by u propagation, may be necessary. This can become very time consuming if the number of source and target pairs is large. However, the following lemma gives a sufficient condition for a candidate connection to be valid. It is a conservative claim, i.e., a candidate connection may be valid even though the condition in Lemma 3 is not satisfied. However, the use of this lemma will significantly reduce the computation time since it obviates additional implication analysis for each candidate connection.

Lemma 3 *Let s be a source gate of the implication analysis and let t be a target gate that is marked with a u in some time frame $T \geq 0$. If, in addition, t is never marked with a logic value in any time frame during the implication process, then adding the connection from s to t is valid.*

3.4 Implementation

The sequential rewiring method was implemented in a package called iLOOPS (implication based LOgic Optimization for Power on Sequential circuits). A outline of the algorithm is shown in Figure 5.

Inputs to iLOOPS is a circuit C, the power-up delay value N, and a delay constraint d (maximum allowed percentage increase in the clock period). We assume that the switching information at the output of each gate in C is computed either via simulation or by signal and transition probability propagation. The sequential implication at line 6 will return D and L, where L is the leftmost time frame reached by the implication step, and D is a dictionary whose elements are of the form (T_i, g_i), where T_i is a time frame of a target gate g_i that is marked by u. Note that according to Theorem 1, T_i DFFs must be added if the connection is valid. To avoid the potentially severe area penalty, at present only those connections with $T_i = 0$ are considered. If a connection is valid (see Lemma 3), it is added and the associated delay cycle value is accumulated. If the accumulated delay cycle is greater than N, the procedure terminates and returns the transformed circuit and the total delay value. Otherwise the algorithm continues until no more source nodes can be found.

4 Experimental Results

The sequential rewiring algorithm was implemented in a program called *iLOOPS*. Using the gate delay model given in Equation 2, each technology mapped circuit was simulated using binary waveforms. The switching activity and the average lengths of '1' and '0' runs at each node were computed. This information is used to select the source nodes and the logic value to start the implication process. The power for each circuit is estimated by using the Equation 1. Post transformation static timing analysis was also done to measure the impact of the transformations on the performance. The maximum allowed increase in delay was set to 3%.

The MCNC91 benchmark circuits were used as test cases. Since the fundamental difference between sequential rewiring (which includes combinational rewiring techniques) and combinational rewiring is that the former optimizes across latch boundaries, the benchmark circuits were converted to highly pipelined circuits by inserting latches. The combinational logic between the latches was synthesized using SIS-1.2 with the *script.rugged* file and technology mapped using *lib2.genlib* library and mapping command *map -s -n 1 -AFG -p*.

The results on these circuits are shown in Table 1. The third column shows the number of DFFs in the circuit. The fourth, fifth and sixth columns show the percentage power reduction, delay increase and area increase as a result of sequential rewiring. The column labeled PR(CR) is the power reduction due to pure combinational rewiring. Since the portion of the power consumption due to the clock signal cannot be reduced using the transformations, a column labeled PREC is included, which indicates the power reduction excluding the clock signal. The column labeled by *Delay* indicates the value of n, where the transformed circuit is an n-delay replacement of the original.

4.1 Summary of Results

Overall, the average power reduction due to sequential rewiring was 9.23% while the corresponding number for combinational rewiring was only 2.5%. Note that these numbers include the clock power. Not surprisingly, the power reduction is doubled when the power consumed by the clock signals is not considered. Therefore, in general the proposed sequential rewiring technique is very effective.

The number of delay cycles n (of the n delay replacement) for all the circuits is not very high. This is because in the sequential implications, most of the logic values can only propagate forward within a small local sub-network. For the delay penalty (column labeled *DI*), the simple heuristic of updating the timing locally after adding a connection works well (most transformations satisfied the delay constraint) for all cases except for *term1* and *t481*. Based on our experience with combinational rewiring [14], we believe that this high delay increase can be eliminated by simply removing a small portion of the added connections. However, our current implementation does not yet include such a step and other features such as post transformation delay optimization [14]. Note that some circuits have a large number of DFFs, and the proposed sequential rewiring approach can easily result in significant power optimization with very little computational effort.

5 Acknowledgement

This work was carried out at the Center for Low Power Electronics which is supported by the National Science Foundation, the Department of Commerce of the State of Arizona, and various companies in the microelectronics industry, including, Analog Devices, Analogy, Burr Brown, Hughes Aircraft, Intel, Microchip, Motorola, National Semiconductor, Rockwell, Sicom, SMI, and Western Design.

References

[1] M. Abramovici and M. A. Breuer. On redundancy and fault detection in sequential circuits. *IEEE Transactions on Computers*, C-28(11):864–865, Nov. 1979.

[2] R. I. Bahar, M. Burns, G. D. Hachtel, H. Shin E. Macii, and F. Somenzi. Symbolic Computation of Logic Implications for Technology Dependent Low Power Resynthesis. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 163–168, 1996.

[3] L. Entrena and K.-T. Cheng. Sequential logic optimization by redundancy addition and removal. In *Proceedings of the International Conference on Computer-Aided Design*, pages 310–315, 1993.

[4] S. Iman and M. Pedram. Multi-Level Network Optimization for Low Power. In *Proceedings of the International Conference on Computer-Aided Design*, pages 372–377, 1994.

[5] M. A. Iyer, D. E. Long, and M. Abramovici. Identifying sequntial redundancies without search. In *Proceedings of the Design Automation Conference*, 1996.

[6] M. A. Iyer, D. E. Long, and M. Abramovici. Surprises in sequential redundancy identification. In *Proceedings of the European Design and Test Conference*, 1996.

[7] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. In *Proceedings of the International Conference on Computer-Aided Design*, pages 398–402, 1993.

[8] C. Pixley, V. Singhal, A. Aziz, and R. K. Brayton. Multilevel synthesis for safe replaceability. In *Proceedings of the International Conference on Computer-Aided Design*, pages 442–449, 1994.

[9] D. K. Pradhan, M. Chatterjee, M. V. Swarna, and W. Kunz. Gate-Level Synthesis for Low-Power Using New Transformations. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 297–300, 1996.

[10] B. Rohfleisch, A. Kolbl, and B. Wurth. Reducing Power Dissipation after Technology Mapping by Structural Transformations. In *Proceedings of the Design Automation Conference*, pages 789–794, 1996.

[11] V. Singhal and C. Pixley. The verification problem for safe replaceability. In D. L. Dill, editor, *Proceedings of the Conference on Computer-Aided Verification*, pages 311–323. Springer-Verlag, June 1994. 1994.

[12] V. Singhal, C. Pixley, A. Aziz, and R. K. Brayton. Exploiting power-up delay for sequential optimization. In *Proceedings of the European Design Automation Conference*, pages 54–59, 1995.

[13] S. B. K. Vrudhula and H. Y. Xie. Techniques for CMOS Power Estimation and Logic Synthesis. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 21–26, 1994.

[14] Q. Wang and S. B. K. Vrudhula. Multi-level Logic Optimization for Low Power using Local Logic Transformations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 270–277, 1996.

PR = Power Reduction, DI = Delay Increase, AI = Area Increase
PR(CR) = Power Reduction by Combinational Rewiring,PREC = Power Reduction Excluding Clock Power

Ckt	Gates	DFFs	PR	DI	AI	PR(CR)	PREC	Delay	CPU
cu	80	15	17.09%	1.75%	6.97%	2.71%	41.21%	9	0.6
cordic	86	17	9.17%	0.00%	10.22%	8.10%	19.88%	8	0.6
lal	123	23	3.79%	0.00%	8.91%	1.52%	9.00%	6	0.5
pcler8	130	21	8.28%	0.52%	10.00%	0.30%	12.06%	8	0.6
count	182	21	12.98%	2.73%	10.40%	0.00%	19.47%	4	0.7
term1	212	42	6.59%	10.83%	5.94%	3.09%	13.06%	10	0.6
t481	719	31	5.40%	10.88%	2.09%	1.76%	9.18%	5	9.5
dalu	1003	64	6.34%	2.19%	2.15%	1.80%	11.18%	24	8.2
k2	1170	86	13.46%	0.31%	3.58%	2.76%	26.90%	20	30
Average	412	36	9.23%	3.25%	6.70%	2.45%	17.99%	10	5.7

Table 1: Experimental results on sequential rewiring for low power.

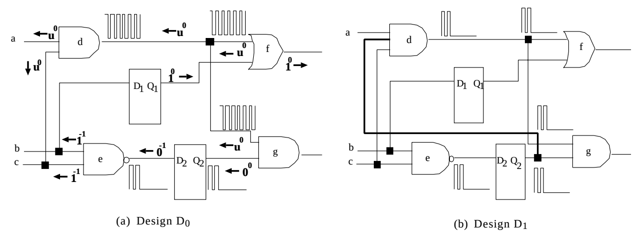

Figure 1: A example of sequential rewiring for power reduction (a) the original design, (b) the new design after the transformation.

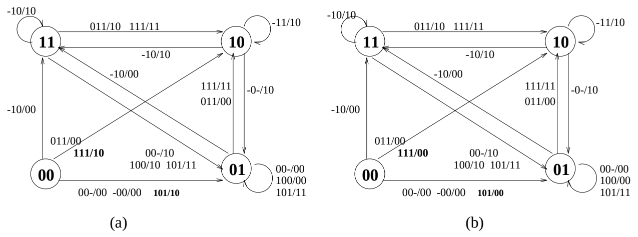

Figure 2: (a) STG of the original design, (b) STG of the design after adding the connection in Figure 1.

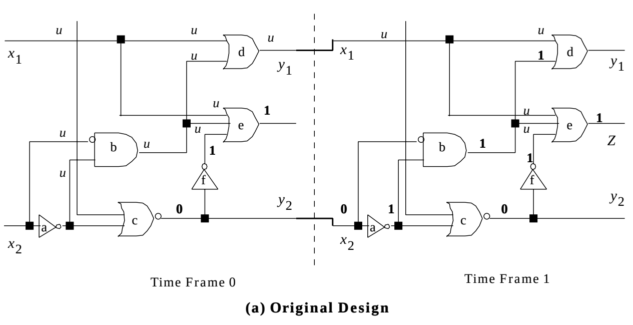

Figure 3: Validity of sequential implications

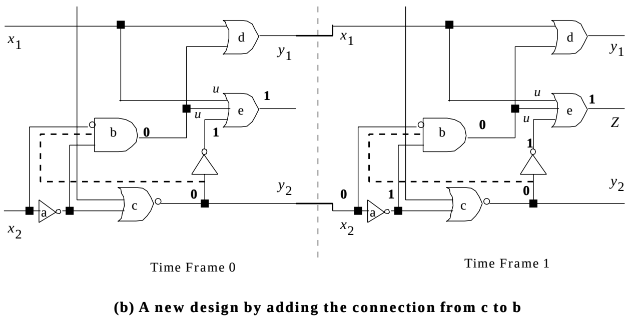

Figure 4: Validity of sequential implications (cont.)

```
procedure iLOOPS (C,N,d) {
0. int cycle =0; boolean stop = false;
1. while(!stop){
2.   select a source $s$ gate with low activity;
3.   if no $s$ can be found { stop = true;}
4.     else{
5.         select logic value $v$ to imply at $s$;
6.         $(D, L)$ = sequential_implication($s$,$v$);
7.         for each time frame $T$ in $D$ {
8.             take a gate $t$ from $D$;
9.             if add connection from $s$ to $t$ is valid {
10.                if add connection will not violate the delay constraint {
11.                  add connection from $s$ to $t$;
12.                  cycle = cycle + |L|+|T|;
13.                  if (cycle $\leq$ N) stop = true;
14.                  } } } } }
15. return ($C$,cycle); }
```

Figure 5: iLOOPS algorithms.