Latency and Latch Count Minimization in Wave Steered Circuits

Amit Singh, Arindam Mukherjee, Malgorzata Marek-Sadowska

Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106, USA
{asingh@cornet.ece, arindam@cornet.ece, mms@ece}.ucsb.edu

Abstract

Wave Steering is a new design methodology that realizes high throughput circuits by embedding layout friendly synthesized structures in silicon. Wave Steered circuits inherently utilize latches in order to guarantee the correct signal arrival times at the inputs of these synthesized structures and maintain the high throughput of operation. In this paper, we show a method of reordering signals to achieve minimum circuit latency for Wave Steered circuits and propose an Integer Linear Programming(ILP) formulation for scheduling and retiming these circuits to minimize the number of latches for minimum latency. Experimental results show that in 0.25µm CMOS technology, as much as 33.2% reduction in latch count, at minimum latency, can be achieved over unoptimized Wave Steered circuits operating at 500 MHz.

1. Introduction

Wave Steering is a layout friendly design methodology that realizes high throughput circuits by embedding synthesized structures in silicon. These synthesized structures are variants of Binary Decision Diagrams (BDDs)[1]. Since the first published work in [9], the Wave Steering methodology has been extended to both ASIC structures[10], reprogrammable FPGA fabrics[11][12] and Finite State Machines (FSMs)[5][6] with as much as 3-4 times throughput increase over non-Wave Steered circuits. In the Wave Steering design methodology, circuits inherently utilize latches to guarantee the correct arrival times at the inputs. Inside the synthesized structures they are used for signal skewing, and on the interconnects they are used to guarantee high throughput. In this paper, we propose an Integer Linear Programming(ILP) formulation for retiming Wave Steered Circuits to minimize the number of latches while keeping the circuit latency minimum. This latch minimization formulation does not compromise on the high circuit throughput which is a unique feature of the Wave Steered methodology.

Wave Steering integrates logic and physical syntheses for a specified clock frequency. It pipelines a BDD-like structure (henceforth referred to as a Logic Block(LB)) to the granularity of one level, which is controlled by a single variable (input signal). This not only makes high frequency pipelining possible, but also introduces *variable ordering* in the LBs as a degree of freedom that is absent in other design styles. Unlike classical micro-pipelining schemes, no logic redundancy needs to be introduced in order to have a high throughput execution in the fine-grained pipelined stages. Since Wave Steering uses a two-phase non-overlapping clocking scheme[9][10], variables that are inputs to consecutive levels have to be applied one clock phase apart. Depending on the arrival times of the different input variables of a LB, an assignment of the variables to levels in the LB can be done so as to reduce the latency of the LB. In order to preserve the high frequency of operation inside an LB, *interconnect pipelining* may need to be done for signals between LBs.

We show a method of scheduling and reordering variables inside LBs, to achieve the minimum latency for a given decomposed and placed netlist of LBs. The scheduling/ordering which achieves this minimum latency for an LB is called a *Minimum Latency Ordering (MLO)* and several such orderings can exist. For this latency, we formulate the problem of minimizing the number of latches. For simplification, we do not consider routing resource constraints/ congestions or circuits with feedbacks in our formulation. We also assume that all LBs are complete binary trees and this prevents any LB deformation after reordering. LBs which are complete binary trees have application in Field Programmable Gate Arrays (FPGAs)[11][12].

We organize the rest of the paper as follows: Section 2 discusses previous work related to area minimization using retiming and points out the difference and similarity with our problem. Section 3 highlights the underlying ideas of the Wave Steering methodology. Section 4 includes definition of key terminologies followed by the problem definition. The next section presents the underlying ideas behind our optimization approach. Section 6 proposes an efficient ILP formulation of this min-latency min-area problem. This is followed by Experimental results and Conclusions.

2. Previous Work

Since the seminal work of Leiserson and Saxe [4], researchers have investigated new ways of using retiming to improving circuit performance. Most work in the field of area and clock period minimization for circuits has used classical retiming techniques put forth in [4]. Retiming involves the relocation of flip-flops across logic gates to allow the circuit to be clocked at a faster clock speed (minperiod retiming). Minarea retiming then aims to minimize the number of flip-flops[7] needed for the circuit to operate at the faster clock speed. Several papers have been published, extending the work in [4] to handle level-clocked circuits, retiming for low power and even peripheral retiming for pipelined circuits[8].

However, to our knowledge, there is little published work[3][7] that deals with reordering and scheduling of signal arrival times at the input boundary of LBs such that both the circuit latency and the number of sequential elements are minimized. This paper explores this relationship in the context of Wave Steered circuits. While not retiming in the classical sense, our goal is to find a minarea solution for a Wave Steered circuit, operating at a user specified clock period such that the global circuit latency is minimum. We show that an *MLO* and scheduling of signals based on their arrival times at LB inputs achieves the minimum circuit latency, and some *MLOs* give the minimum latch count.

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA

3. Wave Steering

Wave Steering operates on synthesized structures that are variants of Binary Decision Diagrams (BDDs)[1]. Binary Decision Diagrams (BDDs) are graph representations of Boolean functions that are widely used in many fields of Computer Aided Design (CAD). For a complete description of BDDs, we refer the reader to [1]. A physical implementation of a BDD is exclusively made of just one type of 2:1 multiplexer node, which can be Pass-Transistor-Logic(PTL) mapped using 2 nfets[2]. However a straightforward PTL mapped BDD implementation has too much delay to be acceptable without some kind of pipelining. Wave Steering offers an attractive solution to drastically decrease this delay since the finest pipeline granularity can be obtained if a single level, controlled by a single variable, is allowed to be a pipeline stage. In such a case, the inputs of a LB will be spatially distributed along the pipeline stages. Since the whole idea of pipelining is to allow for the co-existence of multiple data waves corresponding to different input vectors, we have to skew input variables in application time to allow for pipelined operation.

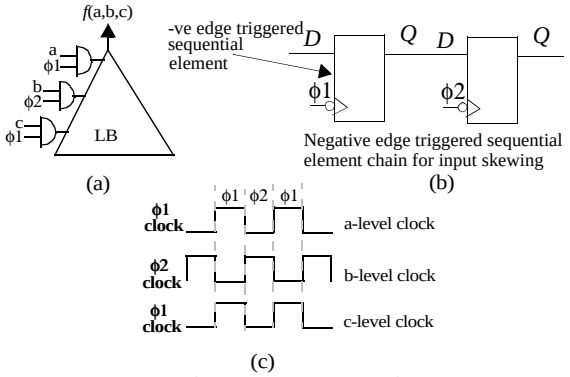


Figure 1: Wave Steering

Figure 1.a shows a Wave Steered LB, that implements a function *f* of 3 variables a,b,c. Each variable is applied at a particular level in the LB, and this level ordering is determined by the arrival times of the variables. If all variables arrive at the same time, then latches are used to skew the variables at the LB input points(Figure 1.b). Each latch is a negative edge triggered sequential element as shown in Figure 1.b. In this paper, our aim is to reorder and schedule signals in such a way to allow for maximum latch sharing. In figure 1.a, the variable ordering is c-b-a, because c is applied at the lowest level (with respect to the output), followed by b and then a. As shown in figure 1.c, we use a 2-phase clocking scheme with a 'ϕ1 clock' and its non-overlapping complement, the 'ϕ2 clock'. The ϕ1 clock is "on" during the ϕ1 phase and "off" during the ϕ2 phase and vice-versa. Variables a and c are clocked by the ϕ1 clock, and variable b by the ϕ2 clock, such that during the ϕ1 phase the a-level and c-level are computing while the b-level is isolating the computations carried out in the other 2 levels, and during the ϕ2 phase the b-level is computing while the a-level and the c-levels are shut off. We denote the period of the 2-phase non-overlapping clock pulsing the circuit by ϕ, and thus each level of the Wave Steered circuit can be clocked in a single phase of duration $\phi/2$. This allows for multiple data waves to co-exist in the LB at the same time since the inputs are spatially spread along the pipeline stages. Since each stage is a single level of multiplexer cells in the structure, the pipelining granularity is very small. We need no latches at the outputs of the multiplexer cells explicitly to hold the signal values constant, since the dynamic node output capacitances of the multiplexer cells act as natural capacitors. In general, if there are *n* levels (corresponding to *n* input variables) in the structure, there will be *n* pipeline stages, and $\left\lceil \frac{n}{2} \right\rceil$ data waves can co-exist inside such a Wave Steered design thereby allowing for high throughput functioning of the circuit. A given circuit can be decomposed into a network of LBs so that the circuit functionality is retained without compromising the clock period. In this paper we do not deal with circuit decomposition issues, but concentrate on optimization techniques for latency and latch count minimization, assuming that we have a decomposition and placement given to us. The next section formally defines the latency and latch minimization problem and defines some key terms used in the rest of the paper.

4. Key terms and Problem Formulation

Figure 2 shows a small network of LBs connected using pipeline latches such that the circuit operates at the required clock period ϕ. For simplicity's sake, in this figure all LBs are of height *h* = 4. Outputs from LBs *i, k, p*, and *m* are inputs to LB *j*. Let $V=\{v_{ij}\}$

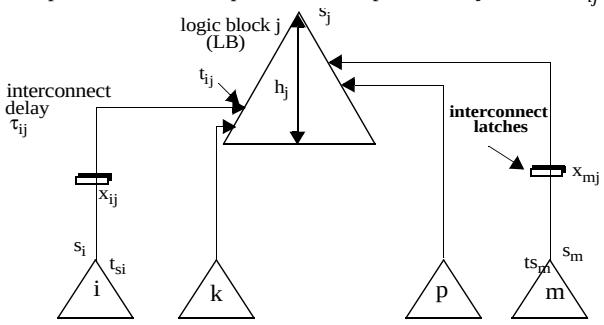


Figure 2: Terminology

be the support set of the function f implemented by the LB *j* where *i* is the source and *j* is the sink LB. $S=\{s_i\}$ is the set of output signals of the fanin LBs of *j*. In this example, $S_j=\{s_i,s_k,s_p,s_m\}$. t_{si} denotes the arrival time of any signal s_i at the output of its corresponding LB. We use τ_{ij} to denote the interconnect propagation delay of a signal s_i to the corresponding LB *j*'s input. If the propagation delay of a routed signal is $\tau_{ij}>\phi/2$ then interconnect latches are required to route the signal s_i to the input of the LB *j*. These interconnect latches are called *hard latches*. In the case where $\tau_{ij}<\phi/2$, latches may be needed to skew the input signals (Figure 1). These latches are called *soft latches*. x_{ij} denotes the total number of *hard* and *soft* latches in the path from signal s_i to an input of an LB *j*. As shown in Figure 2, t_{ij} is the arrival time of a signal s_i at the input of the LB *j*, given by $t_{ij} = t_{si} + \tau_{ij}$. In the network of placed LBs, all routing is assumed to be without congestion and allows for maximum latch sharing between source and sinks. All latches are distributed on the *stem* of the net, which is the wiring between the net source and its one sink requiring the maximum number of latches. Other sinks requiring lesser number of latches tap-off at

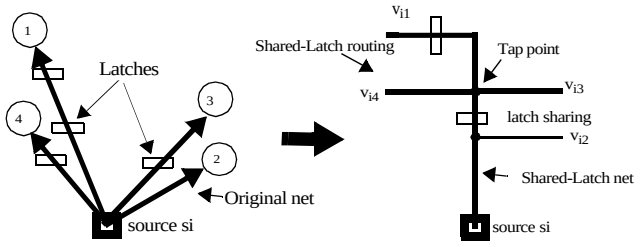


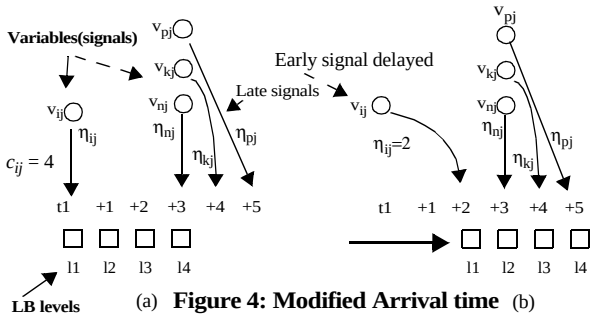
Figure 3: Shared-Latch Routing Model

different points on the stem. The placement ensures that all sinks are clustered together, and all *hard* latches, if any, are shared by all the sinks and are inserted on the net *stem*. This is the *Shared-Latch* routing topology (Figure 3).
We now give a formal problem statement:

Problem: *Given a decomposed and placed Wave Steered netlist having Shared-Latch net topology, find a variable ordering of input signals for a fixed clock period f, such that the circuit latency is minimized (L_{min}) and the number of latches for L_{min} is optimally reduced (ff_{min}).*

5. Variable Ordering and Retiming for Latch Count Minimization

In this section, we outline an efficient method of finding a variable ordering inside each LB in a decomposed and placed Wave Steered netlist, such that the overall latency of the netlist is minimized. Since a Wave Steered circuit is designed for a given clock period, this method gives the *MLO* of variables and LBs in the netlist for a given frequency of operation. For simplicity's sake, we assume that: *(i)* all LBs are complete trees, (ii) all routing between any two points is Shared-Latch routing and allows for maximum latch sharing between different segments of the net(Figure 3).



(a) **Figure 4: Modified Arrival time** (b)

In order to find L_{min}, we derive the latency bound of an individual LB in terms of its t_{ij}. For all LBs, we order the input signals in increasing order of their t_{ij}. Figure 4 shows an ordered list of input signals to an LB. In this figure, we have a situation where the arriving signals introduce a stall in the LB. This means that there is at least one signal that arrives more than $\phi/2$ after the previous signal thereby increasing the LB latency. In order to tackle this situation, we move the early arriving signals forward (in Figure 4, signal 1) in time by η_{ij} half cycles (Figure 4.b) by inserting soft latches so that all arriving signals are in consecutive levels in the LB. The procedures for calculating η_{ij} of signals are shown below.

procedure 1: derive latest modified arrival time

1 **sort** (set of input signals {v_{ij}} of LB) according to *arrival times* in *increasing order*;
2 *current_time = -1;*
3 for every input signal v_i of LB in the sorted list {
4 *current_time* = *current_time* + 1;
5 *current_time* = maximum (*current_time*, t_{ij}); /* t_{ij} = arrival time of v_{ij} */
6} *latest_modified_arrival_time = current_time;*

procedure 2: derive all modified arrival times

1 *current_time = latest_modified_arrival_time+1*; /* from procedure 1*/
2 **sort** (set of input signals {v_{ij}} of LB) according to *arrival times* in *decreasing order*;
3 for every input signal v_{ij} of LB in the sorted list {
4 *current_time* = *current_time* - 1;
5 *current_time* = maximum (*current_time*, t_{ij});
6 η_{ij} = (*current_time* - t_{ij});
7}

Procedure 1 schedules the LB signals in increasing order of arrival times and finds the latest modified arrival time for all LBs in the network of LBs. The latest modified arrival times are then used by procedure 2 to find the modified arrival times of inputs of all LBs in the network, such that all input signals of any LB are scheduled in increasing order of their arrival times in consecutive LB levels. We denote the modified arrival time of a signal by $t_{ij} + \eta_{ij}$. In the case where 2 or more signals have the same arrival time, they are randomly assigned consecutive levels. This is an MLO and leads to the Wave Steering without stalling condition. This condition states that if two variables v_{ij} and v_{kj} arrive at different levels in a LB j that are d_{ik} half cycles apart, then

$$\delta_{ik}\frac{\phi}{2} = \left(ts_i + x_{ij}\frac{\phi}{2}\right) - \left(ts_k + x_{kj}\frac{\phi}{2}\right)$$

Let 'c_{ij}' denote the number of levels below the output of the LB where the variable v_{ij} is applied at its modified arrival time ($t_{ij} + \eta_{ij}$). Then after the application of this signal, it will take c_{ij} half cycles ($c_{ij}.\phi/2$ units of time) for the current vector computation to be completed. Thus the output evaluation time of the LB, t_{sj}, is given by

$$t_{sj} = \left\{(t_{ij} + h_{ij} + c_{ij})\frac{f}{2}\right\} \qquad \textbf{(1)}$$

This formula holds for the output evaluation time of any LB for any and all inputs of that particular LB.

All paths to each primary output in a Wave Steered circuit have equal delay. To derive the global lower bound on the input-output circuit latency L_{min}, we order the variables in the LBs in increasing order of arrival times and schedule them using (1) in a topological sweep from the inputs to the outputs. This is the *MLO* scheduling. The aim is to reorder variables such that each LB delay, and hence the overall circuit latency is minimized. However, this variable ordering is not unique - there may exist other variable orderings, including other *MLO*s for which the circuit latency remains L_{min}. The set of variable orderings that achieve the circuit latency of L_{min}, is denoted by *{VO_{min}}*. We have the following lemmas and theorems with these assumptions: a) Wave-Steering without stalling condition holds, b) all nets have Shared-Latch routing topology, and c) each level in an LB can be assigned only one variable.

Lemma 1

Let t_1^A be the output evaluation time for any LB A under an *MLO* VO_{min}. Let VO be any other ordering that differs from VO_{min} only in the ordering of the inputs to A, and t_2^A be the output evaluation time of A for VO for the same input arrival times. Then $\left(t_2^A \geq t_1^A\right)$ always holds.

Proof

Any variable ordering VO differs from VO_{min} only in the ordering of inputs inside LB A, but the individual input arrival times remain unchanged. Thus the list of inputs sorted by their arrival times (as was the case for VO_{min}) might not remain sorted for VO. Thus a late arriving input may be ordered before an earlier arriving input in the list. Since the ordering determines the output evaluation time of LB A according to (1), for the ordering VO, the LB evaluation time can either remain same as that determined by VO_{min} or

increase, but cannot be smaller than the one determined by VO_{min}. Thus ($t_2^A \geq t_1^A$) holds.

Lemma 2

Consider an *MLO*, VO_{min}. Let $\{t_b^A\}$ be the output evaluation times for all the LBs under VO_{min}. Let VO be a *non-MLO* and $\{t_b^B\}$ be the output ready times for VO. Then $\forall b(t_b^B \geq t_b^A)$.

Proof

Let us assume without any loss of generality, that the arrival times of all primary inputs for the two orderings are same and remain unchanged. Starting from the primary inputs (logic level 0), we levelize the circuit such that the logic level of any LB is 1 more than that of its input at the highest level. This process continues till the primary outputs are reached. For level 0 LBs having all primary inputs, for any variable ordering VO not equal to VO_{min}, the output evaluation time either remains the same or increases according to Lemma 1 and (1). Now consider an LB A at a higher level which has at least one fanin LB. If the arrival times of all the input variables to A increase, then it is easy to see that the output evaluation time of the LB will increase whatever may be the ordering inside the LB. Similarly if the arrival times of all the input variables remain the same for the new ordering VO, any ordering of variables within the LB A which is not *MLO*, can only increase the output evaluation time of this LB (Lemma 1 and formula (1)). Now consider the case where the arrival times of some inputs of A increase (procedures 1 and 2). If the arrival time of an element *i* in a particular position in the sorted arrival time list for VO, is later than that of an element in the same position of the list for the ordering VO_{min}, then the output evaluation time of the LB can only increase. Even if the i^{th} variable's arrival time does not increase from its value for VO_{min} in the new ordering, due to the increase of the arrival time(s) of any previous variable(s) in the list, the LBs output evaluation time will increase. Hence at the end of the evaluation for all LBs, the latest arrival time of all the LBs will either stay the same for the ordering VO as it was for VO_{min}, or increase; however it can never decrease.

Lemma 3

The c_{ij} values for all variables in the supports of all LBs sum up to a constant value whatever may be the variable ordering.

Proof

For an LB with *n* inputs (v_{1j}, v_{2j}, v_{3j}, ..., v_{nj}), the corresponding c_{ij} values are (1,2,3,...n). Thus, whatever may be the variable ordering in any LB with *n* inputs, $\sum_i c_{ij} = \frac{n}{2}(n+1)$,which is a constant for a given decomposition. Since the number of LBs also remain unchanged for different reorderings, the summations of c_{ij} values over all LB inputs is a constant.

We now state the following theorems based on Lemmas 1, 2 and 3.

Theorem 1

L_{min} determined by applying the *MLO* in the LBs is a global achievable minimum latency for the given decomposition and placement of the Wave Steered netlist.

Proof

The proof follows from Lemma 1 and Lemma 2. The latency of a circuit is determined by the output evaluation times of its LBs at the highest logic level, the outputs of which are the primary outputs of the circuit. Since the previous lemmas state that the output evaluation times of LBs can never be improved from that achieved by an *MLO* by any other variable ordering, L_{min} as obtained by *MLO* of variables (and hence LBs) is the global achievable minimum latency for the decomposed and placed Wave Steered circuit.

Theorem 2

The latch count in a Wave Steered circuit is minimized by an *MLO* for a Shared-Latch routing topology.

Proof

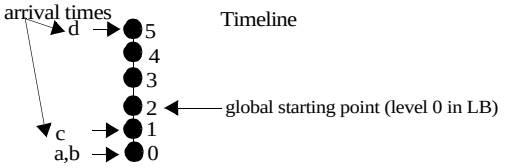
Each path in a Wave Steered circuit from a primary input to a primary output consists of alternating wires and LBs. We introduce a graph representation of a Wave Steered circuit wherein all LB inputs and outputs, including primary inputs and primary outputs are represented by nodes. There are edges connecting the outputs of LBs to the corresponding fanout LB inputs and those that connect the inputs of LBs to their outputs. Let us denote the former edges as *wire* edges. The latter edges will be called *virtual* edges. The delay associated with a *virtual* edge will be the 'c_{ij}' value of the corresponding variable, and that with a *wire* edge will be the number of latches on the interconnect. Thus a path will have alternating *wire* and *virtual* edges. Whatever may be the variable ordering, the graph representations of a Wave Steered circuit will always be isomorphic, and all paths in a given representation will have the same delay as obtained by summing up their *wire* and *virtual* edge delays.

According to Lemma 2, for any *non-MLO*, the output evaluation times of the LBs in a Wave Steered circuit do not decrease from those obtained by an *MLO* schedule. Let Δ be the increase in the arrival time of a primary output of the circuit for a *non-MLO* schedule, over the *MLO* one. Since the delays of all paths to any of the primary outputs are equal, this means that delays of all paths to this primary output increase by a similar amount of Δ. This increase of Δ in the path delays can occur due to two factors: *(i)* an increase in the delay of virtual edges and *(ii)* an increase in the delay of wire edges. However, the summation of the delays contributed by the *virtual* edges in the circuit remains constant for any variable ordering (*Lemma 3*). This implies that the Δ increases in the path delays are contributed solely by increases in the delay of wire edges in all the paths. This translates into an increase in the total number of latches in the circuit. Hence, we can say that any *non-MLO* cannot reduce the latch count from that obtained by an *MLO*.

Note: Not all MLO's may achieve the minimum number of latches in the Wave Steered circuit**.**

6. ILP Formulation for Latency-Latch Optimal Variable Ordering

Theorems 1 and 2 form the basis of solving the problem stated in Section 4. Starting with an initial *MLO* that achieves L_{min} but not neccesarily the minimum number of latches, we group all LB input variables according to their arrival times, and find tight



[a,b,c] can be reordered amongst themselves starting at position 2
[d] cannot be reordered.
[a,b,c] = [2,4] where 2 is the Lower position bound and 4 is the Upper position bound.
[d] = [5,5]

Figure 5: Upper Lower Bounds

Upper and *Lower* bounds of application times of variables in these groups, such that the LB output evaluation times remain unchanged. Figure 5 shows an example. In this example, signals a and b arrive at time 0, signal c arrives at time 1 while signal d arrives at time 5. Since the variables have to be scheduled in consecutive levels to satisfy the Wave Steering without stalling condition, signals a,b and c are delayed (d cannot be scheduled earlier than it arrives) such that they occupy positions 2,3 and 4, though not in any particular order. Now the problem becomes one of reordering a,b and c. We derive an *Upper(Up)* and *Lower(Low)* application time bound on each group of variables. Hence [a,b,c] = [2,4] where time 2 is the earliest that one of these variables can be applied and time 4 is the latest that one of them can be applied, without changing the latency. [d] = [5,5] since it has no flexibility of reordering. In forming the ILP constraints, we note that the total number of latches in the net s_i with sinks in {j} is given by:

$$y_i = \textbf{\textit{Maximum}}(x_{ij}) \qquad (2)$$

where x_{ij} is the number of latches between any source *i* - sink *j* pair in a net. We define w_{ij} as an integer label associated with each variable v_{ij} (where *i* is the source of a net and *j* is one of the sink LBs) which represents the relative change in v_{ij}'s position after reordering. Each variable in a LB has one such w_{ij} label and depends on the change in level from some previous ordering. Figure 6 shows the different w_{ij} values for the example in Figure 5 can take. The

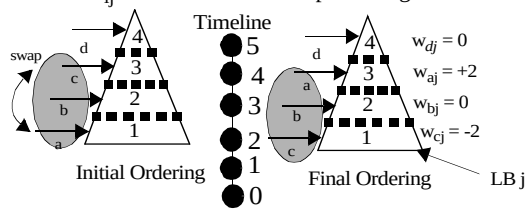


Figure 6: w_{ij} calculation

value of this label can be negative (reordered to a lower level), positive (reordered to a higher level) or zero (no change in level after reordering). All w_{ij}'s are bounded by (Low_{ij} - p_{ij}) and (Up_{ij} - p_{ij}) where p_{ij} is the initial modified arrival time of v_{ij} in a LB. Note that the modified arrival time of a variable v_{ij} is given by ($t_{ij}+\eta_{ij}$). Since the *Up* and *Low* bounds use the global circuit timeline as a reference point, we need to subtract the p_{ij} values from these bounds to make the w_{ij} labels relative to the individual LB's timeline. The procedure for finding these tight *Up* and *Low* bounds is shown in Figure 7.

Hence, our ILP is: *Minimize* $\quad F = \sum_i y_i$

subject to:

$$\forall LB \forall w_{ij} (Low_{ij} - p_{ij}) \le w_{ij} \le (Up_{ij} - p_{ij}) \qquad (i)$$

$$\forall LB (w_{ij} + c_{ij} \ne w_{kj} + c_{kj}) \qquad (ii)$$

$$\forall i \left[\forall j \left[y_i \ge \left(\left\lfloor \frac{w_{ij}}{2} \right\rfloor + x_{ij} \right) \right] \right] \qquad (iii)$$

The objective function aims to minimize the total number of latches in the Wave Steered circuit. The first constraint imposes strict upper and lower bounds on each variable in all LBs - this essentially keeps L_{min} from changing and ensures that the final ordering is indeed an MLO. The second constraint says that each variable can only be assigned a single level (i.e. two or more variables cannot collapse in the same level). The third constraint gives the relationship between the number of latches on a net segment after reordering, the w_{ij}'s of the sink LB and the original number of latches on this particular segment. Note that since we use a 2-phase negative edge triggered clocking scheme, an increase by 2 levels of the application of a variable v_{ij} only increases the number of latches by 1. Constraint (*iii*) says that total latches on a net *i* is greater than or equal to the total latches on a particular source-sink net segment after reordering. This is true in our Shared-Latch net topology. During our ILP formulation, we ignore those variables that have equal *Up* and *Low* bounds since they have no reordering flexibility. This condition reduces the complexity of our ILP.

procedure 3: derive variable application times

```
1  call procedures 1 and 2;
2  global-time-pointer = latest_modified_arrival_time;
3  group inputs with same arrival times;
4  L = list of input groups sorted in decreasing order of arrival times;
5  k = 1; /* starting with the first element of L */
6  while L_k in L {
7     iList = { ∅ };
8     do {
9        iList = iList ∪ k ;
10       if (k < |L|) {k = k + 1;}
11       else {break;}
12    } while ((global-time-pointer - t_{k-1} - Σ_{m ∈ ilist} |Lm| + 1) > 0)
13    for every k in iList {
14       for every v_ij in L_k {
15          lower-bound of v_ij = max(t_ij, (global-time-pointer -
          Σ_{m ∈ ilist} |Lm| + 1));
16          upper-bound of v_ij = global-time-pointer;
17       }
18    }
19    maxk = max(iList);
20    global-time-pointer = t_{maxk} - 1;
21 }
```

Figure 7: Procedure Derive Bounds

Theorem 3

The latch reduction obtained by restricting the variable application times between the specific *Up* and *Low* bounds is not inferior to one that considers unrestricted variable application times, in an MLO scheduled circuit with Shared-Latch synthesized nets.

Proof

For any LB, certain variables may have overlapping application time intervals as found using procedure 3. Let us denote these variables as $[v_i]$. If any reordering inserts a variable v_j in $[v_i]$ such that its application time interval does not overlap with those of other variables in $[v_i]$, then it means that either *(i)* the arrival time t_j of v_j is later than the latest possible application times of all the other variables that were there in $[v_i]$, or *(ii)* the latest valid application time of v_j is smaller than the earliest arriving variable that was already there in $[v_i]$. In either case a late arriving input will be ordered before an earlier arriving one and we need to re-evaluate the output evaluation time of the LB using procedure 1. However, the output evaluation time of the LB under the new ordering can only increase from that which can be achieved if the list in the procedure 1 was to remain sorted. According to Theorem 2, such an ordering would end up increasing the latch count in the circuit, and hence it would not be considered by any optimizing method. Thus the optimum latch count solution lies among the orderings that are done within mutually exclusive groups of inputs having overlapping application time intervals.

7. Experimental Results

We tested our ILP formulation on two kinds of benchmarks--MCNC benchmarks(with no feedback loops) and 2 array multipliers. All circuits were decomposed into LBs of heights of up to 8 levels and placed using a commercial placer. We then calculated

Table 1: Experimental Results

Circuit	LBs	Random	MLO	NoSLS	ff_{min}	L_{min}(ns)	SC
C6288	449	1844	1605	1939	1058	50	1.45
C3540	398	1444	1196	1024	668	28	2.39
C880	158	398	313	281	273	20	2.49
C1908	123	525	421	451	326	24	2.47
C7552	604	4471	3827	3885	3375	36	3.87
C2670	286	3388	3218	3172	3073	28	4.0
C432	119	283	181	157	131	20	2.32
alu2	355	731	455	339	293	26	3.42
alu4	584	1387	858	681	583	46	4.5
dalu	254	871	714	757	539	20	2.28
8bm	71	31	18	11	11	40	1.2
16bm	271	145	43	32	26	80	1.2
Avg.	**306**	**1293.2**	**1070.7**	**1060.8**	**863.0**	**34.83**	**2.6**

L_{min} for each circuit using formula (1) based on the signal arrival times. For this ordering, we generated and fed the ILP constraints to a commercial ILP solver. In Table 1, we compare our ILP latch count results to those obtained through: 1) a *random* ordering, 2) an *MLO* for the given placement and decomposition and 3) an ordering (*NoSLS)* in which no soft latches are shared on any net. Columns 1 and 2 show the circuits and the number of LBs of varying heights upto level 8. Column 3 shows the number of latches in a random variable ordering. Column 4 shows the latch count from an *MLO*. The next column(*NoSLS*) shows the total latch count obtained from the ILP solver when we do not share any soft latches on a net (i.e. there is special skewing circuitry inside each sink LB). This latch count can be reduced by 18.6% if we use the Shared-Latch topology. The final latch count obtained from the ILP solver for the Shared-Latch topology is in the sixth column(ff_{min}). This latch count is obtained for an *MLO* different from the *MLO* in Column 4. All these numbers are for a clock period of 500 MHz in 0.25µm CMOS technology. The seventh column shows the minimum circuit latency L_{min} for the obtained *MLO*. The final column shows the relative increase in latency in the Wave Steered design style over a commercial *Standard Cell(SC)* library-mapped and timing-optimized circuits. This increase in latency is more a factor of decomposition and placement rather than a weakness in the Wave Steering design methodology. The average throughput for these unpipelined SC circuits is 53.3 MHz which is 10 times slower than the Wave Steered circuits. We observe that an *MLO* is able to reduce the latch count by 17.2% over a random ordering. This is reduced by another 19.3% by using our ILP formulation(column 6). The results also indicate that we achieve throughputs of 500 MHz at the cost of 2.6 times increase in latency over unpipelined non Wave Steered SC circuits. By further increasing the clock period of our Wave Steered circuits (upto 833 MHz in 0.25µm CMOS technology), we can trade off a higher latency for even higher throughput. This ability to choose the clock period is an attractive feature of the Wave Steered methodology.

8. Conclusions and Future Work

We have presented an efficient solution for latency and latch count minimization for Wave Steered circuits and have provided the theoretical bases. We demonstrate the necessary conditions for latency minimization and show that for the same clock period, we can reduce the total latch count by as much as 33.2% over a randomly ordered Wave Steered circuit. A lower latch count, without compromising the circuit throughput, makes it easier to synthesize the clock trees and save the total dynamic power consumption at the same frequency of circuit operation. Any placement tool which ensures that all sink LBs of all nets are placed locally amongst themselves, can benefit from the optimality of our results obtained using the Shared-Latch net topology. Currently, we are looking into topology generation and feed-forward decomposition issues.

9. Acknowledgment

This work was supported in part by the NSF grant CCR 9811528 and in part by California MICRO program through Xilinx.

10. References

[1] R.E. Bryant, "Graph-based Algorithms for Boolean functions manipulation", IEEE Trans. Computers, Vol. C-35, pp. 677-691, Aug. 1986.

[2] P. Buch, A. Narayan, A.R. Newton, A. Sangiovanni-Vincentelli, "Logic Synthesis for Large Pass Transistor Circuits", ICCAD '97, November 1997.

[3] J. Grodstein, E. Lehman, H. Harkness, H. Touati, B. Grundmann, "Optimal latch mapping and retiming within a tree", proc. IEEE/ACM International Conference on Computer-Aided Design, 1994. pp.242-245

[4] C.E. Leiserson, J.B. Saxe, "Retiming Synchronous Circuitry", Algorithmica, (1991) 6:5-35.

[5] L. Macchiarulo, S.M. Shu, M. Marek-Sadowska, "Wave Steered FSMs", proc. DATE 2000, pp.270-276.

[6] L. Macchiarulo, M. Marek-Sadowska, "Wave-Steering One-hot Encoded FSMs", proc. 37th Design Automation Conference, June 2000, pp. 357-360.

[7] N. Maheshwari, S.S. Sapatnekar, "Efficient Retiming of Large Circuits", IEEE Transactions on VLSI, Vol. 6, No.1, pp.74-83, March 1998.

[8] S.Malik et al, "Performance optimization of pipelined logic circuits using peripheral retiming and resynthesis", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.12, (no.5), May 1993. p.568-78

[9] A. Mukherjee, R. Sudhakar, M.Marek-Sadowska, S.I. Long, "Wave Steering in YADDs: A Novel Non-iterative Synthesis and Layout Technique", Proc. Design Automation Conference '99, pp 466-471.

[10] A. Mukherjee, M. Marek-Sadowska, S.I. Long,"Wave Pipelining YADDs- A Feasibility Study", Proc. IEEE Custom Integrated Circuits Conference, '99, pp 559-562.

[11] A. Singh, L. Macchiarulo, A. Mukherjee, M. Marek-Sadowska, "A Novel High-Throughput FPGA Architecture", Eighth ACM International Symposium on FPGAs,pp. 22-27, February 2000.

[12] A. Singh, A. Mukherjee, M. Marek-Sadowska, "Interconnect Pipelining in a Throughput Intensive FPGA Architecture", proc. Ninth ACM International Symposium on FPGAs,pp.153-160, February 2001.