Engineering

Electrical Engineering fields

Okayama University *Year* 2001

A global routing technique for wave-steered design methodology

Nobuo Funabiki
Okayama University

Amit Singh
University of California

Arindam Mukherjee
University of California

Malgorzata Marek Sadowska
University of California

A Global Routing Technique for Wave-Steered Design Methodology

Nobuo Funabiki[1], Amit Singh[2], Arindam Mukherjee[2], Malgorzata Marek-Sadowska[2]
[1]Department of Communication Network Engineering, Okayama University, Japan
[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA

Abstract

Wave-Steering is a new circuit design methodology to realize high throughput circuits by embedding layout friendly structures in silicon. Latches guarantee correct signal arrival times at the inputs of synthesized modules and maintain the high throughput of operation. This paper presents a global routing technique for networks of wave-steered blocks. Latches can be distributed along interconnects. Their number depends on net topologies and signal ordering at the inputs of wave steered blocks. Here, we route nets using Steiner tree heuristics and determine signal ordering and latch positions on interconnect. The problem of total latch number minimization is solved using SAT formulation. Experimental results on benchmark circuits show the efficiency of our technique. We achieve on average a 40% latch reduction at minimum latency over un-optimized circuits operating at 250 MHz in 0.25μm CMOS technology.

1. Introduction

Wave-steering is a layout-friendly design methodology to realize high throughput circuits by embedding synthesized modules in silicon. These synthesized modules are variants of Binary Decision Diagrams (BDDs) [2]. Since the first work in [4], the wave-steering design methodology has been extended to ASIC structures [5], reprogrammable FPGAs [6][7], and finite state machines [8][9]. In each instance, throughput has been improved as much as 3-4 times over non-wave-steered circuits. A circuit composed of wave-steered blocks inherently utilizes latches to guarantee correct arrival times at their inputs. Latches are used for signal skewing and for high frequency operations on the interconnects between blocks. In this paper we present a global routing technique for wave-steered circuits that determines routing topologies using rectilinear Steiner trees. Next we find an optimal assignment of signals to block levels and latch positions on interconnects for maximal latch reduction.

The ability to guarantee by construction a specified system clock period is an attractive feature of the wave-steering design methodology. To achieve the operation with a specified clock frequency we utilize a variant of pipelining within the blocks and on interconnects. Unlike classical micro-pipelining schemes, this system does not introduce logic redundancy.

Our global routing and latch assignment technique are performed in four steps. In the first step, the rectilinear routing path for each net is found by using the *1-Steiner Tree* algorithm [9]. In the second step, latches are assigned on interconnects to satisfy delay constraints. In the third step, signal arrival times to logic blocks (LBs) are evaluated for the minimum achievable latency. At that time the lowest and highest LB input levels assignable to each signal are calculated. Finally, in the last stage, signal input levels are determined by formulating this problem as the Satisfiability problem (SAT) and applying an existing SAT solver [11]. To simplify the formulation we do not consider routing resource constraints and/or congestions, or circuits with feedbacks. We also assume that all LBs are complete binary trees, which is the case in Field Programmable Gate Arrays (FPGAs)[6][7]. This assumption prevents any LB deformation due to input assignment.

We organize the rest of the paper as follows: Section 2 briefly introduces the underlying ideas of the Wave-Steering design methodology. We formulate the problem in Section 3. Section 4 describes the four stages of our global routing and latch assignment technique. Section 5 shows experimental results on benchmark circuits. Finally, concluding remarks and future works are provided in Section 6.

2. Wave-Steering design methodology

2.1 Wave-Steered LBs

All LBs in a Wave-Steered circuit are variants of a Binary Decision Diagram (BDD) [2]. BDD's physical implementation is made of one 2:1 multiplexer node type. We use Pass-Transistor-Logic (PTL) mapped nodes built of 2 nfets[1][3]. Because a straightforward PTL mapped BDD implementation has too much delay to be acceptable, the wave-steering methodology introduces a fine pipeline granularity to drastically decrease the delay. A single level, controlled by one signal, becomes a pipeline stage. The inputs to an LB are spatially distributed along the pipeline stages. To allow the co-existence of multiple data waves corresponding to different input vectors in a single design, we must skew input signals properly in application time.

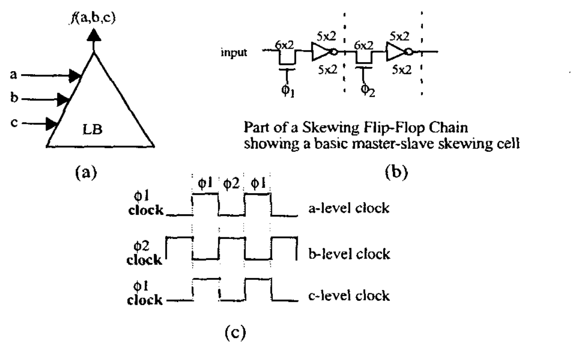

Figure 1: Wave-Steering principle

Figure 1.a shows an LB in wave-steering methodology, to implement a function *f* of 3 signals *a*, *b*, *c*. Each variable must be assigned to a particular level in the LB considering their arrival times. If several signals arrive at the same time, dynamic latches are used to skew them (Figure 1.b) [4][5][6]. Figure 1.a shows that *c* is assigned to the lowest level, followed by *b* and *a*. Figure 1.b shows a 2-phase clocking scheme with a '$\phi 1$ clock' and its non-overlapping complement, the '$\phi 2$ clock'. Both clocks have a $\phi 1$ phase followed by a $\phi 2$ phase. The $\phi 1$ clock is "on" during the $\phi 1$ phase and "off" during the $\phi 2$ phase and vice-versa. Signals *a* and *c* are clocked by the $\phi 1$ clock, and *b* is by the $\phi 2$ clock. During the $\phi 1$ phase the *a*-level and *c*-level are computing while the *b*-level is isolating the computations, and during the $\phi 2$ phase, the *b*-level is computing while the other levels are shut off. Because the clocking period is denoted by ϕ, each level of an LB can be clocked in a single phase of duration $\phi/2$. As a result, since the inputs are spatially spread along the pipeline stages, multiple data waves coexist in the LB at the same time. Since each stage is a single level of multiplexer cells, the pipelining granularity is very small. The dynamic node output capacitances of the multiplexer cells act as natural capacitors; therefore no latches are needed at the outputs of the multiplexer cells to explicitly hold the signal values constant. In general, if there are *n* levels (corresponding to n input signals) in an LB, there will be *n* pipeline stages, and $\left\lceil \frac{n}{2} \right\rceil$ data waves co-exist inside this LB.

2.2 Network of Wave-Steered LBs

A larger circuit may be built from a network of wave-steered blocks. An example of a small network is shown in figure 2. Each LB may have an arbitrary number of levels. In the wave-steering methodology, the efficient latch assignment to interconnects becomes an essential task not only to guarantee the synchronized operation of the circuit with the high frequency, but also to reduce the cost of hardware and power consumption arising from latch requirements.

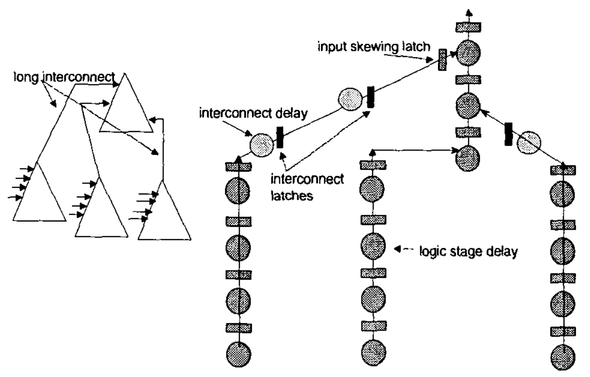

Figure 2: A network of Wave-Steered LBs

Depending on the role a latch plays, it is classified as either a *hard latch (interconnect latch)* or a *soft latch (input skewing latch)*. Hard latches fragment interconnects into fixed delay segments. They must be assigned such that the signal propagation time between two consecutive latches on the interconnect is less than the half of the user-specified clock period. We refer to this requirement as *interconnect delay constraint*. On the other hand, the soft latches are assigned such that LB can properly operate without a stall. Since the wave-steering methodology utilizes a two-phase non-overlapped clocking scheme [4][5], input signals at two consecutive input levels of an LB have to reach there a half clock phase apart. Thus, the soft latch is introduced on interconnects to satisfy the timing constraint to LB input signals.

Actually, the number of required soft latches at an LB input is equal to the timing difference between the actual arrival time and the requested arrival time of the corresponding signal. The actual arrival time is evaluated by considering the output time from the former LB and the number of hard latches on the interconnect. The requested arrival time is given by the signal output time from the LB, which is uniquely determined on the assumption of the minimum latency for the circuit operation. However, if several sinks of one net are assigned the same number of soft latches, they can be shared by allocating the latches before the branching point. The total number of soft latches is uniquely determined for a given hard latch assignment under the minimum latency operation. Therefore the input signals to LBs should be assigned to proper levels such that as many soft latches as possible can be shared.

Note that in this paper, we assume the circuit decomposition and placement done by other algorithms. The next section formally defines the global routing problem for the wave-steering design methodology.

3. Global routing problem formulation

The LB placement and interconnect net list are given as inputs. The goal of the global routing problem is to find a path for every net with hard/soft latch assignment such that no pipeline hazards exist, and the circuit latency and the total number of assigned latches are minimized. Here, for simplification, we restrict the routing path such that 1) any path is rectilinear, 2) no routing congestion is considered (no limitation on the number of nets passing through the same channel), and 3) the total path length is minimized.

Hard latches should be assigned to routing paths or interconnects such that the interconnect delay between any two consecutive latches is no more than half of the clock period. Because the signal delay time usually depends on the number of loading gates [10], for a net driving k gates, the maximum interconnect length between two consecutive latches is given by a function of k, $\Delta[k]$.

After the hard latch assignment, we can calculate signal arrival times at the LB inputs, the system latency, and the upper/lower LB level limits assignable to each input signal in topological order from primary inputs to primary outputs. Inputs to LBs are assigned such that no two of them share the same level and the output is available as soon as possible (*ASAP*).

The final input signal level assigned is determined such that the number of shared soft latches are maximized (Figure 3). This subproblem is solved by transforming it into a satisfiability problem (SAT).

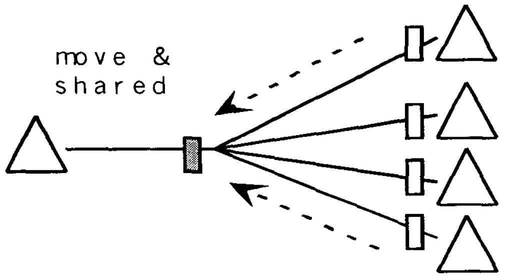

Figure 3: Soft latch sharing

4. Global routing and assignment

The global routing and latch assignment technique is divided into 4 stages. Sections 4.1-4.4 discuss these subproblems in detail. We denote the number of levels, or height, in the j_{th} LB as h_j and its output evaluation time as out_j. An input i to the j_{th} LB has the arrival time $arrival_{ij}$. These notations are used throughout the remainder of the paper.

4.1 Routing path selection

To solve this subproblem, we adopt the "1-Steiner algorithm" by Robins [10]. This algorithm finds a near-optimal rectilinear routing path for a given set of terminal locations by repeatedly adding one Steiner point until no further improvement can be made. In our technique, the routing paths for all nets are found sequentially without considering congestion possibly caused by the already routed nets.

4.2 Hard latch assignment

Hard latches are assigned to each net sequentially. In order to obtain the minimum latency, hard latches are first assigned to each interconnect from the source to the sinks by the procedure **forward assignment**. There, after each interconnect segment has been sorted topologically from the source to the sinks, hard latches are assigned at the most distant locations from the previous latches. Then they are shared as much as possible by moving them backward before net branches by the procedure of **backward sharing** (Figure 4).

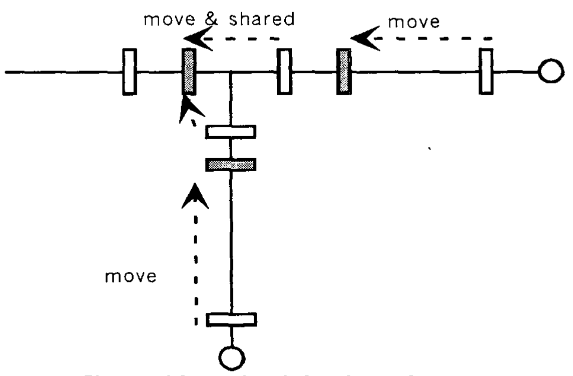

Figure 4 Hard latch backward movement

```
procedure forward_assignment
 1. Initialize the point list on the path as $Q$ = {source}.
 2. If $Q$ is null, terminate the procedure.
 3. Pick up the first point, $z$, in $Q$.
 4. Assign (hard) latches on the path section between $z$ and its first branch or sink with $\Delta[1]$.
 5. If it reaches sink, go to 2.
 6. Assign a latch just before the branch on the path if the distance between the last latch in 4. and the branch is larger than $\Delta[2]$.
 7. Find the most distant point from z such that the latch at z can cover outgoing wires:
    1) Sort branches on outgoing wires from $z$ in ascending order of distance from $z$.
    2) Count the number of outgoing wires if a latch is assigned just after branch node.
    3) Find the largest branch to satisfy distance <= $\Delta[fanout]$.
 8. Assign one latch to each outgoing wire with $\Delta[fanout]$.
 9. Insert points where latches are assigned into $Q$ and go to 2.
```

```
procedure backward_sharing
 1. Initialize the point list on the path as $Q$={sink_1, sink_2,...., sink_m}.
 2. If $Q$ is null, terminate the procedure.
 3. Pick the first point $z$ in $Q$.
 4. Starting from $z$, move latches backward as much as possible until a branch is reached.
 5. If a branch has latches on all of its $k$ outgoing wires placed just after the branch point and each of their neighboring latches in the sink direction are located within $\Delta[k]$ from the branch point, merge those latches which are just after the branch point and move the resulting latch before the branch point.
 6. Insert this merged latch position into $Q$, and go to 2.
```

Property 1: forward_assignment provides the hard latch assignment for the minimum system latency.

Proof: Any hard latch except the last one to a sink covers the longest interconnect section. Thus, any forward shift of a latch creates interconnect sections which are not covered by a hard latch, which violates the delay constraints.

Property 2: backward_sharing may reduce the number of hard latches.

Here, we note that for simplification, **backward_sharing** only considers the hard latch reduction under the condition that the total number of hard latches is not increased on any path from a primary input to a primary output. If we allow a subset of k hard latches on the outgoing wires from one branch to be merged into one latch, the number of hard latches may be further reduced without increasing the system latency. However, in this case, every time hard latches are merged, the increase of the system latency must be checked to avoid it. Thus, **backward_sharing** does not guarantee the minimality on the total number of hard latches.

4.3 Signal arrival time evaluation

Knowing the hard latch assignment, we calculate for each net the signal arrival times at its sinks, and the output times from its source. The nets are sorted topologically from primary inputs to primary outputs. The signal arrival time at a sink is determined by adding up the signal output time from the net's origin and the number of hard latches assigned on the interconnect path. When arrival times of all inputs of an LB are calculated, the LB's output time is determined.

```
procedure signal_arrival_time
 a. Calculate signal arrival times at the net sinks (LB inputs) when the net's source arrival time has been already determined. Initially, LB inputs which are net sinks originating at primary inputs are calculated.
    $arrival_{ij} = out_k + num\_of\_latch_{kj}$
 where
    $arrival_{ij}$: arrival time to $i$-$th$ input to $j$-$th$ LB
    $out_k$: output time from k-th fanin LB
    $num\_of\_latch_{kj}$: number of hard latches on the path from k-th LB to j-th LB.
    Here, we assume i-th input to j-th LB has a source at k-th LB.

 b. Calculate signal output times for LBs in which all input arrival times are obtained.
    1) Sort arrival times of LB inputs (assume $m$ inputs) in ascending order: $time_0$, $time_1$, ...., $time_{m-1}$.
    2) Initialize output time by:
       $out_k = time_0$.
    3) Initialize the pointer by p=0.
    4) If $p=m$, terminate the procedure.
    5) Update the output time by:
       $out_k = \mathrm{MAX}(out_k + 1, time_{p+1})$
    6) Increment p by 1, and go to 4).
```

Property 3: signal_arrival_time calculates the earliest possible output time from an LB for given LB input signal arrival times.

Once signal arrival times for all LB inputs are found, we determine the lowest possible and the highest possible input permissible levels for each input signal to a given LB. Within the bounds are all the possible levels to which a particular signal can possibly be assigned without increasing the LB's latency (and hence the circuit latency). Then, the lowest permissible level to each LB input signal is determined from the following equation:

$$low_{ij} = Max(0, h_j - (out_j - arrival_{ij} - 1)) \qquad (1)$$

The highest permissible level is determined by the number of signals which must be assigned to higher levels in the LB. The procedure to calculate the highest permissible levels of input signals to a particular LB is shown below.

```
procedure highest_assignable_level
1. Sort input signals to the LB in descending order of $low_{ij}$.
2. Initialize highest permissible level, $high_{ij}=h_j$.
3. Initialize pointer $p=h_j$, and the number of signals to be
   assigned in higher levels by $num\_signal=0$.
4. Update $num\_signal$ by adding the number of signals where
   $low_{ij} = p$.
5. If $num\_signal = high_{ij} - p + 1$ (current assignable space is
   occupied by such signals), update $high_{ij}$ of signals where
   $low_{ij} <= p$ by $high_{ij}=p-1$, and initialize $num\_signal=0$.
6. If $p=2$, terminate the procedure, else decrement $p$ by $1$ and go
   to 4.
```

4.4 Input level assignment

The LB input level assignment subproblem is formulated as a variant of the satisfiability problem (SAT) [11], and is solved using an existing SAT solver. The goal of the SAT instance is to find a variable assignment such that every clause becomes true. A *clause* is a union of *literals* which are variables or their negations. First, we define a variable:

$x_{ij}^{k} = 1$: the *ith* input of the *jth* LB be assigned to the *kth* level.

$x_{ij}^{k} = 0$: not assigned there. (2)

We construct clauses to represent the constraints and the objective function of this subproblem. The first constraint is to assign every LB input signal to one of its permissible levels. The clause corresponding to the i-th signal of the j-th LB is described by:

$$\prod_j \prod_i \sum_{k = low_{ij}}^{high_{ij}} x_{ij}^{k} \qquad (3)$$

where Σ represents a logical *OR* function.

The second constraint says that no two signals can be assigned the same level. The clause is written in the conjunctive normal form (CNF) as:

$$\prod_j \prod_{i, n \in J} \prod_{k = low_j}^{high_j} (\bar{x}_{ij}^{k} \vee \bar{x}_{nj}^{k}), i \neq n \qquad (4)$$

where Π represents a logical *AND* operation and J is the set of all inputs to LB j.

The objective of this subproblem is to maximize the number of soft latches which can be shared by different LB input signals on the same net. If each of b outgoing signals from one branch of a net is assigned p soft latches at LB inputs, these p soft latches are moved backward before the branch and are shared by all of them. As a result, a total of $p(b-1)$ soft latches can be removed.

Now, we consider the i-th signal to the j-th LB. If it is assigned to the kth level, the number of soft latches assigned there is given by:

$$soft_{ij}^{k} = out_j - (h_j - k) - arrival_{ij} \qquad (5)$$

In order to move p soft latches, this number must be p or larger:

$$soft_{ij}^{k} \geq p \qquad (6)$$

By solving these equations, the lowest level with p soft latch assignment is obtained:

$$lowp_{ij} = p - out_j + h_j + arrival_{ij} \qquad (7)$$

Thus, we describe as follows the set of clauses to represent the condition that every sink signal departing from one branch is assigned p soft latches:

$$\sum_{lowp_{ij}}^{high_{ij}} x_{ij}^{k} \wedge \sum_{lowp_{lm}}^{high_{lm}} x_{lm}^{k} \wedge \ldots \wedge \sum_{lowp_{st}}^{high_{st}} x_{st}^{k} \qquad (8)$$

where it is assumed that the i-th signal of the j-th LB, the l-th signal of the m-th LB, and the t-th signal of the s-th LB come from the same branch of a net. In the example in Figure 5, the set of objective clauses for $p=1$ is given by:

$$(x_{11}^{3} \vee x_{11}^{4}) \wedge (x_{12}^{1} \vee x_{12}^{2} \vee x_{12}^{3}) \wedge (x_{13}^{2} \vee x_{13}^{3}) \qquad (9)$$

Note that some of such objective clauses may not be satisfiable simultaneously in conventional designs. In such situations, we adopt a heuristic approach selecting satisfiable objective clauses sequentially in the descending order of the number of soft latches shared.

```
procedure SAT_optimization
1. Compose a SAT instance by including the clauses for two con
   straints.
2. Sort the sets of objective clauses in descending order of the
   number of reduced soft latches if they are satisfied.
3. If an unprocessed set of objective clauses remains, add the first
   such set into the SAT instance, else terminate the procedure.
4. Solve the SAT instance by a SAT solver.
5. If this SAT instance is not solved, remove the added set of
   objective clauses.
6. Go to 3.
```

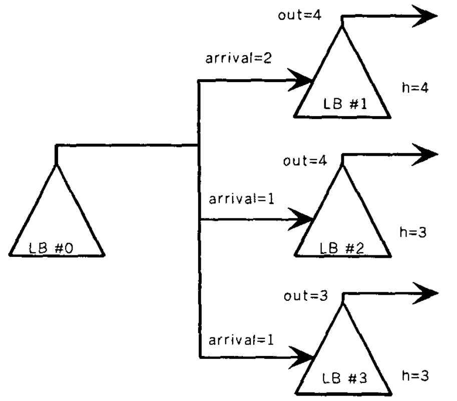

Figure 5: Example-Soft latch sharing

Property 4: Any feasible level assignment of LB input signals requires the same number of soft latches if no latch sharing is done.

Proof: The number of soft latches for the j-th LB is given by: $soft_j = \sum_{i=0}^{h_j-1} (h_j - out_j + arrival_{ij} - level_{ij})$, where $level_{ij}$ is the assigned level to the i-th signal of the j-th LB. Because h_j, out_j, $arrival_{ij}$, and $\sum_{i=0}^{h_j-1} level_{ij} = h_j\frac{(h_j-1)}{2}$ are all constants, $soft_j$ is constant regardless of the assignment $level_{ij}$.

5. Experimental results

We tested our global routing technique on two types of benchmarks, MCNC benchmarks (with no feedback loops) and 2 array multipliers. We considered two different circuit decompositions so that we can see how the decomposition and the related interconnect delay affect the routing results. In Case 1, the heights of LBs do not exceed 4 levels. The resistance and capacitance of a wire between two neighboring LBs is 11.67 Ω and 35.67pF respectively (assuming all LBs have aspect ratio 1 and dimensions of 100µm x 100µm in 0.25µm CMOS). In Case 2, the height is up to 8 levels and the delay parameters are 19.50 Ω and 65.00pF per LB span respectively. All delay parameters are obtained based on a 0.25µm TSMC technology. The clock frequency is fixed at 250 MHz. The LBs are placed using a commercial placer, where any LB is placed on a grid point.

Tables 1 and 2 show experimental results for these circuits for each case. The tables list each circuit's name (*Circuit*), the number of nets (*Nets*), the maximum number of latches on a path from a primary input to a primary output (*Latency*), the total number of hard latches (*Hard_L*), the total number of soft latches if they are not shared (*Soft_L*), the reduced number obtained by sharing (*Reduction*), and its percentage (%) of the total.

When we compare the required numbers of hard and soft latches in Case 1 and 2, Case 1 (=2359.4 on average) provides better results than Case 2 (=3775.0). Case 1 also gives better average latency. While this may appear counter-intuitive, this difference can be explained by the fact that in Case 1, the decomposition is such that nearly all LBs are of equal heights (h = 4). This results in smaller interconnect delays between LBs, and fewer latches are required to balance interconnect delays between LBs of different heights. A better decomposition for Case 2 will lead to substantially better latency results.

Table 1: Experimental results (4 Levels)

Circuit	Nets	Latency	Hard_L	Soft_L	Reduction	%
C6288	481	51	63	13324	7163	53.76
C3540	448	31	215	3969	1513	38.12
C880	218	18	64	1275	594	46.59
C1908	156	28	123	2002	544	27.17
C7552	919	27	524	4991	2129	42.66
C2670	519	28	184	1878	945	50.32
C432	155	28	43	865	128	14.80
C499	115	15	32	607	205	33.77
alu2	365	27	80	1892	461	24.37
alu4	598	31	215	3969	1072	27.01
dalu	345	29	201	6167	1966	31.88
8bm	87	28	50	900	439	48.78
16bm	303	57	225	9145	5587	61.09
Avg.	**362.2**	**30.6**	**155.3**	**3921.8**	**1717.7**	**38.49**

When we compare the number of hard and soft latches in the tables, the latter number is approximately 25 times greater than the former one for Case 1 and seven times than Case 2 before the soft latch sharing. The reduction of soft latches is critical for the efficient design in the Wave-Steering design methodology. Our technique can reduce the total number of soft latches by about 40% on average.

In order to reduce soft latches further, a proper circuit decomposition is critical. Figure 6 depicts the relationship between the average number of soft latches per sink and the number of sinks per net for "*alu4*" and "*C7552*". This graph suggests that nets with more sinks usually require more soft latches per sink. In such nets, some sinks may request relatively earlier arrival times and some may request later times. Because these soft latches cannot be shared, more soft latches have to be assigned. The decomposition into smaller fanout nets is desired for the wave-steered circuits.

Table 2: Experimental results (8 levels)

Circuit	Nets	Latency	Hard_L	Soft_L	Reduction	%
C6288	481	78	931	13111	7294	55.63
C3540	448	45	785	6355	3605	56.73
C880	218	33	252	2775	1165	41.98
C1908	156	39	331	2829	1108	39.17
C7552	919	61	4411	9338	3529	37.79
C2670	519	46	629	5200	3130	60.19
C432	155	52	210	2193	314	14.32
C499	115	23	146	873	216	24.74
alu2	365	46	415	3315	577	17.41
alu4	598	60	974	8203	1513	18.44
dalu	345	33	586	5676	2539	44.73
8bm	87	28	50	900	439	48.78
16bm	303	57	225	9145	5587	61.09
Avg.	**362.2**	**46.2**	**765.0**	**5377.9**	**2367.9**	**40.08**

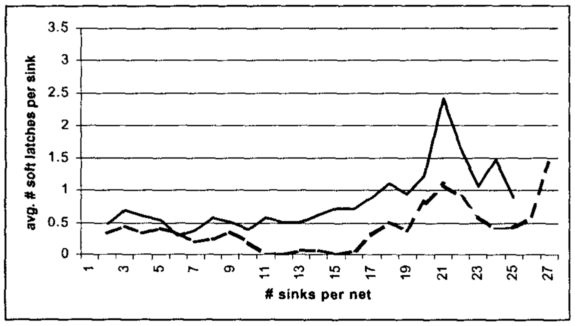

Figure 6: Sinks/net vs. avg.# soft latches/sink

6. *Conclusions and future work*

In this paper we have presented a global routing technique for the wave-steering design methodology. Given a circuit decomposition and its placement, we automatically find a rectilinear routing path hard-latch assignment. After the signal arrival times and the permissible levels to LB input signals are calculated, an optimal signal level assignment is determined by solving the appropriate satisfiability problem. Current results indicate that the number of soft latches is approximately ten times of that of hard latches, and their reduction is an essential task in the wave-steering methodology.

In our future work, the algorithm to maximize the number of sharing soft latches will be refined so as to further reduce the burden of soft latches. We adopt a heuristic in this paper, where sets of objective clauses are justified on a one-by-one basis, and the sets of clauses once satisfied will never be removed. More sophisticated scheduling of objective clause sets and/or dynamic removals of satisfied sets will be our future objective. Furthermore, we will consider the rerouting of paths after we obtain the hard/soft latch assignments such that they are shared as much as possible.

7. Acknowledgment

The last three authors were supported in part by the NSF grant CCR 9811528 and in part by California MICRO program through Xilinx.

8. References

[1]. V. Bertacco et al, "Decision Diagrams and Pass Transistor Logic Synthesis", Proc. ACM/IEEE Int'l Workshop on Logic Synthesis, pp. 1-5, May 1997.

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

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

[4] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, and S. I. Long, "Wave Steering in YADDs: A Novel Non-iterative Synthesis and Layout Technique", Proc. DAC'99, pp 466-471, 1999.

[5] A. Mukherjee, M. Marek-Sadowska, and S. I. Long, "Wave Pipelining YADDs- A Feasibility Study", Proc. IEEE Custom Integrated Circuits Conf. '99, pp. 559-562, 1999.

[6] A. Singh, L. Macchiarulo, A. Mukherjee, and M. Marek-Sadowska, "A Novel High-Throughput FPGA Architecture", Proc. Eighth ACM International Symp. FPGAs, pp. 22-27, 2000.

[7] A. Singh, A. Mukherjee, M. Marek-Sadowska, "Interconnect pipelining in a throughput intensive FPGA architecture", Proc. 9th International Symposium on FPGAs, Feb. 2001, pp.153-160

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

[9] L. Macchiarulo and M. Marek-Sadowska, "Wave-Steering One-hot Encoded FSMs", Proc. DAC'2000, pp. 357-360, 2000.

[10] S. H. Gerez, "Algorithms for VLSI Design Automation", Wiley 1998.

[11] N. Funabiki and T. Higashino, "A Minimal-State Processing Search Algorithm for Satisfiability Problems," to appear in Proc. IEEE SMC 2001.

[12] M. R. Garey and D. S. Johnson, "Computers and Intractability: a Guide to the Theory of NP-Completeness," Freeman, 1979.