OPTIMAL USE OF 2-PHASE TRANSPARENT LATCHES IN BUFFERED MAZE ROUTING

S. Hassoun
Tufts University, Medford, MA

Abstract

Clocking frequencies continue to increase due to the demand for higher performance. Together with the larger die sizes, multiple clock cycles are now required to cross a chip. A routing tool must thus insert registers as well as buffers while minimizing the path latency. This paper addresses optimal buffered path construction across multiple clock cycles using 2-phase transparent latches. We demonstrate the benefits of routing using latches over registers, and we present a polynomial routing algorithm. Our results confirm the correctness of our algorithm.

1. INTRODUCTION

Future SoC designs give rise to new problems in routing and buffer insertion. A particular concern is that multiple clock cycles are now required to cross a chip. This is due to the increase in die sizes and the continued demand for high performance and thus higher clocking frequencies. Another concern is the need to route through complicated terrains due to circuit blockages (e.g. IP blocks, memories – typically used and reused to decrease time to market) and wire blockages (e.g. data paths).

Routing and buffering long routes that require multiple clock cycles while using registers was recently addressed [4]. The authors introduce an algorithm that minimizes latency (or number of registers) along a route. The resulting register-to-register delays are less than the clock period. The proposed algorithm is polynomial and builds upon the Fast Path framework proposed in[6].

We address in this paper the problem of buffered routing while using transparent (i.e. level-sensitive) latches. Such latches are typical in high-performance designs, and they can improve on the performance when compared to using registers[3]. We demonstrate the benefits of using latches in routing, and we present a polynomial algorithm to do such optimal routing. To find the optimal buffered-latched path between a sink t and a source s, we explore all routing and latch insertion points within a given routing area while considering both physical and wire obstacles. Our models and algorithms builds on those developed in[6, 4]. We restrict our discussion here to 2-phase clock schedule; the work, however, can be extended to handle any clock clock schedule.

The remainder of the paper is partitioned as follows. Section 2 presents our routing and clocking models. Section 3 demonstrates the use of 2-phase level-sensitive latches in routing and its benefit over registers. Section 4 defines our routing problem and presents our algorithm. We present experiments in section 5 and conclude in section 6.

2. BACKGROUND

2.1. Clock Schedule Model

We use the SMO clocking formulation[5] to model our clocks. A 2-phase clock schedule, Φ, is an ordered collection of 2 periodic signals, ϕ_1 and ϕ_2, having a common period π. Because phases are periodic, a *local time zone* of width π is associated with each phase. Each phase ϕ_i is characterized by two parameters e_i and ω_i. Parameter e_i represents the absolute time when ϕ_i begins (relative to an arbitrary global time reference). Parameter ω_i is the length of time that ϕ_i is active (latch is open). We assume that the design intention and thus the clock schedule specify that a signal departing from a latch k must be captured by the next latching edge (which occurs after the latching edge of k) of the following latch l.

Because we perform our routing starting from the sink and moving towards the source, it is useful to translate a time measurement a from the local time zone of ϕ_i into the *previous* local time zone of ϕ_j. To do so, we subtract from a a phase shift operator $E'_{i,j}$ defined as:

$$E'_{i,j} = \begin{cases} e_j - e_i - \pi & \text{if } i < j \\ e_j - e_i & \text{otherwise.} \end{cases}$$

$E'_{i,j}$ is π less than the original $E_{i,j}$ phase shift operator in[5] which translates a time measurement a from the local time zone of ϕ_i into the *next* local time zone of ϕ_j.

A 2-phase non-overlapping clocking scheme is demonstrated in Figure 1. If the clock period π is 10 time units, $\omega_1 = 7$, $\omega_2 = 3$, $E'_{1,2} = -7$, $E'_{2,1} = -3$, then an arrival of 8 in ϕ_2's time zone translates to an arrival of 11 relative to the *previous* occurrence of ϕ_1's time zone.

The earliest arrival time at the output of a latch k clocked by ϕ_i is at the opening edge: $\pi - w_i + latch\_propagation$. The latest arrival at the input of a latch clocked by ϕ_i is at the closing edge: $\pi - setup\_time$. Both times are in reference to ϕ_i's time zone.

2.2. Routing Model

To model physical and wiring blockages, one may construct a grid (maze) graph $G(V,E)$ over the potential routing

0-7803-7761-3/03/$17.00 ©2003 IEEE

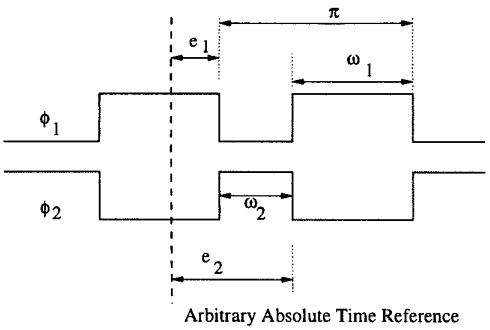

Figure 1. An example 2-phase clock schedule.

area, whereby each node corresponds to a potential insertion point for a buffer or latch element, and each edge corresponds to part of a potential route. Edges in the grid graph which overlap wiring blockages are deleted, and nodes that overlap physical obstacles are labeled blocked. More precisely, we define a label function $p : V \leftarrow \{0,1\}$ where $p(v) = 0$ if $v \in V$ overlaps a physical obstacle and $p(v) = 1$ otherwise.

For each edge $(u,v) \in E$, let $R(u,v)$ and $C(u,v)$ denote the capacitance and resistance of a wire connecting u to v. Let $R(g)$, K(g), and $C(g)$ respectively denote the resistance, propagation delay, and input capacitance of a given buffer or latch element g. We use the resistance-capacitance π-model to represent the wires b, a switch-level model to represent the gates, and the Elmore model to compute path delays.

A *path* from node s to t in the grid graph G is a sequence of nodes $(s = v_1, v_2, \ldots, v_k = t)$. An *optimized path* from s to t is a path plus an additional labeling m of nodes in the path. We assume that the target node has a latch: $m(t) = l$, and a phase assignment, $phase(m(t)) = k$, where $k \in \{\phi_i, \phi_2\}$. We also assume that the signal arrive at the target by the closing edge of ϕ_i. We assume that the source node will also have a latch; however, we do not assign it a phase. We assume that the required time at the source to be the opening edge of the phase clocking the source latch. Let B be a buffer library consisting of non-inverting buffers. Each node $v \in V - s,t$ may be assigned a buffer from B or a latch, or not have a gate (corresponding to $m(v) = 0$). When assigned a latch, v is also assigned a phase. The assumptions regarding the phase assignments and required arrival times at the sink are simply to simplify the presentation in this short paper.

3. ROUTING WITH LATCHES V.S. REGISTERS

To understand the benefits of routing with latches v.s. registeres, consider Figure 2. Assume that the specified clock period is 10 ns, and that the delay through each grid edge is 1 ns. The optimal routed path with registers from source to sink in Figure 2(a) requires 3 registers. Thus, there is a 3 clock period latency, or a 30 ns time latency. The circuit blockage requires placing a register on each side of the blockage. Slack thus exists in the routing segments on both sides of circuit blockage.

Assume that transparent symmetric latches are used in Figure 2(b). In this case, the time latency is from the rising edge

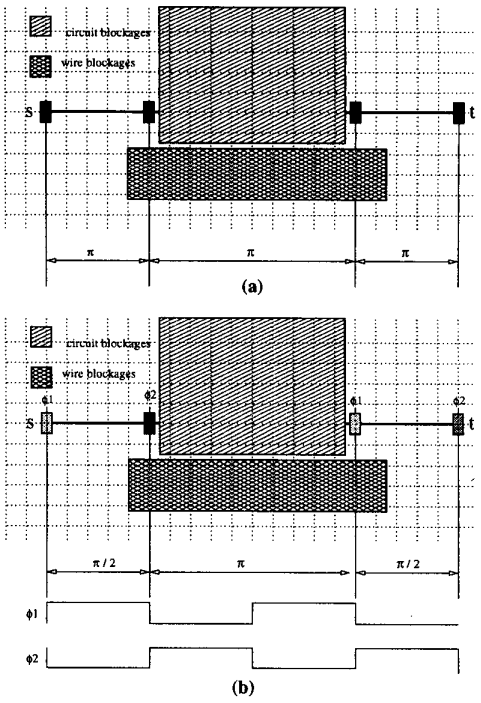

Figure 2. An example of routing within a single-clock domain

of ϕ_1 to the second falling edge of the ϕ_2, for a total time latency of 20 ns. Like in (a), latches are forced to surround the circuit blockage to accommodate the long delay. However, there is no slack along the routing segments on both sides of the circuit blockage. Latches thus offer greater time flexibility which could result in a smaller latency.

4. PROBLEM & SOLUTION

Our general routing problem can be formally defined as follows:

Problem: Given a routing graph $G(V,E)$, a set $I = B \cup \{l\}$, and two nodes $s,t \in V$, find a feasible buffered-latched path from s to t such that the time latency from s to t is minimized.

The time latency here refers to the time from the opening edge of the latch at the source to the closing edge of the latch at the sink.

Several key issues were considered when developing the routing algorithm. First, latches must be inserted periodically whenever the timing constraints dictate that a latch is needed. Furthermore, latches are clocked by different phases and hence latch ordering must be respected. Our example circuit in Figure 2 required alternating ϕ_1 and ϕ_2.

Second, latches cannot be inserted too far apart, otherwise, the routed signal will not be latched properly and timing violations occur. Furthermore, since signals propagate across latches only during latch transparency periods, and

we are propagating from sink towards source, we compute *arrival time* at each grid point – in contrast to *delays* that were used in[4, 6].

Finally, since latches have a convenient *local time zone* notion, and a phase shift operator that translates from one local time zone to another, we will use the shift operator to update the arrival time at the latch input upon placing a latch at a grid point. To facilitate comparing partial solution, the expansion of candidate solutions from sink to source will proceed in waves of partial solutions wherein each wave corresponds to a different number of latches.

The Latched-Buffered Path (LBP) algorithm is presented in Figure 3. Like the algorithms in[4, 6], we wish to explore all buffering and latching solutions, and we need to prune solutions as we route the signal from the sink back to the source. The core data structures are two priority queues, Q and Q^*, that sort candidates based on the latest arrival time. We also use two arrays A_1 and A_2 to mark whether a solution with $m(v) = l$ has been generated for node v to prevent multiple candidates with same-phase latches from getting inserted at v.

The algorithm begins by initializing Q to the set containing a single sink candidate, while Q^* is empty. The current phase is set to be the one that must precede the phase of the latch at the sink. Candidates are then iteratively deleted from the Q and expanded either to add an edge (Step 5) or a buffer (Step 7) or a latch (Step 8). Buffers and latches are only inserted when $p(v) = 1$: when there is no overlap with a blocking physical obstacle (Step 6).

When candidate solution expands via edges or buffers, the arrival time is updated by subtracting the necessary RC delay. The new solution is then only added to the *current* queue Q if no timing violation will occur. However, when a new latch is inserted, the arrival time is set to be the minimum of $\pi - setup(l)$, the closing edge of the latch (i.e. a signal cannot possibly arrive at the output of the latch before then), or by subtracting the necessary RC delay. Moreover, to account for transforming into a previous time domain, the arrival time is updated by subtracting the phase shift operator $E'_{current\_phase,previous\_phase}$. This candidate solution is then added to the *next* queue, Q^*, if no timing violation will incur.

If the source is reached, it is pushed onto the Q in Step 5, and when it is eventually popped from the queue, it is returned as the optimum solution (Step 4). With each addition to either queue, candidates for the current node are checked for inferiority and then pruned accordingly. The complexity of the aglorithm is polynomial, as the one presented in [6].

5. EXPERIMENTAL RESULTS

We implemented our algorithm in C and ran the experiments on a Sun Solaris Enterprise 250. We use estimated parameters for a 0.07μ technology as reported by Cong and Pan [2]. We use a single buffer size of 100 times minimum gate width and triple wide wires, and assume delay characteristics for the latches to be identical to that of the buffer. As in [1], we use a 25 by 25 mm chip and place the source and sink 40 mm apart. These choices guarantee a significant number of clock cycles will be required to traverse from s to t. We used a grid separation of $0.5mm$, which corresponds to a 100×100 grid.

Our results are presented in Table 1. For each clock period, we used a 2-phase symmetric schedule. We report the time latency, the number of latches, the number of inserted buffers, and the run time. As π decreases, more latches are needed but less buffers are used as latches are used as buffers to reduce RC delays. The run time decreases as π decreases as more aggressive pruning occurs with shorter clock periods.

We then compare our results with routing using registers and without using any at the bottom part of the table. We see that when π is 250, the latches are able to achieve a smaller clock latency. The run times are smaller in the register case because one can prune more aggressively[4].

Table 1. Buffered Latched Routing statistics as a function of π and grid separation of 0.5 mm and a grid of 100× 100.

Routing with latches				
π	Latency	# Latches	# Buffers	Run Time
1000	3000	4	12	100.36
400	2800	12	2	82.3
250	2875	21	0	59.03
Routing with registers				
π	Latency	# Latches	# Buffers	Run Time
1000.00	3000	2	13	49.93
400.00	2800	6	7	36.08
250	3000	11	0	27.20
Routing without registers or latches				
∞	2738.57	-	16	48.65

6. CONCLUSION

Automated buffered routing is a necessity in modern VLSI design. Any CAD tools currently performing buffer insertion will eventually have to deal with synchronizer insertion. This paper described the important problem of routing signals and synchronizing them using transparent latches. The two contributions of this paper are the polynomial algorithm for latched buffered routing and demonstrating that latched buffered routing can achieve better performance than registered buffered routing.

7. REFERENCES

[1] J. Cong. "Timing Closure Based on Physical Hierarchy". In *Proceedings of the International Symposium on Physical Design*, pages 170–174, 2002.

[2] J. Cong and Z. Pan. "Interconnect Performance Estimation Models for Design Planning". *IEEE Transactions on Computer-Aided Design*, 20(6):739–752, June 2001.

[3] C. Ebeling and B. Lockyear. "On the Performance of Level-Clocked Circuits". In *Advanced Research in VLSI*, pages 242–356, 1995.

[4] S. Hassoun, C. Alpert, and M. Thiagarajan. "Optimal Buffered Routing Path Constructions for Single and Multiple Clock Domain Systems". In*Proc. of the IEEE International Conference on Computer-Aided Design (ICCAD)* 2002.

[5] K. Sakallah, T. Mudge, and O. Olukotun. "Analysis and Design of Latch-Controlled Synchronous Circuit". In*Proc. 27th ACM-IEEE Design Automation Conf.*, pages 111–7, 1990.

[6] H. Zhou, D. F. Wong, I.-M. Liu, and A. Aziz. "Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations". *IEEE Transactions on Computer-Aided Design*, 19(7):819–824, July 2000.

Latched Buffered Routing Algorithm (G,B,s,t,m',l,Φ)

Input: $G(V,E) \equiv$ Routing grid graph
$B \equiv$ Buffer library
$s \equiv$ source node
$t \equiv$ sink node
$m' \equiv$ initial labeling with $m'(t) = l$
$l \equiv$ latch for clocking signal
$\Phi \equiv$ given clock schedule: $\pi, E'_{1,2}, E'_{2,1}, \omega_1, \omega_2$

Vars: $Q \equiv$ priority queue of candidates
$Q^* \equiv$ queue holding next candidate wave
$\alpha = (c,a,m,v) \equiv$ Candidate at v
$A_i \equiv$ Marking of nodes with inserted $l(\phi_i), \forall_i \in P_\Phi$

Output: $m \equiv$ Labeling of complete s-t path

1. $Q \leftarrow \{(C(r), \pi, predecessor(phase(m'(t))), m', t)\}$.
   $Q^* = \emptyset$, $A_i(v) = 0, \forall v \in V$ and $\forall i \in P_\Phi$
   $current\_phase = predecessor(phase(m'(t)))$
2. **while** ($Q \neq \emptyset$) **or** ($Q^* \neq \emptyset$) **do**
   if ($Q = \emptyset$) **then**
   $Q = Q^*$, $Q^* = \emptyset$.
   $current\_phase = predecessor(current\_phase)$
3. $(c,a,m,u) \leftarrow extract\_min(Q)$
4. **if** $u = s$ **and** $a - K(l) - R(l) \cdot c \geq \pi - \omega_{current\_phase}$ **then**
   return labeling m.
5. **for each** $(u,v) \in E$ **do**
   $c' \leftarrow c + C(u,v)$
   $a' \leftarrow a - R(u,v) \cdot (c + C(u,v))/2$
   if $a' \geq \pi - \omega_{current\_phase} + K(l) + min(R(B \cup l)) \cdot c'$ **then**
   push (c', a', m, v) onto Q and prune
6. **if** $p(u) = 1$ **and** $m(u) = 0$ **then**
7. **for each** $b \in B$ **do**
   $c' \leftarrow C(b)$, $m(u) = b$
   $a' \leftarrow a - R(b) \cdot c$ - K(b)
   if $a' \geq \pi - \omega_{current\_phase} + K(b) + R(b) \cdot c'$ **then**
   push (c', a', m, u) onto Q and prune
8. **if** $A_{current\_phase}(u) = 0$ **and** $a - K(l) - R(l) \cdot c \geq \pi - \omega_{current\_phase}$ **then**
   $m(u) = l$, $A_{current\_phase}(u) = 1$
   $a' = min(\pi - setup(l), a - K(l) - R(l) \cdot c) - E'_{current\_phase, pred(current\_phase)}$
   push $(C(l), a', m, u)$ onto Q^* and prune

Figure 3. The Latched-Buffered Path (LBP) Algorithm.