Minimizing Sensitivity to Clock Skew Variations Using Level Sensitive Latches

François-R. Boyer*, El Mostapha Aboulhamid*, and Yvon Savaria†

Abstract — **We propose a method for improving the tolerance of synchronous circuits to delay variations on the clock distribution. Instead of retiming and clock skew scheduling applied to edge-triggered flip-flops, as used by most other methods, we use level-sensitive latches placed based on a schedule of the operations. The resulting circuit can have a non-zero tolerance even at the optimal clock period, which is impossible with edge-triggered flip-flops.**

1 Introduction

As clock frequencies get higher and the clock distribution network gets larger, the fluctuation on the arrival time of the clock signals gets important compared to the clock period. Clock distribution networks can be made more precise [4], but they are not perfect. For a circuit to work properly at high speed, it must be designed to have tolerance to delay fluctuations.

Previous methods trying to maximize the tolerance used mainly retiming and clock skew scheduling [3]. Each of these approaches used separately will not guarantee maximum tolerance, and combining them is not trivial. The method in [5] finds an optimal combination of these two, using a mixed-integer linear program, for which there is no known polynomial time algorithm. Also, only single-phase edge-triggered circuits are supported. We will show that the optimal solution they find is no more optimal if level-sensitive latches and multiple phases are permitted.

We present a new way to have a higher tolerance to the timing variations on the clock, placing level-sensitive latches according to a schedule found by a software-pipelining technique. This is an extension to what we presented in [1], where the goal was to achieve the optimal clock period, but where clocks were supposed to be perfect.

The main contributions of this work are:

- It permits tolerance to clock skew and clock jitter while running at optimal clock period. Method [5] trades clock frequency for higher tolerance.
- It permits more tolerance than any method based on retiming and clock skew scheduling.
- It demonstrates the existence of a tradeoff between register count (hardware complexity) and tolerance.
- The method has a polynomial execution time, and it is very fast as illustrated by the experimental results.

Combining small time complexity and possibility of tradeoff between tolerance and hardware complexity permits an exploration of the design space, where a satisfying tolerance, not necessarily optimal, is obtained, while keeping the register count within an acceptable limit.

* DIRO, Université de Montréal, Québec, Canada.
E-mail: {boyerf, aboulhamid}@IRO.UMontreal.CA.
† DGEGI, École Polytechnique de Montréal, Québec, Canada.
E-mail: savaria@VLSI.PolyMtl.CA.

2 Preliminaries

2.1 Input Circuit Definition

As most methods do, we support single-phase edge-triggered circuits formed by combinational computing elements separated by registers, similarly to the original retiming article [7]. With the reference model, circuits are represented as a finite, vertex-weighted, edge-weighted, directed multigraph $G = \langle V, E, d, w \rangle$. The vertices V represent the functional elements of the circuit, and they are interconnected by edges E. Each vertex $v \in V$ has a propagation delay $d(v) \in \mathbb{Q}$, which is the maximum delay before its outputs stabilize. Each edge $e \in E$ is weighted with a register count $w(e) \in \mathbb{N}$, representing the number of registers separating two functional elements. All registers are edge-triggered and controlled by the same clock.

We extend the d and w functions for paths in the graph. For any path $v \overset{p}{\rightsquigarrow} v_k = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \dots \xrightarrow{e_{k-1}} v_k$, we define:

$$d(p) = \sum_{i=0}^{k-1} d(v_i) \qquad w(p) = \sum_{i=0}^{k-1} w(e_i)$$

For registers a on an input arc of operation v, and b on an input arc of v', we define $a \overset{p}{\rightsquigarrow} b$ as $v \overset{p}{\rightsquigarrow} v'$.

We also support lower bounds on the delay of an element, which requires another vertex-weight $d_{\min}(v) \in \mathbb{Q}$. To simplify figures and readability, the minimum delay is considered zero in the examples.

2.2 Scheduling and software-pipelining

A *schedule* s is a function $s : \mathbb{N} \times V \to \mathbb{Q}$, where $s_n(v) \equiv s(n, v)$ denotes the time at which the n^{th} iteration of operation v is starting. A schedule s is said to be *periodic* with period P (all iterations having the same schedule), if:

$$\forall n, \forall v \in V,\ s_{n+1}(v) = s_n(v) + P$$

A schedule is valid iff the operations terminate before their results are needed (whilst respecting resource constraints if any). If the only constraints come from data dependency, s is *valid* iff for all edges $v \xrightarrow{e} v'$,

$$s_n(v) + d(v) \le s_{n+w(e)}(v').$$

A periodic schedule that satisfies those constraints can be found in $O(|V|\,|E|)$ using a longest path algo-

rithm like Bellman-Ford, for any valid period. The problem of finding the minimal period is the same as the well known minimal cost-to-time ratio cycle problem, the cost being the number of registers and the time being the combinational delays, which can be solved in $O(|V| |E| \log(|V| d_{max}))$ using Lawler's algorithm [6]. A cycle is called *critical* if its ratio is the same as the minimum ratio, and a path is critical if all its edges are on a critical cycle. Note that the optimal period is the inverse of the minimum ratio.

For example, the circuit of Fig. 1 (the same as in [7]) has a clock period of 24. Using retiming techniques we can obtain a single phase clocked circuit with a period of 13. An optimal period of (3+7)/1=10 can be achieved. This minimal period is determined by the critical cycle shown with the gray arrow [1]. Meeting this performance necessitates a multi-phase clock; and a feasible register placement is shown in Fig. 2. On non-critical edges, more than one register may be required to respect the desired maximum distance between registers; this can increase tolerance to timing variations as will be explained later.

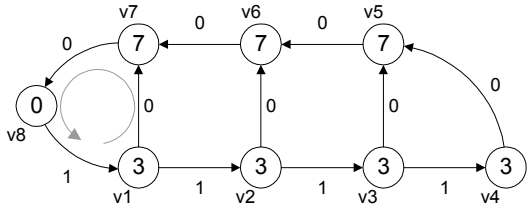

Fig. 1. A simple circuit with delays on vertices and register count on edges. A critical cycle is shown in gray.

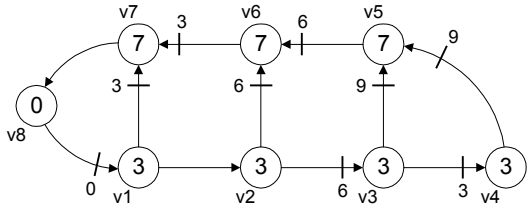

Fig. 2. A possible way to place registers in Fig. 1, with a period of 10. The phases of the registers are indicated beside them.

3 Clock and Register Constraints

We consider that registers can be activated at any time using multiple clock phases. Each register is activated at a specific phase of the clock, according to the schedule, all clock signals having the same period. This permits to follow any periodic schedule.

3.1 Edge-triggered Registers

In a circuit with edge-triggered registers, a register must not be activated before the combinational circuit preceding it has stabilized. That is, a register placed on an input of operation v must be activated following a valid schedule for v. There is another constraint, as the circuit calculates values at each cycle. All registers must be activated before the result start to change for the next result. Therefore, for each path $a \overset{p}{\rightsquigarrow} b$, between registers a, and b, with no other register between them, we have:

$$s_n(a) + d(p) \le s_{n+k}(b) \le s_{n+1}(a) + d_{min}(p)$$

where k is 0 or 1 if b depends on a from the same or from the previous iteration, respectively.

This forces to place at least one register on each path p longer than the period $P + d_{min}(p)$, for periodic schedules. Algorithm `BreakPath` [1] resolves this problem when d_{min} is considered to be zero.

If there is no slack in the schedule, the time at which registers are activated must be exact, as any deviation will make the schedule invalid. A circuit using edge-triggered registers on a critical path has zero tolerance to schedule variations when running at the optimal clock period. The method in [5] finds the maximum tolerance for single-phase edge-triggered circuits with a relaxed period, if only retiming and clock skew scheduling is permitted.

For example, on the circuit of Fig. 3 (containing only a critical cycle of Fig. 1) at the optimal period of 10, registers can be placed at either `a` or `b`, or both `a` and `b` using two clocks as shown on Fig. 4.

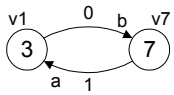

Fig. 3. The critical cycle from Fig. 1, with the two possible positions for registers (a, b).

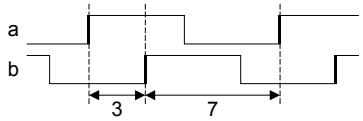

Fig. 4. Clocks when both a and b are edge-triggered registers in Fig. 3.

3.2 Level-sensitive Registers

With level-sensitive registers, the constraints seem similar, but are in fact more flexible, as will be shown in the following. A value must not be latched (by the disabling edge of the clock) before it has stabilized, but the register can be enabled some time before (by the enabling edge of the clock). To correctly follow the schedule of operations, and not delay them, a latch placed on an input of operation v must be enabled before the schedule of v. The register must still be enabled at the schedule of v and must be disabled before the value starts to change for the next one. Then it must stay disabled long enough for a valid value to propagate to other registers. If we have a valid schedule for an edge-triggered circuit, we can use level-sensitive latches if the following constraints can be met with a non-null enabling period (the enable time different from the disable time, for each register). For each path $a \overset{p}{\rightsquigarrow} b$, between registers a and b, with no other register between them,

$$s_{n+k}(b_e) \le s_{n+k}(b) \le s_{n+k}(b_d) \le s_{n+1}(a_e) + d_{min}(p),$$

where a_e and b_e are the enabling time of the latches, and a_d and b_d is the disabling time of the latches; the other values are as in the edge-triggered case.

Here we consider the value of register a to be valid at time $s_n(a)$, even if it is disabled only at a later time: $s_n(a_d)$. As the original edge-triggered schedule was valid, we can prove that this is true [1].

Even if there is no slack in the schedule of operations, there may be slack on the enable and disable time of registers. Since no clock is sent according to the schedule of operations, small errors in clocks arrival times may be tolerated.

With the same example as in the edge-triggered case (Fig. 3), using level sensitive latches permits to have imperfect clocks even at the optimal period, as shown on Fig. 5.

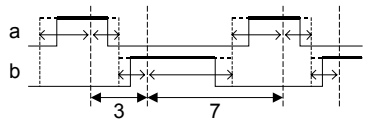

Fig. 5. Clocks when level-sensitive registers are at both a and b in Fig. 3. All clock edges can move as indicated by the small arrows, without changing the circuit's behavior.

4 Maximum Tolerance to Clock Variations

This section analyses how much error can be tolerated on the enable time and the disable time of a register. We say that a circuit has a tolerance of δ if all clock edges can be randomly moved by $\pm\delta$ time units from their nominal values, without changing the result produced by the circuit. Note that in [5], the tolerance τ is defined as the width of the interval, thus $\tau = 2\delta$.

4.1 Tolerance at Optimum Clock Period

THEOREM 1. At optimum clock period, the maximum tolerance for registers a and b on a critical path, without registers between them, is $(P - (s_{n+k}(b) - s_n(a)) + d_{min}(a \rightsquigarrow b))/4$, where $d_{min}(a \rightsquigarrow b)$ is the minimum $d_{min}(p)$ for all paths $a \overset{p}{\rightsquigarrow} b$.

PROOF. On a critical path, when the circuit runs at maximum frequency, the schedule must be followed exactly. From previous section, we must satisfy the following constraint for all paths $a \overset{p}{\rightsquigarrow} b$:

$$s_{n+k}(b_e) \le s_{n+k}(b) \le s_{n+k}(b_d) \le s_{n+1}(a_e) + d_{min}(p)$$

The lower bound on $d_{min}(p)$ is $d_{min}(a \rightsquigarrow b)$, from the definition, and as the schedule in periodic $s_{n+1}(a_e) = s_n(a_e) + P$ (if there is no error on the clock). So we have that:

$$s_{n+k}(b_e) \le s_{n+k}(b) \le s_{n+k}(b_d) \le s_n(a_e) + P + d_{min}(a \rightsquigarrow b)$$

To have a tolerance of δ, we must be able to move all clock edges by that value and still satisfy the constraint. We want to maximize δ under:

$$s_{n+k}(b_e) + \delta \le s_{n+k}(b) \le s_{n+k}(b_d) - \delta$$
$$s_{n+k}(b_d) + \delta \le s_n(a_e) + P + d_{min}(a \rightsquigarrow b) - \delta$$
$$s_{n+1}(a_e) + \delta \le s_{n+1}(a)$$

We obtain the maximum δ when the inequations are at equality. Putting them together, we get:

$$(s_{n+k}(b) + \delta) + \delta = (s_n(a) - \delta) + P + d_{min}(a \rightsquigarrow b) - \delta$$
$$\Leftrightarrow \delta = (P - (s_{n+k}(b) - s_n(a)) + d_{min}(a \rightsquigarrow b))/4 \quad \square$$

The tolerance to clock edge deviations depends on the distance between registers; a shorter time between them gives a higher tolerance. The maximum tolerance will be obtained when registers are placed on all edges in the graph.

LEMMA 1. At optimum clock period, the maximum tolerance on critical paths is $(P - \max_{v \in V} \{ d(v) - d_{min}(v) \})/4$.

PROOF. To minimize the distance between registers and have a valid schedule, the distance will be the delay of one operation. LEMMA 1 follows directly from THEOREM 1, as the longest operation will determine the maximum distance between registers. $\square$

LEMMA 2. At optimum clock period, the required number of registers on each cycle of the graph will be multiplied by at least $1/(1 - 4\,\delta/P)$ if all paths are critical and d_{min} is zero.

PROOF. From THEOREM 1, $\delta = (P - dist)/4$, where $dist$ is the distance between registers with a path between them. In the optimum case, the registers will have equal distances; a cycle containing m registers will have a length of $m \cdot dist$. P is optimal and all paths are critical, so for any cycle $P = m \cdot dist/n$, where n is the original number of registers. We obtain $\delta = (P - P \cdot n/m)/4$, which means that $m/n = 1/(1 - 4\,\delta/P)$. $\square$

For our example of Fig. 3, the maximum tolerance is $(10 - 7)/4 = ¾$, if d_{min} is considered zero. Looking at Fig. 5, we see that to maximize the length of the four arrows in the section lasting 3 time units, each arrow must be one fourth of the 3 time units. This is also the maximum tolerance for the circuit of Fig. 2, as the longest delay between registers is the same.

To show how more registers will give more tolerance, we will use the circuit of Fig. 6. The circuit has an optimal period of 16 and would have no tolerance at that speed if one edge-triggered register were used. If level-sensitive registers are placed at a and c, a tolerance of $(16 - 8)/4 = 2$ can be achieved, and if registers are added at b and d, that tolerance can go up to $(16 - 4)/4 = 3$. In that later case, the clocks will be as shown in Fig. 7.

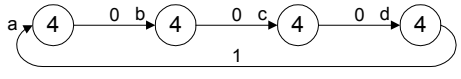

Fig. 6. Simple loop circuit to show possible tolerance.

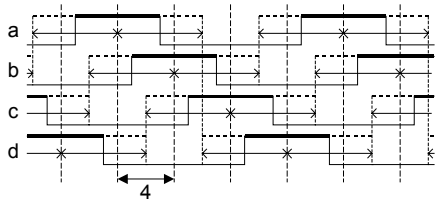

Fig. 7. Clocks with highest tolerance for circuit Fig. 6 with period 16.

4.2 Tolerance with a Relaxed Period

If we give some slack time to the circuit, using a longer period or on non-critical paths, the circuit could have a higher tolerance. As a small error on $s_n(a)$ can be accepted, the tolerance could be higher than half the width of the enabling interval (between a_e and a_d) of the register. Of course we must still guarantee that $s_n(a_e) \le s_n(a_d)$, which is not a problem because whatever the variations on the delays are, the transitions on a wire will always keep the same order. We do not currently have a method to optimize tolerance on non-critical paths, they are optimized as if they were also critical, which may underestimate the actual tolerance.

5 Experimental Results

We have implemented a method to place registers based on THEOREM 1. The distance between registers must be bounded by a value lower than the optimal period in order to achieve a non-zero tolerance. It has been applied to some `.edif` circuits from LGSynth93 benchmark suite [8]. To compare the results with those presented in [5], we tried to use the same circuits and delay model, but their delays are a bit randomized. Gate delays are $a + b\cdot(fanout \pm ½)$, where a and b depends on the gate type and come from the library `iwls93.mis2lib`. The library gives a and b for the rise and fall time, for each case we add or subtract ½ from the number of connections to the output of the gate (*fanout*), then the highest and lowest values of the four calculated delays are used as maximum and minimum delays, respectively.

In the first part of Table 1, we obtain the same tolerance as in [5] without deviating from optimal clock period. A zero tolerance would be obtained with the method [5]. The last two circuits are there to show that our method is fast even on larger circuits. The columns $|V|$ and $|E|$ give the size of the circuit graph in number of vertices and edges, P is the optimal period and the period at which we want to run the final circuit. The targeted tolerance is δ, and the register count in the resulting circuit with that tolerance is ×reg times that of the original circuit (not the one at optimal period). The time taken for the optimization and register placement, in seconds on a PII 450MHz, is shown in the column CPU (s). The time for the placement of registers to achieve a specified tolerance is about 12% of that total time. *To compare with the time taken by the method in [5], the circuits s713 and dk512 took 13.6 hours and 21.5 hours, respectively, also on a PII (as stated in their paper)*. Our method is fast but does not currently minimize the number of registers, and it does not exploit the slack on some paths.

The number of registers to achieve a certain tolerance is shown on Fig. 8. Both axis have been normalized: the tolerance is the fraction of the period and the register count is the factor by which the number of registers increased compared to the original circuit. The solid line shows the theoretical minimum number from LEMMA 2. The dots are the values obtained by trying different tolerances on the benchmark circuits.

6 Conclusion and Future Work

We showed that a circuit with a tolerance higher than that of any edge-triggered circuit could be obtained using level-sensitive latches activated by different clock phases. The tolerance cannot be over ¼ the period, when running at the optimal period, but can be non-zero, which is impossible with edge-triggered circuits. A higher tolerance requires more registers, which leads to possible tradeoffs. Currently the algorithm does not minimize the number of registers and does not exploit slack times on non-critical path, or when running the circuit at a relaxed clock period. We plan to develop an algorithm that gives higher tolerance with fewer registers, using those slack times.

Circuit	$\|V\|$	$\|E\|$	P	δ	×reg	CPU (s)
bbtas	35	75	4.63	0.315	2	0
dk14	73	199	6.16	0.7	1.25	0
dk512	44	122	4.83	0.35	3.2	0.016
s208	43	93	4.22	0.835	28.17	0.016
s713	397	575	47.34	1.89	2.05	0.125
s9234.1	2931	4057	30.25	2	2.10	0.750
s38417	23985	33248	27.78	2	2.36	8.703

Table 1. Tolerance (δ) and register increase (×reg) for some circuits.

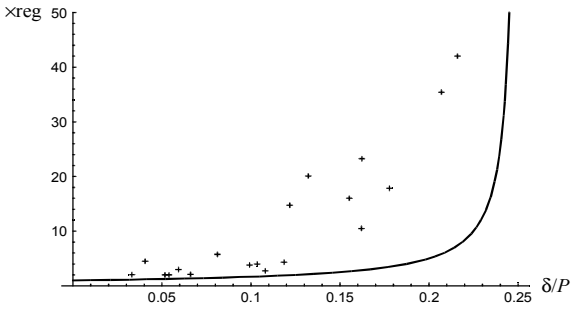

Fig. 8. Register count for different tolerance from 0 to ¼ the period. Solid line is the bound from LEMMA 2; dots are actual results.

References

[1] F. R. Boyer, E. M. Aboulhamid, Y. Savaria, and M. Boyer, "Optimal design of synchronous circuits using software pipelining techniques," *ACM Transactions on Design Automation of Electronic Systems*, vol. 7, no. 2, April 2002, in press, available on www.acm.org.

[2] F. R. Boyer, E. M. Aboulhamid, and Y. Savaria, "An efficient verification method for a class of multi-phase sequential circuits," *IEEE International Conference on Electronics, Circuits & Systems*, December 2000, in press.

[3] J. P. Fishburn, "Clock skew optimization," *IEEE Transactions on Computers*, vol. 39, pp. 945-951, July 1990.

[4] E. G. Friedman, "*Clock distribution networks in VLSI circuits and systems*," IEEE press, 1995.

[5] E. G. Friedman, X. Liu, and M. C. Papaefthymiou, "Minimizing sensitivity to delay variations in high-performance synchronous circuits," *Proceedings of Design Automation and Test in Europe*, 1999, pp. 643-649.

[6] E. Lawler, "*Combinatorial Optimization: Networks and Matroids*," Saunders College Publishing, 1976.

[7] C. E. Leiserson, and J. B. Saxe, "Retiming synchronous circuitry. *Algorithmica*," vol. 6, no. 1, 1991, pp. 3-35.

[8] http://cbl.ncsu.edu/CBL_Docs/lgs93.html