Timing Uncertainty Analysis for Time-of-Flight Systems

John R. Feehrer and Harry F. Jordan
Optoelectronic Computing Systems Center
Campus Box 525
University of Colorado, Boulder, CO 80309-0525
{feehrer,harry}@boulder.colorado.edu

Abstract

Time-of-flight synchronization is a new digital design methodology that eliminates all latching devices, allowing higher clock rates than alternative timing schemes. Synchronization is accomplished by precisely balancing connection delays. Many effective pipeline stages are created by pipelining combinational logic, similar in concept to wave pipelining but differing in several respects. Due to the unique flow-through nature of circuits and to the need for pulse-mode operation, time-of-flight design exposes interesting new areas for CAD timing analysis. This paper discusses how static propagation delay uncertainty limits the clock period for time-of-flight circuits built with opto-electronic devices. We present algorithms for placing a minimum set of clock gates to restore timing in feedback loops that implement memory and for propagating delay uncertainty through a circuit graph. A mixed integer program determining the minimum feasible clock period subject to pulse width and arrival time constraints is discussed. Algorithms are implemented in XHatch, a time-of-flight CAD package.

1 Introduction: Time-of-Flight Design

Time-of-flight synchronization is a new approach to digital systems design that eliminates all bistable latching elements, allowing pipelining within combinational logic and potentially much higher clock rates than achievable with conventional or wave pipelining. A time-of-flight digital circuit is a sequential circuit in which all memory is realized by explicit feedback loops and interacting signals are synchronized by adjusting path delays between components. In conventional pipelined systems, the longest sensitizable combinational logic path between pipeline registers puts a lower bound on the clock period [1]. Wave pipelining [2, 3, 4, 5] eliminates some registers by pipelining combinational logic, using registers to break feedback loops and periodically re-synchronize waves. In contrast, "time-of-flight" circuits use clock gates to re-synchronize pulses which may traverse feedback loops. A very high number of effective pipeline stages, or "waves," is theoretically possible with this technique. Maximum clock rate is limited only by the controllability of propagation delays through switching and interconnection devices and by switching bandwidths. Computational latency and computational throughput are decoupled to a larger extent than is possible with the alternative pipeline timing schemes.

Critical requirements for time-of-flight systems are high switching bandwidth and highly controllable propagation delay. These features characterize electro-optic switches and optical fibers, and have been exploited successfully in building digital optical time-of-flight systems [6]. A simple general purpose stored-program optical computer using no bistable devices [1] for data storage or synchronization has been built and tested in the laboratory to demonstrate the principles [7].

A niche commercial application for time-of-flight techniques is packet routing processor design for packet-switched telecommunications or multiprocessor interconnection networks in which data is already in optical form. The goal is to eliminate opto-electronic conversion bottlenecks to allow 10-20 Gbit/sec link bit-rates. Research in our group has focused on design, simulation, and construction of deflection-routed optical networks [8, 9]. Packet routing processor designs that use this routing protocol are highly amenable to pipelining and time-of-flight synchronization. With recent developments in integrated optics, GaAs opto-electronics, and flip-chip bonding, there is the exciting prospect of building time-of-flight processors on a single substrate [10]. We believe the unique flow-through nature of time-of-flight circuits will create a fertile new area for CAD research in timing optimization as technology continues to improve. Our motivation for writing this paper is to stimulate discussion and further interest in time-of-flight methods within the CAD community.

Because of the absence of bistable devices to correct for logic delay variations, time-of-flight system design requires very detailed timing analysis; arrival times of pulses at node inputs must obey two-sided constraints. Delays over interconnections must be precisely adjusted so that arrival times always coincide. A CAD tool is necessary to manage the complexity of meeting these synchronization constraints. The tool balances delays by adjusting lengths of optical waveguide interconnections (single-mode optical fiber in the case of the optical computer). It also accounts for variations in device propagation delays. Variations may

[1] For our purposes, the term "bistable devices" refers to the general category of static-storage devices, including edge-triggered and level-sensitive flip-flops or latches.

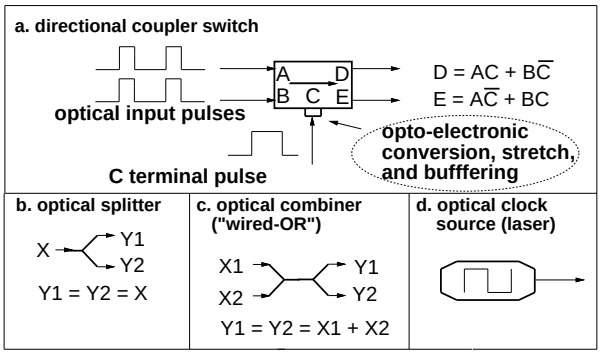


Figure 1: Time-of-flight circuit components

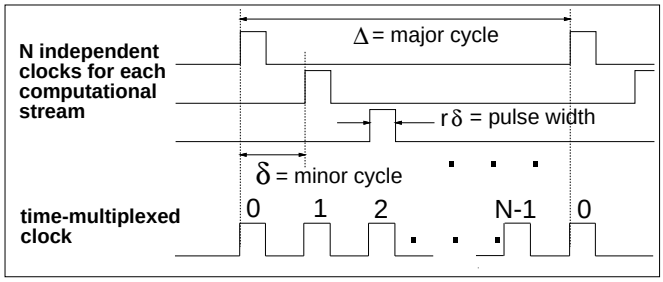


Figure 2: Timing model for time-of-flight design

be static, caused by manufacturing process differences, as well as dynamic, induced by temperature and power supply fluctuations, dispersion, and laser jitter. We refer to the combined effect of these variations as *delay uncertainty*. Due to space limitations, we focus this paper on the static component of delay uncertainty.

Section 2 introduces circuit components and the time-of-flight timing model. Section 3 presents an algorithm for placing a minimum set of clock gates covering all feedback loops. Section 4 discusses how to compute uncertainty accumulated along signal paths, and uses the results in an integer program to minimize the clock period. In Section 5 we propose additional problems for future work. Our algorithms are implemented in a digital optical CAD package called XHatch, available via anonymous ftp [11]. XHatch is an X-Windows program performing schematic capture, delay distribution, clock period minimization, simulation, and optical power loss analysis.

2 Components and Timing Model

In optical time-of-flight circuits, information is constantly moving. Memory is implemented with optical delay lines which recirculate streams of pulses [6]. A delay line memory's length depends on the clock rate, number of bits to be stored, and properties of the physical medium. Because there is no steady state anywhere in the circuit, there is no meaning to "settling time." Capacitive charge or discharge does not contribute to the delay of a pulse as it travels along a path, yielding uniform rise and fall times and decreased static skew, and eliminating the data-dependent delay seen in CMOS and other VLSI technologies. Terms such as "path sensitization" and "false paths" [1] imply level-mode (non-return-to-zero) signaling and do not apply to the pulse-mode time-of-flight circuits we have implemented.

While our experience in time-of-flight design is limited to use of the 2x2 coupler switches described below, implementation with conventional logic gates is also being studied. The progress in controlling CMOS delays [5] makes this prospect especially interesting. The following features characterize the time-of-flight circuits we discuss in this paper:

- Switches are logically complete 2x2 optical directional couplers. Figure 1.a shows a schematic illustration. The devices are electro-optically switched from the "cross" to the "bar" state by converting the optical pulse entering the C terminal to an electrical pulse which drives an electrode. The electric field created by the pulse generates a birefringence [12] and thereby alters the coupling of power between the two waveguides.

- Every circuit connection is point-to-point with uni-directional signal flow. A connection begins at a node output or primary input and ends at a node input or primary output. Fanout is done using optical fiber splitters; fiber combiners implement a wired-OR function (see Figure 1).

- Return-to-zero (RZ or pulse mode) signaling is used throughout. A logic 1 is encoded as a pulse rather than simply as a level. Every signal's pulse width falls within a known range, with nominal width equal to the master clock signal width. This mode allows the *major cycle* Δ (lower-bounded by irreducible delays around feedback loops) to be sub-divided into N *minor cycles* (lower bounded by switching bandwidth and delay uncertainty) [13]. Refer to Figure 2. N clocks with period Δ are time-multiplexed, offset by integral multiples of the minor cycle δ. Each controls an independent computational stream. Time-multiplexing increases peak computational throughput by a factor of N. Guard bands are inserted between pulses to provide for timing uncertainty. The multiplexed clock duty cycle is r, $0 < r < 1$.

- Timing restoration is accomplished using clock gating. See Figure 3. The data pulse to be restored is "stretched" by the amount T_s at the C terminal. Laboratory circuits have used a series of cascaded electronic OR gates for this purpose. The connection leading up to the C terminal is adjusted so that the ideal stretched pulse is centered around the optical clock pulse arriving at A. The stretched pulse must completely overlap the clock, over all possible arrival time variations [6]. The clock's arrival is relatively precise; it is directed through the coupler to the D output.

- A global optical clock signal generated by a laser is distributed to "clock gate" switches configured as described above for clock gating.

- Any pair of optical pulses entering a switch or combiner must at least partially overlap over

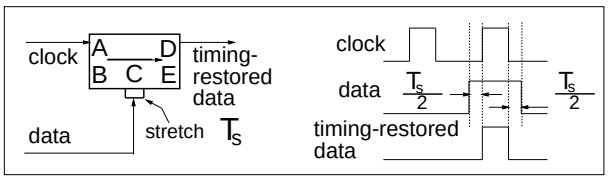


Figure 3: Clock gating and stretch

all manufacturing and operating condition variations.

A time-of-flight circuit is represented by a directed multigraph $G = (V, E)$ which may contain self-loops [14]. Vertices represent devices that combine or fan out signals. More specifically, a vertex represents an *interaction point*, an abstraction of the exact physical center of the corresponding device [15], so that parameters of specific input and output *ports* can be associated with the edges connected to them. Edges represent connections such as optical fibers. A set of timing parameters is associated with every edge. One parameter gives nominal propagation delay, specifying the delay from the edge's initial vertex to its final vertex; this delay is the sum of the source device output port delay, the connection delay, and the destination device input port delay. Other edge parameters describe the worst-case arrival time uncertainty and pulse width for a pulse entering the destination node device. Stretch is employed at every switch C terminal, so that (1) C terminal arrival uncertainty and edge-rate asymmetry produced by the drive electronics do not propagate beyond this point, and (2) output glitches are not generated.

Figure 4 shows a time-of-flight circuit schematic and graph. It is a 4-bit counter whose increment signal, occurring once every 4 major cycles, is generated by switches S4 and S5. The number of major cycles of delay which XHatch is called upon to distribute around each loop is provided by the designer, who specifies "lumped delays" for connections in multiples of the major cycle (these connections are drawn with "bubbles"). Lumped delays give the circuit its desired sequential behavior [15].

2.1 Relationship to Wave Pipelining

Since the time-of-flight methodology is related to wave pipelining, it is instructive to point out their similarities and differences. The two techniques are similar in that: (1) they both create effective pipeline stages by allowing multiple coherent waves of data to propagate over a combinational block, and have mechanisms for periodically restoring timing (registers for wave pipelining, clock gates for time-of-flight), (2) they both require path delays to be balanced to prevent wave interference, (3) they both allow multistage systems (systems with feedback [5]), meaning different numbers of clock cycles of delay between successive registers in wave pipelining or between successive clock gates in time-of-flight, and (4) they both have a global clock whose arrival times at registers [5] or clock gates may be deliberately skewed to optimize the clock period. Significant differences between the two approaches are: (1) in wave pipelining, registers bound wave-pipelined combinational blocks whereas in time-of-flight systems clock gates without static storage capability surround the blocks, (2) wave-pipelined circuit delays can be adjusted by either adding buffers or adjusting drive currents, whereas time-of-flight adjustments are done by altering connection delays only, and (3) wave-pipelined circuits use non-return-to-zero (NRZ) signaling, whereas time-of-flight requires pulse-mode (RZ) signaling due to clock gating, creating timing constraints explicitly involving duty cycle.

The distinction between what Gray et al. [5] call the "overall system speedup factor" $\bar{k}$ and our time-multiplexing factor N is more subtle. In both methodologies, the speedup gained by adding effective pipeline stages must be the same for all loops. While all loops in their wave-pipelined system have the same ratio of effective pipeline stages to physical registers, the analogous ratio of minor cycles to clock gates is not fixed for all loops in a time-of-flight system. Time-of-flight loops can contain arbitrarily many major cycles of delay. A pair of clock gates can span any number of major cycles, limited only by delay uncertainty accumulated between the gates.

3 Optimal Clock Gate Placement

To prevent cumulative timing drift from destroying synchronization, we require that every feedback loop in a time-of-flight circuit contain at least one clock gate in its path. This is a necessary but not sufficient condition for a *functional* circuit. We start with an *idealized* circuit having no clock gates, so the problem is to identify a minimum set of edges in G into which clock gates should be placed, formally stated as:

Problem statement: *Given a time-of-flight circuit graph $G = (V, E)$ for an ideal circuit, find a minimum cardinality edge set $E_f \subset E$, such that the spanning subgraph $G' = (V, E - E_f)$ is acyclic.*

This problem is equivalent to the Minimum Feedback Edge Set (MFES) problem, in the class NP-complete. It was solved by Lempel and Cederbaum [16], and we adapt their method as follows. The time-of-flight circuit graph G is represented using an adjacency matrix containing edge labels and augmented with 1's on the diagonal. The *permanent* expansion for this matrix is computed, giving a sum of products with addition interpreted as OR and multiplication as AND. The permanent is computed as the determinant but with all signs positive. The Boolean absorption law $A + AB = A$ is applied to decrease the number of product terms. Each product of edges in this expression represents a feedback loop. OR and AND are then interchanged to give the dual expression, a product of sums, which is reduced, again with Boolean absorption, to produce a minimal sum-of-products form. Each product in this expression is a set of edges which, if removed, will break all feedback loops. The minimum cardinality product term identifies the MFES. With a few modifications, this technique can handle self-loops and parallel edges. The edges making up an MFES for the counter graph in Figure 4b are

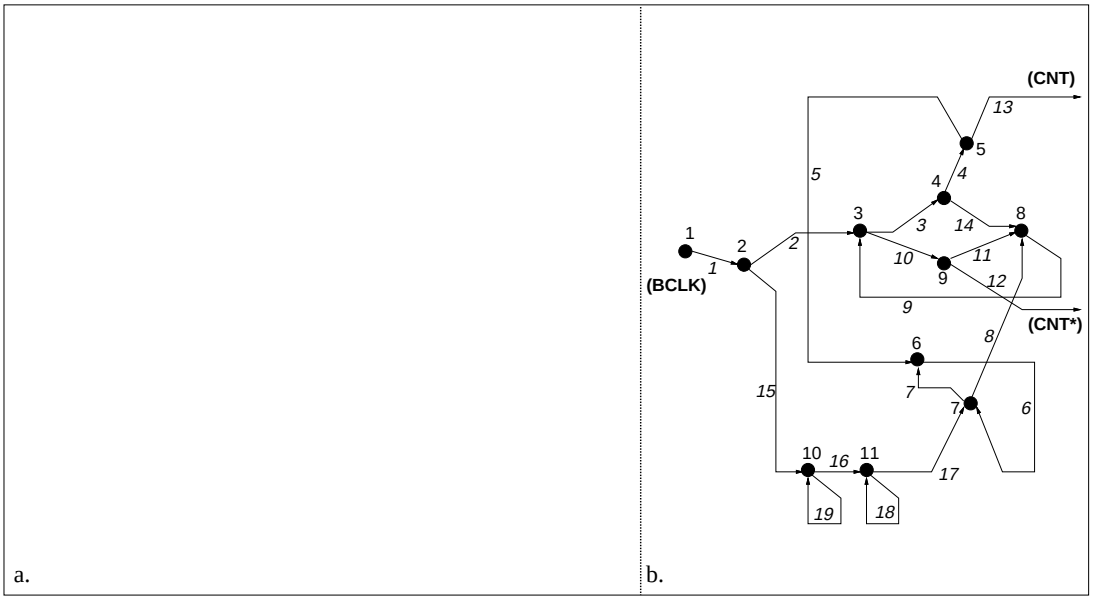


Figure 4: (a) XHatch schematic of 4-bit counter, and (b) its circuit graph

$\{e_6, e_9, e_{18}, e_{19}\}$.

Complexity is a concern, but can be minimized by considering the special structure of time-of-flight circuits. Let FI_{max} be the maximum fanin of any vertex and FO_{max} be the maximum fanout of any vertex. Then if we expand the permanent by minors, at each stage expanding along the row or column with fewest non-zeros, the worst-case number of operations grows as $(\min[FI_{max}, FO_{max}]+1)^{|V|}$. For our circuits, $FI_{max} = 3$ and $FO_{max} = 2$. The XHatch implementation first eliminates from consideration edges incident with zero-fanin or zero-fanout vertices, then does a depth-first permanent expansion and calls Boolean minimization routines in the SIS-1.1 package [17] for the two absorption steps. In practice, the number of expansions is far below the upper bound of $3^{|V|}$, as seen in column 4 of Table 1. Column 5 gives total seconds of run time (recorded on a SPARCStation IPX) to compute the MFES from the adjacency matrix, including permanent expansion time and SIS time.

The exponential complexity of the permanent expansion becomes problematic at around $|V| = 150$. Expansion did not complete for the largest circuits due to insufficient memory. Since proposed integrated optics time-of-flight circuit graphs will exceed this size, alternative MFES solution methods are needed. One alternative we have implemented is to break the circuit graph into its strongly connected components (SCC's) [14], and then apply the technique above to each SCC individually. The last two columns of Table 1 show the number of vertices in the largest SCC, and the total execution time using this method. An optimal solution is obtained with large savings in computational effort for large circuits having relatively small SCC's, such as the Optical computer and Packet synchronizer. For less modular circuits such as the multiplier, we are considering use of an approximation algorithm. For example, the $O(|V| + |E|)$ randomized algorithm proposed by Berger and Shor [18] finds an acyclic subgraph containing at least $\frac{2}{3}|E|$ edges.

There are typically multiple minimum feedback edge sets, in which case we are interested in the one set yielding minimum minor cycle, δ_{min}. To compute the minimum feasible δ for a selected MFES solution, we need to determine (1) worst-case cumulative uncertainty for each signal path, and (2) a set of constraints that relate these cumulative uncertainties to δ. We discuss these issues in Section 4.2.

4 Clock Period Optimization

Previous time-of-flight work [15] dealt with the first step in timing analysis, computing the minimum major cycle Δ_{min}. The second step is to compute δ_{min}. In the first step, we ignore delay uncertainty and use nominal delays rather than solving a stochastic program with delays modeled as random variables [19]. Delay uncertainty is addressed in the second step, where we also compute a value for T_s ensuring that Δ_{min} is feasible under all static delay variations. We assume uncertainty limits δ before switching bandwidth does. Our research is motivated by the use of components for which this assumption is true.

4.1 Minimizing The Major Cycle

The minimum *major cycle* for a practical time-of-flight circuit is constrained by the device latencies and physically irreducible interconnect delays around feedback loops. We refer to the problem of where to increase connection delays beyond their designer-specified minimums in order to synchronize node input arrival times as *delay distribution*. This problem can be formulated as a linear program minimizing the feasible major cycle and the sum of added delays, sub-

Circuit	$\mid V \mid$	$\mid E \mid$	Expansions	Run time (exp. + SIS)	Largest SCC	Run time using SCCs
4-bit serial counter	14	21	55	2	7	3
Serial sorter	32	47	84	7	12	7
Optical computer [7] subsystem 1	115	142	1206	326	10	24
Optical computer subsystem 2	130	162	3863	15691	14	35
Packet synchronizer	140	188	out of mem.	-	14	55
Entire optical computer	238	331	out of mem.	-	10	252
Floating-point multiplier	507	648	out of mem.	-	249	-

Table 1: Performance of MFES solver in XHatch

ject to a set of loop constraints for each of a set of $\mid E \mid - \mid V \mid +1$ basis loops [14]. A loop constraint equates actual loop delay (loop sum of minimum delays plus added connection delays) to the loop sum of designer-specified lumped delays [15].

An alternative delay distribution algorithm [15] distributes delays locally, shifting delays from vertex input edges to output edges to satisfy minimum delay constraints on output edges. This is essentially retiming [20], except that what is redistributed is continuously adjustable delay rather than discrete registers or latches. Wong's "coarse tuning" phase of wave pipeline optimization [2] is similar, though it *combines* the loop basis formulation with a retiming algorithm. In the time-of-flight case, a local delay distribution trial can be done only with a known major cycle, but by embedding the local algorithm in a binary search, it is possible to obtain the minimum feasible major cycle in a logarithmic number of trials [21].

4.2 Minimizing the Minor Cycle

Given Δ_{min} and one or more MFES solutions, we would like to find δ_{min}. Since delay uncertainty limits δ_{min} and clock gates reduce uncertainty, it may be necessary to add additional clock gates to achieve a performance goal expressed as a target minor cycle. Thus we have two problems:

Problem 1: *Given a set of k minimum feedback edge sets* $\mathcal{E}_F = \{E_{f1}, E_{f2}, ..., E_{fk}\}$, *select the one set* $E_{f*} \in \mathcal{E}_F$ *minimizing* δ.
Problem 2: *Given target minor cycle* $\delta_T < \delta_{min}$, *where* δ_{min} *is for set* E_{f*}, *find a minimum cardinality edge set* $E_t \subset E - E_{f*}$ *into which clock gates can be placed so that the minimum minor cycle for the new circuit,* δ'_{min}, *satisfies* $\delta'_{min} \leq \delta_T$.

Lee and Reddy [22] study a generalization of the minimum feedback *vertex set* problem, in the context of locating scan flip-flops. Their objective is to find minimum set of vertices whose removal gives an acyclic graph with maximum distance between any pair of vertices satisfying an upper bound. Similarly, in Problem 2, the "longest path," i.e. the path between clock gates having highest cumulative uncertainty, constrains δ_{min}. However, since there are many other constraints, satisfying $\delta'_{min} \leq \delta_T$ cannot be easily formulated as a generalization of the MFES graph problem. Due to limited space we discuss only Problem 1, solved as follows.

1) Select a minimum feedback edge set $E_{fj} \in \mathcal{E}_F$.

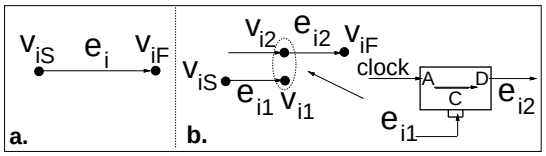


Figure 5: Circuit graph before (a) and after (b) clock gate is placed into edge e_i

2) Augment the acyclic network $G' = (V, E - E_{fj})$ by adding a pair of edges $\{e_{i1}, e_{i2}\}$ and a pair of vertices $\{v_{i1}, v_{i2}\}$ representing a clock gate, for each $e_i \in E_{fj}$, connected as follows (see Figure 5). Let v_{iS} and v_{iF} be the initial and final vertices, respectively, for e_i. v_{i1} represents the "stretch terminal" and v_{i2} represents the clock input and restored output terminals. The initial and final vertices for e_{i1} are v_{iS} and v_{i1}, respectively. The initial vertex for e_{i2} is v_{i2}; the final vertex is v_{iF}. The augmented network G'' is still acyclic, since there is no edge connecting v_{i1} and v_{i2}. Clock gate outputs are source nodes of G'' and stretch terminals are sink nodes.

3) Using a series of $O(\mid E \mid)$ depth-first searches on G'' beginning at each of the $\mid E_{fj} \mid$ clock gate output vertices and extending to the clock gates' stretch vertices, compute the worst-case arrival time uncertainties and pulse widths at each node along every path in G''.

4) Solve a mixed integer program to minimize δ subject to a set of pulse overlap and pulse width constraints using arrival time uncertainties computed in Step 3.

5) Repeat Steps 1-4 for each remaining set in $\mathcal{E}_F$. Record δ for each set.

6) Select the set E_{f*} in $\mathcal{E}_F$ yielding minimum δ; δ_{min} is set to this value of δ.

Our simplifying assumptions regarding device delay uncertainties are that they are bounded, uncorrelated, and additive. Thus, we can associate a bounded *uncertainty interval* $u(e_i)$ with each edge e_i giving worst-case arrival time variation for a pulse entering the final vertex of e_i. See Figure 6. We assume that delay uncertainty distorts leading and trailing edges of pulses uniformly, so a single uncertainty interval suffices for each edge. The nominal pulse *center* is used as a fixed

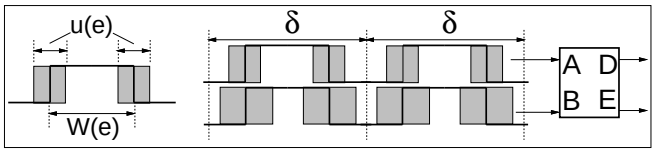


Figure 6: Pulse uncertainty model

reference point from which deviations from nominal arrival are analyzed. The center coincides with the midpoint of the pulse's minor cycle window. Due to uncertainty, the output pulse from a combiner can be wider than its input pulses, so we need also to associate a pulse width, $W(e_i)$, with each edge $e_i \in E$, where $W(e_i) \geq r\delta$.

Arrival time uncertainty increases monotonically as a pulse propagates along a path in G'' having uncertain delays. Assume that connection delay uncertainty is linearly related to nominal connection delay through a constant scaling factor K_f. We take the arrival time uncertainty at the output of a 2x2 switch as the *larger* of the two (A or B port) input uncertainty intervals plus the uncertainty over the corresponding path through the switch. Because the actual mapping of input to output uncertainty depends on the switch state, a more accurate analysis requires simulation. The uncertainty at the output of a splitter is simply the uncertainty interval of the input plus the uncertainty through the splitter. Connections and splitters do not distort pulse width. The same is true for switches, assuming the stretched pulse at C entirely overlaps pulses on A and B. The uncertainty and pulse width assigned to a combiner's output edge are:

$$u(e_j) = \frac{1}{2}\{\max_{e_i \in I(v)} [W(e_i) + u(e_i)] - \max_{e_i \in I(v)} [W(e_i) - u(e_i)]\} + u_{CB} + K_f d(e_j) + u_{IN},$$

$$W(e_j) = \frac{1}{2}\{\max_{e_i \in I(v)} [W(e_i) + u(e_i)] + \max_{e_i \in I(v)} [W(e_i) - u(e_i)]\},$$

where $I(v)$ is the set of input edges to combiner vertex v, e_j is its output edge with delay $d(e_j)$, u_{CB} is the combiner output port uncertainty, and u_{IN} generically signifies input port uncertainty for e_j's final vertex.

Using these device models, we can compute the uncertainty interval for each edge in G'' by traversing the network in a depth-first fashion, beginning at a clock gate output. We extend a search when all input edges for the current vertex have known uncertainty intervals, and backtrack when we reach a stretch terminal or a vertex having at least one input edge with unknown uncertainty. When we have backtracked all the way to the last visited clock gate output, a new search begins at the next unvisited clock gate output. A search can extend from a *switch* input to output as long as both input A and B uncertainties are known, because with adequate stretch, C terminal uncertainty does not affect output uncertainties.

We now explain the constraints which bound the minor cycle; their mathematical presentation follows. The first three constraints deal with uncertainty; the rest concern other timing issues. Uncertainty related constraints use the uncertainty intervals and pulse widths computed as described above, and restrict timing relationships at vertices; in all cases, $u(e_i)$ is the uncertainty interval for e_i *at its final vertex*. The first constraint is that for every pulse, its width must exceed its uncertainty interval so that the pulse overlaps its nominal center; this ensures that two interacting pulses will not miss each other. Second, every pulse must with certainty be less than a minor cycle in width to maintain pulse-mode operation and prevent interference between multiplexed streams (see Figure 6, showing two pulses entering a switch). This applies to both non-stretched and stretched pulses. Third, at each switch (including clock gates), the stretched C terminal electrical pulse must *completely* overlap the A and B input pulses to prevent glitches. Fourth, the major cycle Δ_{min} must be a positive integral multiple N of the minor cycle. Fifth, the pulse width emitted by the laser must be greater than a specified minimum detectable width W_{min}; it is unnecessary to lower-bound downstream pulse widths, since we do not use devices which can shorten pulses. Finally, referring to Figure 5, for every edge e_i in E_{fj}, the MFES under consideration, the sum of delays over the new edges e_{i1} and e_{i2} incident on the clock gate, plus half of the stretch must equal the delay over the original edge e_i. Satisfying this condition preserves synchronization at v_{iF} and centers the ideal stretched pulse around the clock entering v_{i2}. Delays for e_{i1} and e_{i2} are left unknown, and must be at least as large as their specified minimums $m(e_{i1})$ and $m(e_{i2})$.

We assume stretch is done electronically and that it is added in discrete increments, with increment size d_S and uncertainty u_S. (Separate leading and trailing edge uncertainties are needed if low-to-high and high-to-low delays differ significantly.) We make a variable substitution $T_s = pd_S$, where p is a positive integer. If an alternative implementation permits continuous stretch, then the optimization takes the form of a linear program instead of the mixed integer program (MIP) presented as follows. E and V refer to the graph G''. $I(v)$ is the set of input edges for vertex v, E_S is the set of edges entering stretch terminals, V_S is the set of switch vertices, $d(e_i)$ is the nominal delay for edge e_i, and E_{fj} is the MFES under consideration. The MIP is:

minimize δ
subject to:

(1) $W(e_i) \geq u(e_i), \ \forall e_i \in E,$
(2a) $W(e_i) + u(e_i) \leq \delta, \ \forall e_i \in E - E_S,$
(2b) $W(e_i) + pd_S \leq \delta, \ \forall e_i \in E_S,$
(3) $pd_S \geq W(e_i) + u(e_i) - W(e_k) + u(e_k) + pu_S, \ \forall e_i, e_k \in I(v), e_k \in E_S, \ v \in V_S,$
(4) $N\delta = \Delta_{min},$
(5) $r\delta \geq W_{min},$ where r is a constant, $0 < r < 1,$
(6) $d(e_{i1}) + d(e_{i2}) + \frac{pd_S}{2} = d(e_i),$
$d(e_{i1}) \geq m(e_{i1}), d(e_{i2}) \geq m(e_{i2}), \forall e_i \in E_{fj},$
(7) $p, N \in \{1, 2, 3, ...\}, \ \delta > 0.$

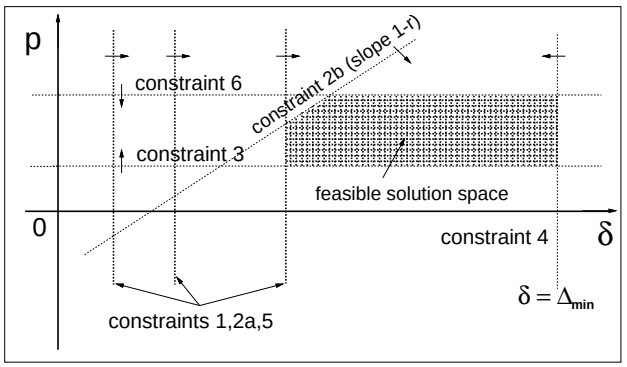


Figure 7: Timing constraint space

The role of δ is made more explicit by replacing each $W(e_i)$ with $r\delta + x(e_i)$, where $x(e_i)$ is non-negative and depends only on uncertainty. The constraint space is shown qualitatively in Figure 7. The feasible region is defined by lower and upper bounds on δ and p, and by the diagonal line with slope $1-r$. The latter arises from the stretched pulse width constraint (2b): as δ increases, for fixed duty cycle r, spacing between pulses increases in absolute terms, creating proportionately more "room" for stretch. Relative magnitudes of the bounds depend on circuit parameters. If constraint 6 is not satisfied for some edge $e_i \in E_{ff}$, delay distribution must be re-run on the new circuit graph, with the minimum delay constraints for e_{i1} and e_{i2} included. In the current formulation of the problem, this re-distribution must be done as a separate step. If e_i is part of the feedback loop that limits major cycle Δ_{min}, then Δ_{min} will increase over its previous value as a result of the re-distribution.

5 Conclusions and Future Work

Time-of-flight circuits are unique in that they use no registers, relying instead on the relatively high controllability of optical delays for synchronization and storage. Clock gates must be placed in a circuit to increase its tolerance to static and dynamic delay variations. We have presented an algorithm to place clock gates using Lempel and Cederbaum's method to solve the minimum feedback edge set problem. SIS performs the Boolean optimizations in this solution procedure. Computation time is reduced for modular circuits by first breaking the circuit graph into its strongly connected components. While our implementation in XHatch has reasonable execution time for smaller designs, approximation algorithms will be needed for large non-modular circuits or for SCC's with over 150 edges. Routines to propagate delay uncertainty through the circuit graph using depth-first search have also been implemented; implementation of the mixed integer program solver to compute δ_{min} is in progress. More work must be done to demonstrate the viability of time-of-flight design. A thorough exploration of the parameter space is needed. When the duty cycle is allowed to vary, for example, the mixed integer program becomes nonlinear. By variable transformation, we can replace nonlinear constraints with those having separable nonlinear functions, permitting use of nonlinear programming methods [19]. Other future projects are incorporation of dynamic delay variations (due to laser jitter and operating condition fluctuations) into the timing constraints, use of stochastic models for delay uncertainty to compute error probabilities, and investigation of the tradeoffs of applying the same stretch at all switches versus having switch-dependent stretch. Finally, we should seek an answer to the question of whether time-of-flight principles can allow faster clock rates in VLSI technologies such as CMOS.

Acknowledgements

This work received funding through the Optoelectronic Computing Systems Center at the University of Colorado, sponsored by NSF ERC grant number ECD 9015128, by the Colorado Advanced Technology Institute, and by NSF grant MIP-9322241. Thanks to Professors Fabio Somenzi and Michael Lightner from Boulder's VLSI CAD group for helpful input, and to Professor Hal Gabow of the Computer Science Dept. for a discussion about minimum feedback edge sets.

References

[1] P. C. McGeer and R. K. Brayton, *Integrating Functional and Temporal Domains in Logic Design*. Kluwer Academic Publishers, 1991.

[2] D. C. Wong, G. D. Micheli, and M. J. Flynn, "Designing high-performance digital circuits using wave pipelining: Algorithms and practical experiences," *IEEE Transactions on CAD*, pp. 25–46, January 1993.

[3] D. A. Joy and M. J. Ciesielski, "Clock period minimization with wave pipelining," *IEEE Transactions on CAD*, vol. 12, pp. 461–472, April 1993.

[4] W. K. Lam, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Valid clocking in wavepipelined circuits," Proceedings of ICCAD, 1992, pp. 518–525.

[5] C. T. Gray, W. Liu, and R. K. G. III, "Timing constraints for wave pipelined systems," Tech. Report NCSU-VLSI-92-06, Dept. of Electrical and Computer Engineering, North Carolina State University, December 1992.

[6] T. Soukup, R. Feuerstein, and V. Heuring, "Implementation of a fiber-optic delay-line memory," *Applied Optics*, vol. 31, pp. 3233–3240, June 10 1992.

[7] P. Main, R. Feuerstein, V. Heuring, H. Jordan, J. Feehrer, and C. Love, "Implementation of a general purpose stored-program digital optical computer," *Applied Optics*, vol. 33, p. 1619, March 10 1994.

[8] D. Blumenthal *et al.*, "First demonstration of multihop all-optical packet switching," *IEEE Photonics Technology Letters*, March 1994.

[9] J. Feehrer, L. Ramfelt, and J. Sauer, "Design and implementation of a prototype optical deflection network," ACM SIGCOMM '94 Conference on Communications Architectures, Protocols, and Applications, August 31 1994.

[10] H. F. Jordan, A. R. Mickelson, B. V. Zeghbroeck, and I. Januar, "An integrated optics, stored program computer," Topical Meeting in Optical Computing, Optical Society of America, March 1993, pp. 318–321.

[11] M. Salerno, "Xhatch user's manual," Tech. Report 91-25, Optoelectronic Computing Systems Center, University of Colorado Boulder, October 1991. ftp cs.colorado.edu:/pub/distribs/xhatch.

[12] A. Yariv, *Optical Electronics*. Saunders College Publishing, 4th ed., 1991.

[13] H. F. Jordan and V. P. Heuring, "Time multiplexed optical computers," Supercomputing 91, 1991, pp. 370–378.

[14] J. A. McHugh, *Algorithmic Graph Theory*. Prentic Hall, Inc., 1990.

[15] J. P. Pratt and V. P. Heuring, "Delay synchronization in time-of-flight optical systems," *Applied Optics*, vol. 31, pp. 2430–2437, May 10 1992.

[16] A. Lempel and I. Cederbaum, "Minimum feedback arc and vertex sets of a directed graph," *IEEE Transactions on Circuit Theory*, vol. CT-13, pp. 399–403, December 1966.

[17] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit synthesis," Tech. Report UCB/ERL M92/41, Electronics Research Lab., University of California, Berkeley, May 4 1992.

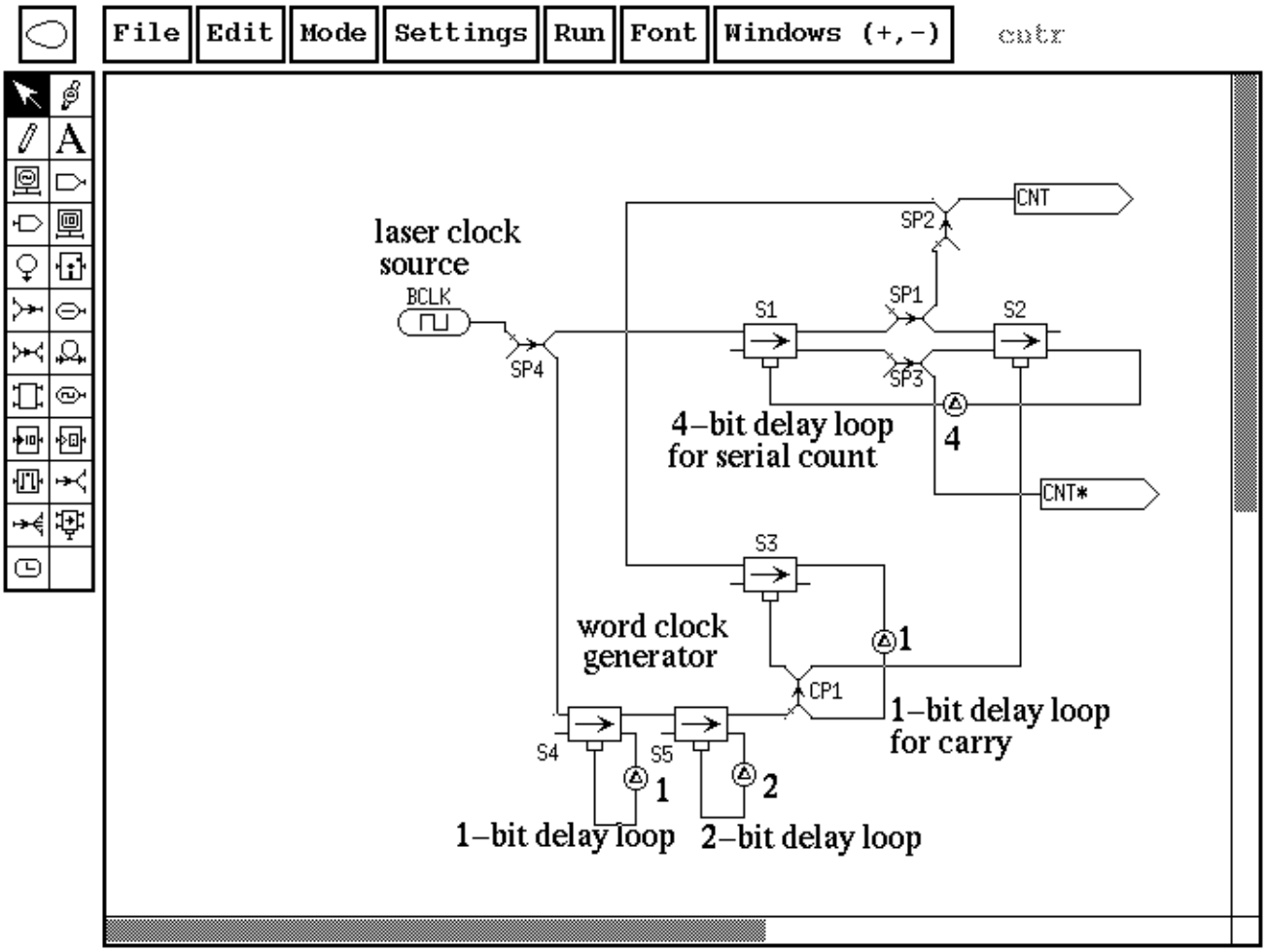
[18] B. Berger and P. W. Shor, "Approximation algorithms for the maximum acyclic subgraph problem," Proceedings of 1st Annual ACM-SIAM Symposium on Discrete Algorithms, 1990, pp. 236–243.

[19] G. Hadley, *Nonlinear and Dynamic Programming*. Addison-Wesley, 1964.

[20] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.

[21] J. R. Feehrer, "Minimizing the major clock cycle in bit-serial time-of-flight synchronized digital circuits," Tech. Report 93-03, Optoelectronic Computing Systems Center, University of Colorado Boulder, March 1993.

[22] D. Lee and S. Reddy, "On determining scan flipflops in partial-scan design," Proceedings of ICCAD, 1990, pp. 322–325.


File
Edit
Mode
Settings
Run
Font
Windows (+,-)
laser clock source
BCLK
SP4
S1
SP1
SP2
CNT
S2
SP3
4
4–bit delay loop for serial count
CNT*
S3
word clock generator
1
CP1
1–bit delay loop for carry
S4
S5
1
2
1–bit delay loop
2–bit delay loop