Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 738983, 6 pages
doi:10.1155/2008/738983

Research Article

Wave Pipelining Using Self Reset Logic

Miguel E. Litvin and Samiha Mourad

Department of Electrical Engineering, School of Engineering, Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053, USA

Correspondence should be addressed to Miguel Litvin, melitvin@gmail.com

Received 1 May 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

This study presents a novel design approach combining wave pipelining and self reset logic, which provides an elegant solution at high-speed data throughput with significant savings in power and area as compared with other dynamic CMOS logic implementations. To overcome some limitations in SRL art, we employ a new SRL family, namely, dual-rail self reset logic with input disable (DRSRL-ID). These gates depict fairly constant timing parameters, specially the width of the output pulse, for varying fan-out and logic depth, helping accommodate process, supply voltage, and temperature variations (PVT). These properties simplify the implementation of wave pipelined circuits. General timing analysis is provided and compared with previous implementations. Results of circuit implementation are presented together with conclusions and future work.

1. INTRODUCTION

Wave pipelining (WP) is a suitable solution for fast arithmetic circuit implementation since it renders high throughput, while reducing the area and power overhead in a pipeline by removing intermediate registers. Such registers result in a latency penalty due to their setup and clock-to-output times and introduce delays for each stage. The area savings are realized due to (a) area devoted to the registers themselves, and (b) area needed for the clock distribution and buffering to control such registers.

In WP designs, each stage holds its output just enough time to guarantee that the next stage will be able to capture the data and start the computation of its own outputs to the following element in the pipe. So ideally, data have to progress *simultaneously* through the stages to achieve the maximum data throughput. Specific timing constraints apply to guarantee no data corruption. These designs are essentially asynchronous, but can be synchronized by the use of input and output registers (implemented with latches, or flip-flops) as long as timing conditions are met so that outputs are captured at an appropriate time. This circuit arrangement is shown in Figure 1.

Self reset logic (SRL) provides circuit implementations where "everything is quiet" when no new data are received. For single-rail implementations, power consumption is "data dependent." In a dual-rail implementation, there will be pulses propagating along the wave path (either at the direct or inverse outputs of each stage); every time new data is presented at primary inputs. We introduce a new family of dual-rail SRL with input disable (DRSRL-ID). A typical cell is shown in Figure 2. In Section 2, we discuss its operation. Section 3 describes the timing constraints that apply when designing wave-pipelining circuits with DRSRL-ID. Section 4 presents an application circuit, and Section 5 presents conclusions and future work.

2. SELF RESET LOGIC

A DRSRL-ID buffer-inverter cell is shown in Figure 2. The gate will generate an output pulse at the direct output (buf) or the inverse output (inv) only if the inputs validate the logic function F or its inverse; otherwise, both outputs will remain at zero. Once inputs evaluation starts, the gate disconnects the inputs for the duration of the cycle time τ defined as follows:

$$\tau = t_{df} + w_p + t_{\text{rec}}, \tag{1}$$

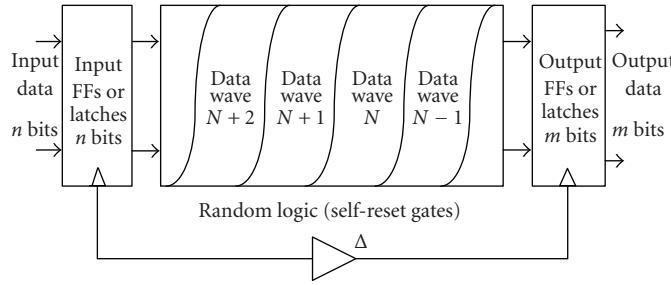

Figure 1: Basic wave pipelined circuit.

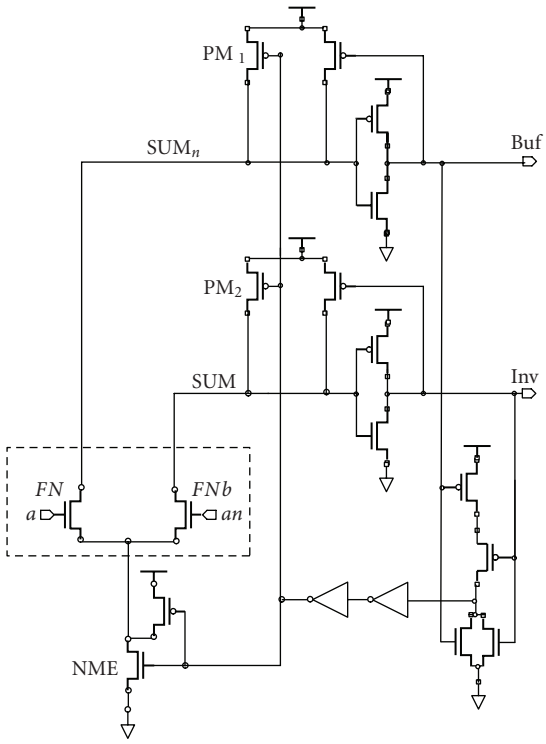

Figure 2: A DRSRL-ID buffer-inverter gate.

where the following definitions hold:

t_{df} : *denotes data delay forward* (the time from the leading edge of the input data transition that validates *F* or *FN* to the leading edge of the pulse at the output);

w_p : *denotes width* of the *output pulse*;

t_{rec} : *denotes recovery time* (the time elapsed from the trailing edge of the output pulse to the trailing edge of the reset pulse).

For inputs to be evaluated, they have to be active for a minimum overlapping time, t_{ovlmin}, that must be longer than the capture time t_c.

t_c *denotes* capture time, the time from the leading edge of the input data transition that validates *F* or *FN* to the falling edge of the pulse at the internal summing node (*SUM or* SUM_n).

We prefer to refer to the output pulse (at either output) so the condition is written as

$$t_{\text{ovlmin}} \geq t_{df}. \tag{2}$$

The width of the output pulse w_p depends strongly on the characteristics of the output stage of the gate, but is independent of the loading while *fan-out* is equal to or less than 8 (for the gate families we have worked with). When a set of inputs validates the logic function *F* (or *Fn*), the corresponding output pulse starts only after the delay t_{df}, but its width w_p depends on the delay through the feedback loop, which postcharges the summing nodes. Also, since we disable inputs, once an output pulse starts, w_p is also *independent of the width of input pulses,* while they satisfy condition (2). Recovery time t_{rec} and delay forward t_{df} can also be made equal for a family of gates. Then, the *cycle time* τ will be a constant for the circuit implemented with these gates. It defines the minimum clock period at which new data can be pushed into the combinational circuit when received from an input register.

Figure 4 shows these timing parameters depicted for idealized waveforms corresponding to the behavior of a typical gate of this type. The outputs *Y* and Y_n correspond to outputs buf and inv of the buffer-inverter cell shown in Figure 2. For a complete description and characterization of these gates, we refer the reader to [1, 2].

An XOR/XNOR gate is shown in Figure 3. In this case, the use of shared elements between the FN and FNb blocks minimizes the number of devices needed. This approach is especially useful when implementing more complex gates. Additionally, in this case, we show the implementation of the self reset without using the extra inverters in the feedback loop. In the present case, as we actually use the internal reset *pulse_rst* signal to disable input readout as well as to control the postcharge of the summing nodes, we can safely play with the width of the resetting pulse, without being affected by the switching activity at the gate inputs.

3. WAVE PIPELINING WITH DRSRL-ID

The wave-pipelining circuit is an asynchronous structure, which can be made to work in a synchronous structure by adding an input and an output register, controlled by clock, as shown in Figure 1. This requires careful selection of the

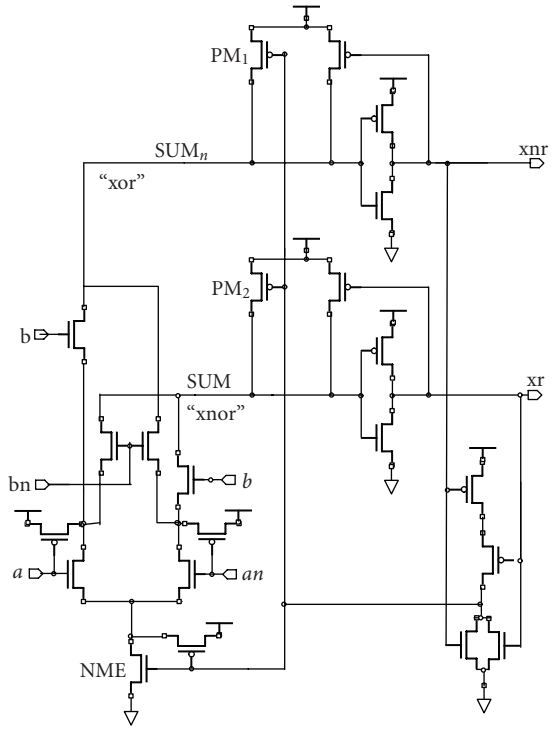

Figure 3: Dual-rail SRL XOR/XNOR gate with input disable.

timing parameters. In the rest of this section, we explain the relationship between these different timing parameters using the following symbols:

- k: number of data waves in the pipeline;
- ck: global clock;
- T_{ck}: period of the global clock;
- ck_{ir}: clock at input register;
- ck_{out}: clock at output register;
- T_L: total latency (time elapsed from launching a data wave from the input register until the corresponding result arrives at the output register);
- T_P: maximum delay through the combinational logic;
- ΔT_P: maximum path delay difference through the combinational logic;
- Δo: phase shift between ck and ck_{out};
- Δi: phase shift between ck and ck_{ir};
- Δ: $T_L \bmod T_{ck} : \Delta o - \Delta i$: constructive skew (phase shift between the clocks that control the launching and receiving registers);
- t_d: register clock-to-Q delay;
- t_s: register setup time;
- t_h: register hold time;
- t_{sk}: uncontrollable clock skew.

The width of the output pulse of the input register w_{pIR} must satisfy the following:

$$t_{df} \le w_{pIR} < T_{ck}. \tag{3}$$

The timing conditions are

$$w_p > \Delta T_{pi} + t_{df} + t_{sk}, \tag{4}$$

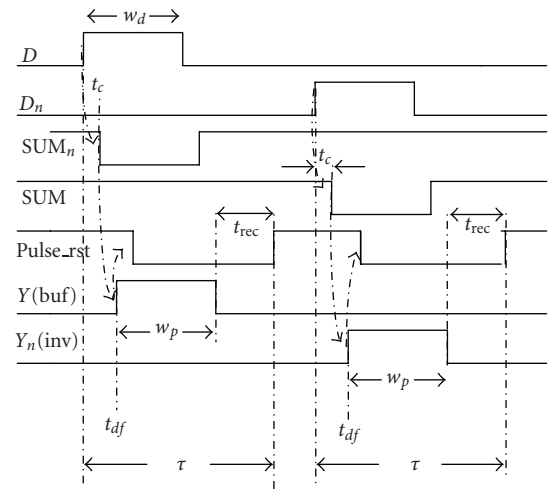

Figure 4: Timing parameters.

where ΔT_{pi} is the worst timing difference expected at any given stage. Then,

$$T_{ck} \ge w_p + t_{\text{rec}}, \tag{5}$$

considering also the delay forward t_{df},

$$T_{ck} \ge w_p + t_{\text{rec}} + t_{df}, \tag{6}$$

that is,

$$T_{ck} \ge \tau. \tag{7}$$

As can be observed in Figure 5, the total latency is

$$T_L = kT_{ck} + \Delta o. \tag{8}$$

Analyzing the situation corresponding to a "late arriving" pulse versus an "early arriving" one, as shown in Figure 6, one can demonstrate that for wave pipelining with DRSRL-ID,

$$T_{ck} \ge \frac{t_d + T_p + t_s + t_{sk} - \Delta}{k}, \tag{9}$$

$$w_p > \Delta T_p + t_s + t_h + 2t_{sk.} \tag{10}$$

Comparing with regular WP CMOS implementation, as shown by [3–5], in that case the conditions for safe pipelining include the following (11) and (12):

$$T_{ck} > \Delta T_p + t_s + t_h + 2t_{sk}. \tag{11}$$

Condition (12) is a two-sided constraint on k, T_{ck}, and $\boldsymbol{\Delta}$, showing the behavior as we sweep frequencies:

$$\frac{T_p + t_d + t_s + t_{sk} - \Delta}{k} \le T_{ck} < \frac{T_p - \Delta T_p + t_d - t_h - t_{sk} - \Delta}{k-1}. \tag{12}$$

Condition (12) applies to the regular WP circuits and defines a set of "valid intervals" of frequencies at which the circuit

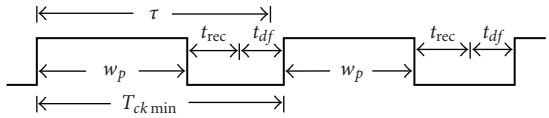

Figure 5: Timing diagram for successive pulses.

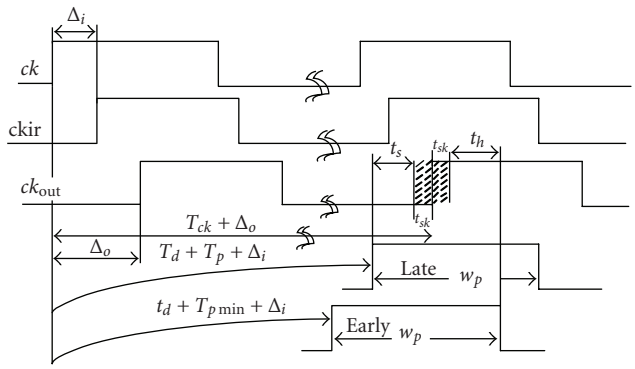

Figure 6: Pulses of the same data wave, with phase shift between input and output register clocks.

will behave in a wave-pipelining mode. The higher the frequency, the narrower the valid clocking interval [4, 5]. So it becomes extremely important to control delays carefully, and there is a strong dependency on the process, voltage, and temperature (PVT) conditions.

In contrast, in DRSRL-ID there is a maximum frequency at which the circuit can operate in WP mode, as stated by conditions (6) and (7). As long as the clock period T_{ck} is greater than the cycle time τ of the gates, early arrival data from data wave $(N + 1)$ will not interfere with late arrivals from the previous data wave. Trying to operate at higher frequencies will generate a situation where, at a given stage, bits would arrive "too early" and will be ignored by that stage, since these gates will still have their inputs disabled. For DRSRL-ID, at frequencies below what condition (7) states, the combinational logic will still function properly (with different k values). The only difficulty resides in capturing the computation result at the output register. Such behavior can be obtained by adding a latch at the end of the combinational logic, which will update when and if new data arrives, that is, converting the last stage into a static one. These characteristics render the technique presented in this paper very desirable.

The conditions on w_p in DRSRL-ID are similar to the conditions on T_{ck} for CMOS WP, rendering a theoretic lower data rate. In other words, we could design for a suboptimal frequency, but building headroom for process, voltage, and temperature (PVT), that is, we accept a maximum operating frequency, and design with a built-in margin.

Observe that condition (4) implies that we need to minimize the timing difference among signals arriving at any given stage ΔT_{pi}, since this directly impacts the maximum achievable operating frequency. One still needs to do "rough tuning" to equalize timing paths at each stage: add gates to shorter paths, and maintain a solid layout engineering that

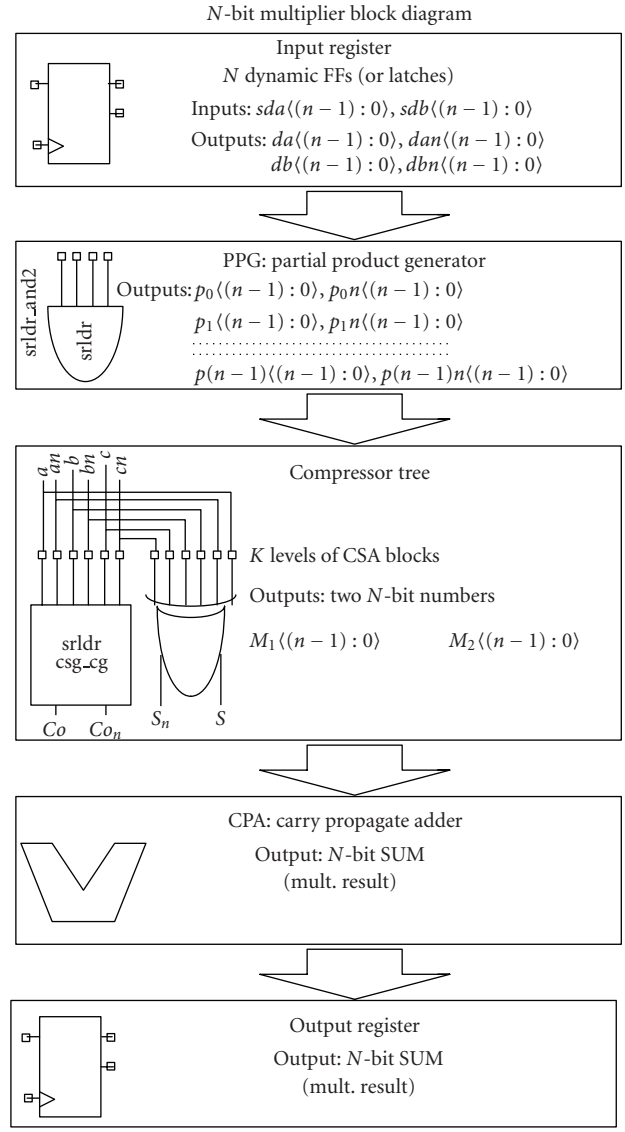

Figure 7: Multiplier block diagram.

looks into equalizing wire loads. The "fine-tuning" proposed in other implementations [4] may not add much in this case because of the "built-in" headroom by the gates. (Fine-tuning refers to the careful resizing of gates at transistor level, according to the needs of each signal path.)

The method described in this paper renders a stable circuit that may meet all specifications on the first approach, at the price of having added this extra margin in the gates themselves.

4. AN ILLUSTRATIVE EXAMPLE

4.1. Wave-pipelining parallel multiplier

Amultiplier was used to illustrate the concepts. It was implemented in a 1.2 V–0.18 μm CMOS process using a library of DRSRL-ID cells. The multiplier consists of three major blocks: the partial product generator (PPG), the partial

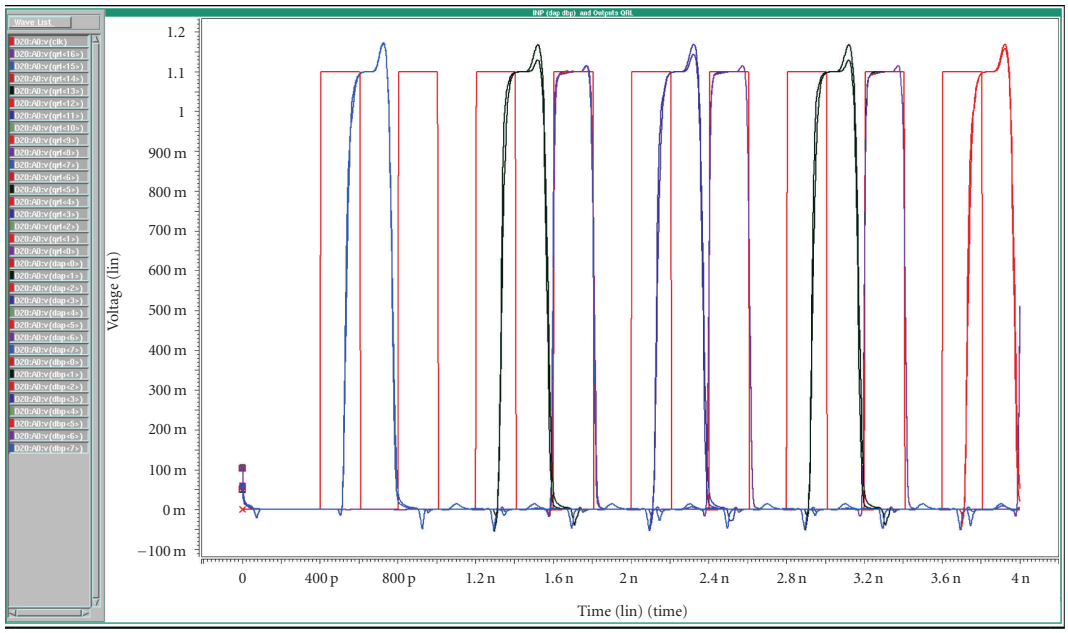

Figure 8: Advancing waves: *clk*, output: $qrl\langle 15:0\rangle$, and inputs dap and dbp.

product reducer (PPR), and an adder (*carry propagate* adder). Figure 7 shows a block diagram.

In the first stage, the partial products (PPs) are generated. Each PP is the product of each bit of the multiplier by every bit of the multiplicand. Thus, for an $n \times n$ multiplication, n PPs (n-bit wide) are generated. These PPs have to be added to obtain the final result.

The next stage is the partial product reducer (PPR), which reduces the n PPs of an n-bit multiplier to two, hence the name of *reducer*. This is the main block of the multiplier, which we have implemented as a Wallace tree using *carry-save adders* (CSA).Timing of the CSA cell has been adjusted so that the delay forward of both outputs (S, Co) is approximately the same. The two final elements are added by means of an adder to generate the final result. We have used the *carry look-ahead* structure proposed in [5], with a slight modification to control the fan-out and the loading at critical points [2].

This block by itself is essentially asynchronous. We have added input and output registers for timing analysis when the multiplier is inserted in a synchronized pipeline. Thereg-isters were implemented by a set of edge-triggered flip-flops. The output register must sample the final stage of the adder while the result pulses are available. For this wave-pipelining application, all paths have been equalized using "rough padding," that is, adding buffers to the shorter paths to get the same number of stages in all cases.

To make the delays through the longest and shortest paths through the logic as close as possible, we tried to tightly control the difference in arrival times of all signals connected to a certain stage. This implies not only the delay equalization of the different elements of logic at a given stage, but also the delay equalization of the interconnects between successive stages and controlling the total loading of a given intermediate driver. A careful layout plan is important, but in the present design there is a certain tolerance level for differences in arrival times. This is true as long as one can guarantee that all valid input pulses at a given stage will overlap long enough to generate the output pulse within the time frame of valid inputs for the next stage.

4.2. Simulation results and analysis

Results of spice simulation of 8-bit multiplier, implemented in a 0.18 μm CMOS process, running at 2.5 GHz data rate, are shown in Figure 8. It can be observed that as the pulse waves advance through the stages of the multiplier, the timing difference among signals at a given stage is minimal, so they conform to a coherent data wave. Here, the following signals are depicted: the global ideal clock *clk*, the output $qrl\langle 15:0\rangle$, together with inputs $dap\langle 7:0\rangle$ and $dbp\langle 7:0\rangle$. Since the inputs shown leave a clock cycle in between, where all input bits are made zero, for clarity, it is easy to observe two nonzero input patterns, before the first output is shown. The pattern shown corresponds to decimal products: (255×255), (3×3), (15×15), (3×3), and (63×63), alternated with (0×0) for power analysis.

In Figure 8, the global clock signal is shown as reference; it is the almost perfect rectangular wave. At the end of its first pulse, we see the overlapping input signals, entering the multiplier. As mentioned above, in this case, inputs remain at zero, during the next clock cycle, and so we do not observe any other activity. At the next clock cycle, the 2nd set of nonzero inputs is applied, but only within the following pulse of the clock we see the first set of outputs (very close together, and almost completely enclosed within that clock pulse).

A new set of nonzero inputs will enter the pipelined circuit, before the multiplier outputs the result corresponding to the second nonzero set of inputs. Actually, the multiplier is able to calculate a new result at each clock tick; we have just interleaved a set of zero-valued inputs in between for clarity.

The maximum timing difference among output bits occurs in this design between bits $rl\langle 0\rangle$ (early arrival) and $rl\langle 9\rangle$ (late arrival) and is approximately 74 picoseconds = ΔT_P. The maximum delay through the combinational logic T_P is 978 picoseconds. The delay through the FF (timing difference between $dap\langle 0\rangle$ and *ckira* is 52 picoseconds = t_d.

Here, T_{ck} = 400 picoseconds, that is, F_{ck} = 2.5 GHz. Setup time t_s = 20 picoseconds, and hold time t_h = 50 picoseconds.

Looking at signals between different stages in the compressor, we have measured the following:

(i) w_p = 210 picoseconds, t_{rec} = 89 picoseconds,
(ii) t_{df}; = 101 picoseconds, at the slowest path, and
(iii) $\tau = w_p + t_{\text{rec}} + t_{df}$:= (210 + 89 + 101) picoseconds = 400 picoseconds.

Since the input register is always sending pulses, by means of the direct outputs Q or the inverse ones Qn, then the power consumption is average no matter what pattern is presented at the inputs. Whenever single rail implementations are possible, there will be power savings, since pulses will be generated at gate outputs only if input signals validate the gate logic function, but gate outputs will remain at zero otherwise.

It is worth noting that as the width of the multiplier grows, the total latency increases, but the data throughput remains unchanged, as far as we can control the wire loading, since the maximum operating frequency depends on the cycle time of the gates.

5. CONCLUSIONS AND PROPOSED FUTURE WORK

Wave pipelining is especially suitable for designs that show a high degree of parallelism and regularity. If that were not the case, the circuit has to be first transformed to achieve such parallelism.The design shown provides a practical proof of the feasibility of using the proposed technique in many applications, where pipelining is suitable. Wave pipelining provides savings in area and timing, since all intermediate storage elements are removed from the circuit, saving also from the point of view of timing overhead. The use of self reset logic provides savings in power and area with respect to a comparable CMOS dynamic implementation, since clock distribution for dynamic gates is avoided, as shown in [2], where a comparison was made between two implementations of an adder: domino logic versus DRSRL-ID. The use of dual-rail self reset logic with input disable functionality (DRSRL-ID) has additional advantages, providing a fairly constant pulse width, and in so doing avoiding "pulse-width adjusting structures" [6]. It provides an additional tolerance in the design for differences in arrival times of signals at any stage. While such tolerance is built-in in the structure of the gate family, it comes at the price of adding to the total cycle time and affects the minimum clock period $T_{ck_{\min}}$ used to pump in new data into the circuit. The reduction in area and power savings, plus the simplified equalization mechanism due to the built-in tolerance, makes this approach suitable for many fast processing designs.

Additionally, if we use as the last stage an SR-latch, which will only be updated each time new data has arrived, then, we are making the last stage "static," and in so doing, we can reduce the operating frequency as we need to interface with the next stage (moving the design from a k-wave mode to a single wave, if so needed). At the same time, we must maintain constraint (3) on the width of the input pulse to the first stage implemented with DRSRL-ID. The recommended approach would be to use a pulse generator, which will generate one pulse at the valid input clock transition.

The basic DRSRL-ID is suitable for structures with feedback, and this is an area we will investigate further.There is also special interest in asynchronous circuit applications. The DRSRL-ID application shown here uses the simplest protocol: "just sending data" and sacrifices elasticity for higher throughput. Many variations are possible according to circuit needs.

ACKNOWLEDGMENT

M. E. Litvin is an IEEE regular member and S. Mourad is an IEEE Fellow. They thank Dr. Fabian Klass for his invaluable comments and advise during the research period.

REFERENCES

[1] M. E. Litvin, "Wave pipelining with self reset logic," Doctoral dissertation, Santa Clara University, Santa Clara, Calif, USA, 2005.
[2] M. E. Litvin and S. Mourad, "Self-reset logic for fast arithmetic applications," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 4, pp. 462–475, 2005.
[3] M. E. Litvin and S. Mourad, "Wave pipelining with self reset logic," in *Proceedings of IEEE International Conference on Electronic Circuits & Systems (ICECS '06)*, Nice, France, December 2006.
[4] E. F. Klass, "Wave pipelining theoretical & practical issues in CMOS," Doctoral dissertation, Stanford Univrsity, Stanford, Calif, USA, 1994.
[5] W. K. C. Lam, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Valid clocking in wavepipelined circuits," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '92)*, pp. 518–525, Santa Clara, Calif, USA, November 1992.
[6] L. Wentai, C. T. Gray, D. Fan, W. J. Farlow, T. A. Hughes, and R. K. Cavin, "250-MHz wave pipelined adder in 2-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 9, pp. 1117–1128, 1994.