Wave Pipelining using Self Reset Logic

Miguel E. Litvin
Electrical Engineering Department
Santa Clara University -500 El Camino Real –
Santa Clara, CA 95053
melitvin@gmail.com

Samiha Mourad, William and Janice Terry Professor
Electrical Engineering Department
Santa Clara University -500 El Camino Real –
Santa Clara, CA 95053
smourad@scu.edu

A novel design approach combining Wave Pipelining and Self Reset Logic provides an elegant solution at high speed data throughput with significant savings in power and area as compared with other dynamic CMOS Logic implementations

I. 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, hold 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 Fig. 1.

![Fig.1: Basic Wave Pipelining circuit](image)

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 Fig. 2. In the next section we discuss its operation. Section III describes the timing constraints that apply when designing Wave Pipelining circuits with DRSRD-ID. Section IV presents an application circuit, and the final Section presents conclusions and future work.

II Self Reset Logic

A DRSRD-ID buffer-inverter cell is shown in Fig. 2. The gate will generate an output pulse at the direct or the inverse output 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 \( \tau \), defined below.

\[
\tau = t_{df} + w_p + t_{rec}
\]

Where:

- \( t_{df} \): Data delay forward. The time from the leading edge of the input data transition that validates \( F \) or \( F_N \) to the leading edge of the pulse at the output.
- \( w_p \): Width of the output pulse, and
- \( t_{rec} \): Recovery time. 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:

\[
d_{fovlmin} \geq t_{df}
\]

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 fanout is equal or less than 8 (for the gate families we have worked with). It is also
independent of the input pulses, while they satisfy condition (2). Recovery time \( t_{\text{rec}} \) and delay forward \( t_{\text{df}} \) can also be made equal for a family of gates. Then the \textit{Cycle time} \( \tau \) 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. For a complete description and characterization we refer the reader to [1,2].

III WAVE PIPELINING WITH DRSRL-ID

The Wave Pipelined 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 Fig. 1. This requires careful selection of the 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.
- \( c_{\text{k}} \): Global clock.
- \( c_{\text{kir}} \): Clock at input register.
- \( c_{\text{out}} \): Clock at output register.
- \( T_{\text{L}} \): Total latency: Time elapsed from launching a data wave from the input register until the corresponding result arrives at the output register.
- \( T_{\text{p}} \): Maximum delay through the combinational logic.
- \( \Delta T_{\text{p}} \): Maximum path delay difference through the combinational logic.
- \( \Delta \): \( T_{\text{L}} \mod c_{\text{k}} \) = \( \Delta_{\text{o}} - \Delta_{\text{i}} \): Constructive skew. (Phase shift between the clocks that control the launching and receiving registers).
- \( t_{\text{d}} \): Register Clock-to-Q delay.
- \( t_{\text{s}} \): Register setup time.
- \( t_{\text{h}} \): Register hold time.
- \( t_{\text{sk}} \): Uncontrollable clock skew.

The width of the output pulse of the input register \( w_{\text{pIR}} \) must satisfy (3):

\[
 t_{\text{df}} \leq w_{\text{pIR}} < c_{\text{k}}
\]

The timing conditions are:

\[
w_{p} > \Delta T_{p} + t_{\text{df}} + t_{\text{sk}}
\]

where \( \Delta T_{p} \) is the worst timing difference expected at any given stage. Then:

\[
 T_{\text{ck}} \geq w_{p} + t_{\text{rec}}
\]

Considering also the delay forward \( t_{\text{df}} \):

\[
 T_{\text{ck}} \geq w_{p} + t_{\text{rec}} + t_{\text{df}}
\]

That is:

\[
 T_{\text{ck}} \geq \tau
\]

The total latency is:

\[
 T_{\text{L}} = k \cdot T_{\text{ck}} + \Delta_{\text{o}}
\]

Analyzing the situation corresponding to a “late arriving” pulse versus an “early arriving” one, as shown in Fig. 3, one can demonstrate that for Wave Pipelining with DRSRL-ID:

\[
 T_{\text{ck}} \geq ( t_{\text{d}} + T_{p} + t_{\text{s}} + t_{\text{sk}} - \Delta ) / k
\]

\[
w_{p} > \Delta T_{p} + t_{\text{s}} + t_{\text{h}} + 2t_{\text{sk}}
\]

Comparing with CMOS implementation, as shown by [3], [4]: in that case, the conditions for safe pipelining include (11) & (12), below:

\[
 T_{\text{ck}} \geq \Delta T_{p} + t_{\text{s}} + t_{\text{h}} + 2t_{\text{sk}}
\]

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

\[
 (T_{p} + t_{\text{d}} + t_{\text{s}} - \Delta) / k \leq T_{\text{ck}} < (T_{p} - \Delta T_{p} + t_{\text{d}} + t_{\text{s}} - \Delta) / (k-1)
\]

The conditions on \( w_{p} \) in DRSRL-ID are similar to the conditions on \( T_{\text{ck}} \) for CMOS WP, rendering a theoretic lower data rate. In other words we could design for a sub-optimal frequency, but building headroom for Process, Voltage and Temperature (P.V.T.).

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 looks into equalizing wire loads. The “fine-tuning” proposed in other implementations, may not add much in this case because of the “built-in” headroom by the gates.

The method 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.

IV. AN ILLUSTRATIVE EXAMPLE

A. Wave pipelined parallel multiplier

A multiplier was used to illustrate the concepts. It was implemented in a 1.2V-0.18um CMOS process using a library of DRSRL-ID cells. The multiplier consists of three major blocks: Partial Products Generator (PPG), the Partial Product Reducer (PPR), and an adder (ADDER). In the first stage the partial products (PP) are generated. Each PP is the
product of each bit of the multiplier by every bit of the multiplicand. Thus for an nxn multiplication, n PP 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 by [5], with a slight modification to control the fanout 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.

B. Simulation Results and Analysis

Results of spice simulation of the multiplier, implemented in a 0.18µm CMOS process, running at 2.5 GHz data rate, are shown in Fig. 4. 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 a coherent data wave. Here the following signals are depicted: the global ideal clock clk, the output qrl<15:0>, together with inputs dap<7:0> and dbp<7:0>. 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 non-zero input patterns, before the first output is shown. The pattern shown corresponds to decimal products: (255x255), (3x3), (15x15), (3x3), (63x63), alternated with (0x0) for power analysis.

The maximum timing difference among output bits occurs in this design between bits r<1<0> (early arrival) and r<1<0> (late arrival) and is approximately 74ps= ΔTp. The maximum delay through the combinational logic Tp is 978ps. The delay through the FF (timing difference between dap<0> and ckira) is 52ps=tq

Here Tck=400ps, that is Fck=2.5GHz. Setup time ts = 20ps and hold time th = 50ps. Looking at signals between different stages in the Compressor, we have measured: wP =210ps, trec=89ps, tdf=101ps, at the slowest path.

\[ τ = w_P + t_{rec} + t_{df} = (210 + 89 + 101)ps = 400ps \]

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.

V. 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 was shown in [2], where a comparison was made between two implementations of an Adder: Domino Logic vs. 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 difference 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 Tckmin 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

The authors thank Dr. Fabian Klass for his invaluable comments and advise during the research period.

REFERENCES

[1] [Litvin-05] Miguel Litvin. “Wave Pipelining with Self Reset Logic”. Doctoral Dissertation, Santa Clara University, CA, 2005.
[2] [Litvin-Mourad-05] Miguel Litvin and Samiha Mourad. “Self Reset Logic for Fast Arithmetic Applications”. IEEE Transactions on VLSI Systems, Vol. 13, No. 4, April 2005.
[3] [Klass-94] E.F. Klass, “Wave Pipelining Theoretical & Practical Issues in CMOS”. Doctoral Dissertation, Stanford, CA, 1994.
[4] [Lam-92] W. Lam, Brayton and Sangiovanni-Vincentelli, “Valid Clocking in Wavepipelined Circuits Proceedings of ICCAD ‘92, Santa Clara, CA, Nov. 1992, pp. 518-525.
[5] [Liu-94] Wentai Liu, T. Gray, D. Fan et al, “A 250-MHz Wave Pipelined Adder in 2-um CMOS” IEEE Journal of Solid State Circuits, Vol. 29, No.9, Sep. 1994.
[6] [Hwang-99] Hwang, Joshi, Henkels, “A 500-MHz, 32-Word x 64-Bit, Eight-Port Self-Resetting CMOS Register File”, IEEE Journal of Solid State Circ. vol 34, No 1, Jan 1999.
[7] [Wendell-95] Dennis Wendell, “Reset Logic Circuit and Method”. U.S. Patent 5430399.
[8] [Heald-93] Raymond Heald, John Holst, “A 6ns Cycle, 256-Kb Cache Memory and Memory Management Unit”, IEEE Journal of Solid State Circuits, Vol. 28, No. 11, Nov. 1993.
[9] [Heald-98] Raymond Heald, Ken Shin, et al. “64-KByte Sum-Addressed-Memory Cache with 1.6-ns Cycle and 2.6-ns Latency”, IEEE Journal of Solid State Circuits, Vol. 33, No. 11, Nov. 1998.
[10] Narayanan-96] Vinod Narayanan, Barbara Chappell, Bruce Fleischer, “Static Timing Analysis for Self Resetting Circuits”. ICCAD 1996.