MODIFIED SRCMOS CELL FOR HIGH-THROUGHPUT WAVE-PIPELINED ARITHMETIC UNITS

Tero Säntti and Jouni Isoaho

{teansa | jisoaho}@utu.fi
phone: +358 2 333 6956 fax: +358 2 333 6950
Laboratory of Electronics and Information Technology
University of Turku Lemminkäisenkatu 14-18 20520 Turku Finland

ABSTRACT

In this paper a modified basic cell for wave-pipelines is proposed. The cell is self resetting and has complementary outputs. Simulations of the cell demonstrate that delay variations for all input combinations are small, and the cell's sensitivity to pulse length variation is reduced. 8x8 and 16x16 -bit multipliers are designed using 0.35μm 2.5V CMOS technology. The proposed units display a cycle time of 620 ps, corresponding to maximum operation frequency of 1.6 GHz.

1. INTRODUCTION

High speed arithmetic units are required to meet the demands of modern signal processing and multimedia applications. The most dominant method to achieve the needed performance is the use of pipelines. Since most designs are based on a clocked approach, it is quite easy to just add registers in long data paths. This has a few setbacks, such as penalties in area, latency and power consumption. Even the throughput of a conventional pipeline is less than optimal, because the added registers require setup and hold time, and introduce delay for every stage. Power consumption rises due to clocking of the registers. Also partitioning may cause some stages to be considerably faster than others. This results in non-optimal clockrate. To resolve this dilemma, wave-pipelining comes to rescue. Eliminating the registers reduces overheads in all problem areas. Unfortunately it also introduces new problems, such as difficulty of design and rather unstable operation. To alleviate the latter, a modified SRCMOS cell is presented. The new cell also partly assists in design, as more stable subcircuits leave more headroom for design and variations in process, voltage and temperature.

2. WAVE-PIPELINING CONCEPT

Wave-pipelining was first introduced by L. Cotten [3] back in 1969. He called it *maximum rate pipelining*. For a while it was more of a beautiful idea than anything to be realized. Modern VLSI technology has raised the interest to study the possibilities of this method. Several different schemes have been tried during the past few years, but none of them has become a standard method. All these schemes share the same principles of operation, even if some are done using bipolar transistors instead of MOS devices. The main idea behind all this is to remove internal registers from pipelines to obtain an attractive group of benefits, including decreased latency and cycle time as well as reduced area and power consumption.

Registers are used to store data between stages in conventional pipelines. In a wave-pipeline every stage holds the data for a short time, while the next stage starts to calculate the outputs to the stage after that. After the next stage has used the inputs, the previous one can be reset. This of course places requirements for the stages. They have to take all the inputs at the same time, and then store the data internally. Also all the outputs have to be completed at the same time. These rules hold for a single bit slice and also for a full width pipeline stage. To get maximum performance, each stage should have an equal delay.

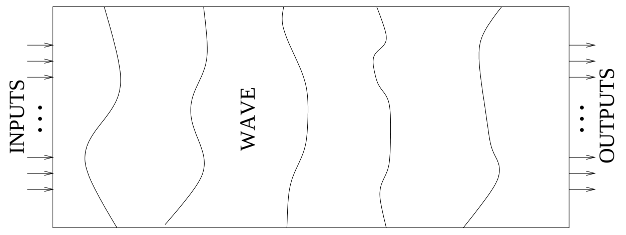

Figure 1: Data "waves" in a wave-pipelined logic block.

Wave-pipelines are internally asynchronous, as data travelling through the pipeline triggers the next active element. They can, however, also be used in synchronous designs quite easily since the inputs can be latched with a clocked latch. This results in externally visible cycle time equal to the cycle time of the latch. Outputs are latched at the next clock after arriving to the end of the pipeline. No data corruption can occur, if each stage is able to pass the data to the next stage before new data arrives. The best use of all the advantages comes when wave-pipelines are used in an asynchronous environment. Even then there are certain important properties to be considered. If a given asynchronous system is based on the principles of Sutherland's micropipelines [7], it is elastic in nature. This is not true for wave-pipelines. Once data is fed into the pipeline, there is no way of stopping it until the end, and if the data is not read in time, it gets overrun by the next data coming down the pipe.

Numerous studies, e.g. [1], [4] and [5], have shown mathematically, that wave-pipelining is better than conventional pipelining, at least in theory. Wave-pipelining is not suitable for components in which datapaths are strongly mismatched in length, delay or fan-out. Most arithmetic operations have been transformed into a form comprising small, identical basic elements thus enabling them to be wave-pipelined efficiently. Adders and multipliers are common examples of this. In a basic ripple carry adder there is no possibility of wave-pipelining due to the carry signal moving horizontally compared to the data moving vertically. If an adder is composed of smaller carry look ahead (CLA) elements, it can be pipelined using conventional pipelining. Still there is a notable difference in the delay of each bit in any given CLA element, rendering the structure incompatible with the requirements for wave-pipelining. The Brent-Kung adder scheme [2] can be implemented in two ways. The standard way has a quite regular layout with rather messy wiring and fan-outs. The other way is an array structure having more regular wiring and fan-out. In a wave-pipeline only fan-out and regularity of wiring are important, especially since wires are becoming the most dominant source of delay in designs.

3. BASIC WAVE-PIPELINE CELL

The basic cell described in this article is based on the work of O. Hauck [6]. He introduced the use of self resetting CMOS (SR-CMOS) to wave-pipelines. The cell is shown in Figure 2. The

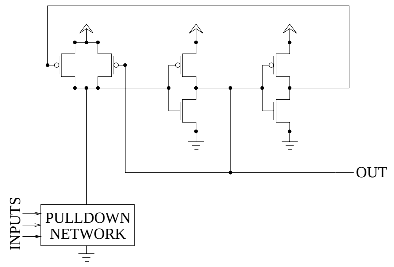

Figure 2: Basic cell as proposed by O.Hauck.

cell operates quite simply. If the pulldown network forms a path to ground, the main node above it is pulled down. The righthand side PMOS device above it is initially open, but since it is only a small keeper transistor the current does not rise too high. Once the main node is close to ground the first inverter turns. It controls the keeper transistor and turns it off to minimise the current. It also sends the same signal to the output. When this first inverter has turned, it starts to turn the second one as well. The second inverter is there merely to provide control for the resetting transistor, which is wide, in order to ensure resetting of the cell. At around this time the inputs should have gone low, and the cell returns to initial conditions.

To improve the reliability of the system, some modifications were made. Some modifications were also made in order to get complementary outputs, since they were needed in the multiplier. The modified cell is shown in Figure 3. The idea behind both of the cells is the same: to get as stable characteristics as possible. The original cell provides almost constant pulse length and is fairly easy to implement for different fan-outs. The pulses could, however, be shortened, if the inputs were to drop low before the internal self resetting comes active. Also too long inputs would cause an extensive leak current and possibly increase the length of the output pulse, causing the same problems at the next stage. In order to counter these hazards, and to bring in complementary outputs, a modified cell is proposed. The modified cell requires seven

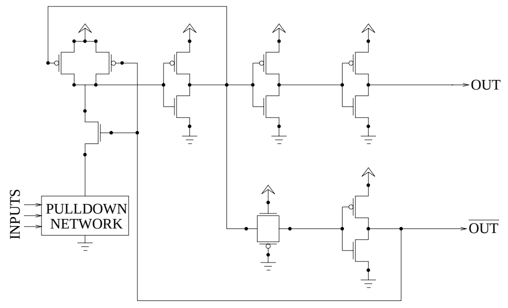

Figure 3: Modified cell.

transistors more than the original one, and has a slightly higher latency. These drawbacks are acknowledged as an inevitable trade-off. Increased reliability comes mainly from the NMOS device added over the pulldown network. It is controlled by the same signal as the resetting transistor on the righthand side above it. Together they isolate the main node from pulldown network and connect it to the operating voltage. As a result the output pulses are cut to be equal in length, and slightly longer input pulses will not cause extensive leaking. They also render the cell insensitive to new inputs for the duration of the output pulse. Since the control for this is taken from the inverted output instead of the output of the second inverter, the pulselength is increased by the delay of the transmission gate. This further improves stability and reliability of the cell. Note that the length of the pulse does not affect latency. It does, however, decrease the number of concurrent waves, or virtual pipeline stages, in the pipeline.

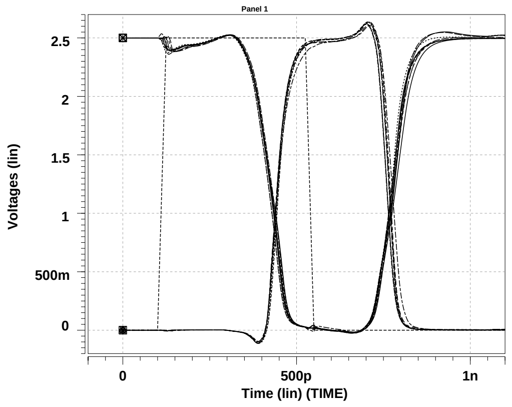

Figure 4: Eyediagram of a full adder cell.

As an example a full adder cell was simulated. It can be seen from the results that all the combinations causing a pulse either to the sum or carry lines are well matched. This keeps delay variations small, thus increasing efficiency and reliability. It also provides for a longer latching window at the end of the pipeline, since all signals are valid for longer time. The length of a pulse is about 310 ps. That and the dead time add up to a minimum cycle time of 620 ps, giving a maximum frequency of 1.6 GHz.

Sizing the transistors in the cell is reasonably easy, in the same order of magnitude as in the original one. This new cell can be configured to display wide variety of different characteristics. The length of the pulse can be controlled easily, as well as the delay. Longer delays are sometimes required to match different datapaths. These can be acquired by resizing the NMOS or the PMOS in the transmission gate. The really demanding part is the designing and sizing the pulldown network. All parallel paths must have the same conductance while open, and only one path at a time is allowed to be open. Otherwise there will be a data dependent mismatch in delay, which is not desirable for reliable operation. The proposed cell helps in the design of the pulldown network, since the designer has to match only the beginnings of the pulses, while the cell cuts the ends at the same time.

4. CASE STUDY: WAVE-PIPELINED MULTIPLIERS

A multiplier was selected to serve as an example because it is algorithmically compatible with wave-pipelining, and it is widely used in DSP and telecommunication applications. The chosen algorithm also includes an adder, which can be implemented independently.

Multiplication is divided into three parts, namely the AND-stage, the partial product reduction tree and the adder. The first stage has only one wave at any given time whereas the others have several, depending on the width of the multiplier. In the AND-stage an AND operation is performed on all input bit combinations. This produces a matrix of NxN bits, forming the product. These bits are then fed into the partial product reduction tree. The tree resembles a Wallace tree [8] in structure, but it has been optimised slightly to reduce the number of cells. For the final part, the adder, a Brent-Kung adder (BKA) was chosen. Advantages of BKA were briefly discussed earlier. Instead of the standard BKA architecture the array structure was chosen.

8x8-bit and 16x16-bit multipliers were constructed and simulated using starHSpice from Avant! corporation and 0.35μm CMOS technology with level 49 transistor models. In Figure 5 the data waves in the 8x8-bit multiplier are shown. Here each signal is an intermediate signal between stages. As can be seen, the stages are almost perfectly of the same length even though the logic varies in the three before mentioned sections.

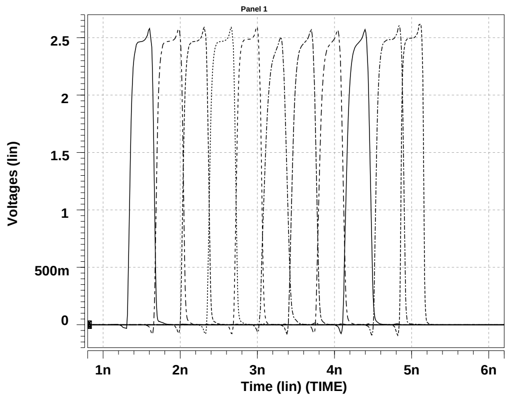

Figure 5: The data waves in the 8x8-bit multiplier.

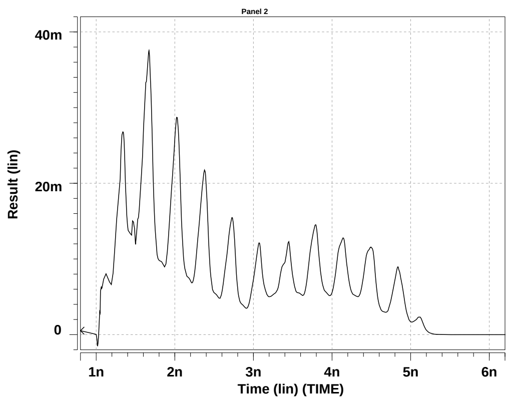

Figure 6: The current behaviour of the 8x8-bit multiplier calculating a single operation.

The minimum cycle time was found to be 620 ps resulting in operation at 1.6 GHz, as predicted by simulations of a single cell. The latency of this multiplier was 3.9 ns. For a single operation, with all inputs high to yield the worst case current, the maximum current spike reached 38 mA while the average was 14 mA. The effects of the stages can be seen in the current shown in Figure 6. The current is highest in the first stage of the partial product reduction tree. The current keeps on getting lower, until the beginning of the adder section. The current of the adder remains relatively stable as the signal moves across the stages. The power consumption is data dependent, as for all units created using this cell. If the input combination doesn't form a path to ground, the rest of the cell remains unchanged, using no power.

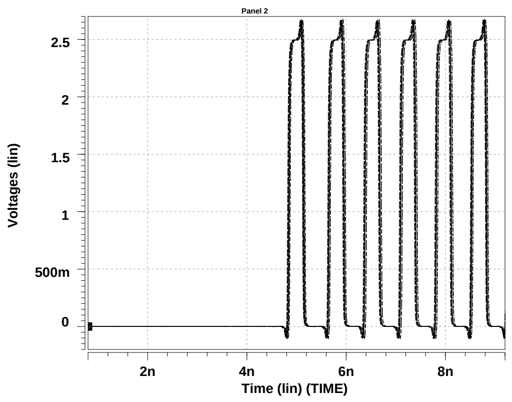

Figure 7: The outputs of the 8x8-bit multiplier at 1.5GHz.

In Figures 7 and 8 the multipliers were driven at the rate of 1.5 GHz. In the figure showing the currents the lower curve is associated with the 8x8-bit multiplier. Fully loading the pipeline raised the highest current spike to 79 mA and the average to 60 mA. The same simulations were run with the 16x16-bit version of the multiplier. In Figure 9 the 16x16-bit multiplier is driven at 1.5 GHz. This shows the latency to be 5 ns, as the first input is applied to the multiplier at 1 ns. The current characteristics are shown in the upper curve of Figure 8. The maximum current spike is 280 mA. Again all inputs were held high to get the worst case current. As stated previously, the power usage is data dependent.

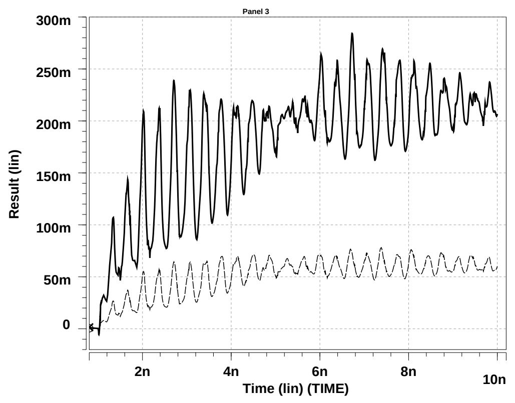

Figure 8: The current behaviour of the 8x8 and the 16x16 -bit multipliers at 1.5GHz, lower and upper respectively.

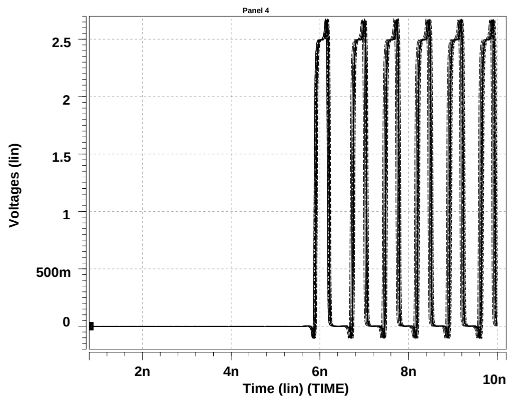

Figure 9: The outputs of the 16x16-bit multiplier at 1.5GHz.

Wider multipliers are being studied. Latencies for wider multipliers were estimated from the delay of one stage and the number of stages required. Estimations are shown in Table 1. The values that are based on simulations are marked with a '*'. The cycle time remains constant regardless of used width. Some latches may be required to resynchronise the data waves in longer pipelines. The optimal placing of these latches is such that all partial pipelines separated by latches are of equal length. The insertion of these latches also increases the latency by their delay. The latches can be designed using similar cells as for the logic, using a simple pull-down network implementing an AND-function between each data bit and the control signal. Thus the output signal is in the form required by the surrounding cells.

5. CONCLUSIONS

The proposed basic cell exhibits good performance and stability with relatively low overheads in area, latency and power consumption. 16x16-bit and 8x8-bit multipliers were simulated, and both of the multipliers run at the maximum frequency of 1.6 GHz. The 16x16-bit multiplier is to be processed to a fully working chip within a year.

width	stages	waves	latency
8	11	5.5	3.9 ns *
12	13	6.5	4.6 ns
16	14	7.0	5.0 ns *
24	16	8.0	5.7 ns
32	17	8.5	6.1 ns **
48	18	9.0	6.4 ns
64	20	10.0	7.1 ns

Table 1: Estimated latencies. Latencies marked with a '*' are obtained from simulations and the latency of 32-bit multiplier (**) is obtained from a preliminary simulation.

Since designing wave-pipelines requires a lot of manual labour, the next research goal is writing dedicated design software. Using cells proposed here it is possible to generate netlists of logic blocks. The netlists can then be passed to place and route software capable of delay balancing the nets. The only truly manual task left is balancing the pulldown networks and designing a few basic cells for different fan-outs. A tool like this would make wave-pipelines a cost efficient way to implement high-performance arithmetic structures.

6. REFERENCES

[1] W. Burleson, M. Ciesielski, F. Klass and W. Liu, "Wave-Pipelining: A Tutorial and Research Survey", *IEEE Transactions on VLSI Systems*, vol. 6, no. 3, pp. 464-474, September 1998

[2] R. Brent and H.T. Kung, "A Regular Layout for Parallel Adders", *IEEE Transaction on Computers*, vol. C-31, no. 3, pp. 260-264, March 1982

[3] L. Cotten, "Maximum rate pipelined systems", in *Proc. AFIPS Spring Joint Comput. Conf.*, 1969

[4] C. Gray, W. Liu and R. Cavin, "Timing Constraints for Wave-Pipelined Systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 987-1004, August 1994

[5] O. Hauck and S. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths", *IEEE International Conference on Electronics, Circuits and Systems*, pp. 1.283-1.286, September 1998

[6] O. Hauck and S. Huss, "Circuit Design for SRCMOS Asynchronous Wave Pipelines", *4th Asynchronous Circuit Design Workshop* , February 2000

[7] I. Sutherland, "Micropipelines", The 1988 Turing Award Lecture, *Communication of the ACM*, vol. 32, no. 6, pp. 720-738, June 1989

[8] C. Wallace, "A Suggestion for a Fast Multiplier", *IEEE transaction on electronic computers*, vol. EC-13, pp. 14-17, February 1964