IEEE International Symposium on Circuits and Systems (ISCAS´2001), ISBN: 0-7803-6685-9, S. IV 144 - 147, Sydney, Mai 2001

DYNAMIC SINGLE PHASE LOGIC WITH SELF-TIMED STAGES FOR POWER REDUCTION IN PIPELINE CIRCUIT DESIGNS

F. Grassert, D. Timmermann

Institute for Applied Microelectronics and Computer Science
Department of Electrical Engineering and Information Technology
University of Rostock
Richard-Wagner-Str. 31, D-18119 Rostock, Germany

ABSTRACT

True single phase clock logic techniques, e.g. with alternating arranged N- and P-logic cells, yield easy to design circuits with standard cells and high speed potential. The disadvantages are a difficult clock tree design and high power consumption. To realize every logic function, dual rail or differential styles are chosen which increase clock load. This paper presents a method to speed up dynamic single clock circuits. The advantage of asynchronous logic is that the critical path delay is the sum of only the evaluation times of the single logic blocks without wasting time for waiting, latches, or redundant logic. Therefore, this work assembles small asynchronous chains of dynamic logic blocks into one period of the global clock to minimize the unused time per clock cycle (AC-TSPC). However, the synchronous single phase clocking scheme is maintained. The advantages of this method are shorter latencies for calculations, power reduction by smaller clock trees and no need for latches, and a simpler clock distribution network due to increased clock skew tolerance. The results of the simulations of an 8x8 bit multiplier in TSPC and in AC-TSPC show an enhancement in power-reduction of 40% for the logic and of 89% for the clock tree with a latency reduction of 40% and more in comparison with TSPC.

1. INTRODUCTION

The true single phase clocking (TSPC) (figure 1) scheme introduced in [6] yields fast dynamic logic with a single clock tree. But the speed of this logic style calls for very fast clocks with low skew. Therefore, clock distribution became a nontrivial problem. Skew tolerant logic styles can greatly reduce the cost of clock trees. Other disadvantages of the TSPC style in [6] are time consuming latches and evaluation times of the P-logic trees which are much bigger than the evaluation times of the N-logic. For fastest designs the minimal clocking frequency is aligned to the evaluation time of the slowest N-logic block. As a result the P-block is nearly unusable for logic evaluation to prevent a slow-down of cycle time.

Domino logic (DOMINO) is a simple realization of the dynamic idea but requires two clocking signals in minimum (figure 1). However, DOMINO logic offers big advantages in speed. Harris et.al. [4] published a skew tolerant realization where a scheme of overlapping clocking signals requires no latches and decreases sensitivity to clock slopes. Propagation delay of the logic path is the sum of the evaluation times of the single logic blocks only plus inverter or static logic delay.

A different approach is the use of dynamic logic styles in asynchronous designs. In [5], a divider using a ring structure was realized and yields no delay in addition to the evaluation time. Such self-timing techniques require completion signals and therefore, differential logic styles or dual rail realizations are used. In [7], Cascode Voltage Switch Logic (DCVSL) was introduced as a differential logic (figure 1) which built the starting point for several new logic styles. It was derived from two DOMINO logic blocks with complementary functions, merged logic trees, and a shared clocked N-transistor. Complex logic functions can be realized in one differential gate with less transistors. The recently published proposals [8,9] demonstrate the ongoing interest in dynamic differential logic.

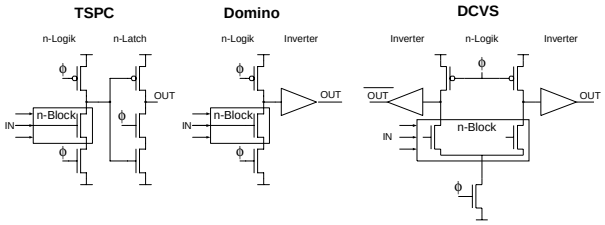

Figure 1. TSPC, Domino, and DCVS logic; TSPC has alternating P-logic and P-latches; DCVSL merges the complementary logic function in the n-block; Domino and DCVS need more than one clock signal

Dynamic logic styles often require dual rail structures to realize every logic function. The difference between dual rail and differential structures is that any single rail logic can be converted to dual rail logic by independently building the complementary logic part. In a differential logic style the complementary functions are merged together and are not independent.

In this paper we present a novel technique which can be implemented in a single clock design. However, subsequent gates are connected in an asynchronous style. This results in shorter evaluation time due to missing delay between stages and due to missing latches. As an example, DOMINO logic is used in a complementary way to generate completion signals for self-timing. An 8x8 multiplier circuit shows the potential of this technique with respect to speed and power-reduction. The main focus for our implementation is put on a design with simple logic cells. Therefore, all transistors were simulated with minimal widths and no special deviations were made. In section 2, the basics of dual rail logic will be explained. The realization of asynchronous logic is introduced in section 3. In section 4 the design of an 8x8 multiplier is explained and the results are discussed in section 5. Section 6 presents the conclusions.

2. DYNAMIC LOGIC FOR DIFFERENTIAL USAGE

Dynamic logic styles as depicted in figure 1 employ the same principle of precharge and evaluation phase. These phases are dictated by the clocking signal. The following explanation is valid for an N-logic block. During precharge (clock low), the P-transistor connects the output-node of the dynamic gate to VDD thus charging the output capacitance to high. In the evaluation phase (clock high) the clocked N-transistor is turned on. Depending upon the input values of the logic tree the output node can be discharged to ground. Therefore, the evaluation phase realizes the logic function. A P-logic block works in a complementary manner with clock signal being high during precharge and vice versa.

TSPC logic (figure 1) uses alternating dynamic N-logic and P-logic blocks connected to N- or P-latches, respectively. In one clock cycle both N- and P-blocks evaluate and precharge. The speed is determined by the slowest action, i.e., evaluation of logic-block and latch (P or N) or precharge of the internal nodes (to high or low).

DOMINO logic uses N-logic blocks only with subsequent static inverting logic, e.g., an inverter. If two sequential gates (dynamic and static part) use the same clock the evaluation proceeds to the first gate and then immediately to the second one. During precharge, each output of the inverting static logic goes low. Therefore, subsequent logic trees can not connect to ground. In two cascaded gates with different clocks the second gate accepts the output signal of the first gate if both evaluation phases overlap. The evaluation phase of the first gate has to hold until the internal node of the second gate is fully settled. Otherwise, information is lost. The precharge of the previous gate does not effect the settled outputs of the next one because of high to low transition only at the inputs.

DCVSL, as an example of a differential logic structure, always evaluates two complementary output values and operates like DOMINO. At the end of evaluation the difference at the outputs can be used for completion detection. The main benefit of differential logic compared with simple dual rail is that the logic tree can be partially shared. An implementation of complex logic functions with less transistors is feasible. However, if two independent domino stages or other single rail logic styles are used instead the same functionality can be realized.

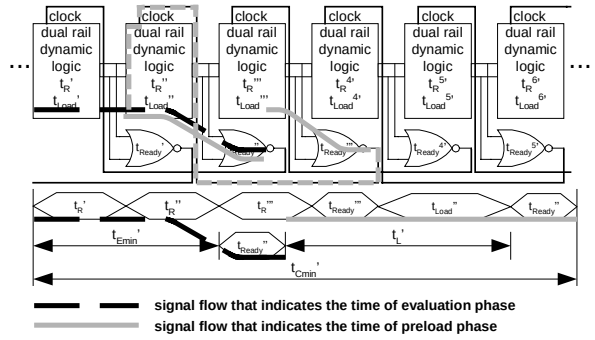

Figure 2. Scheme for self-timing of dynamic dual rail logic; the thick lines show the direction of the signals which define the duration of the phases

3. CONDITIONS OF ASYNCHRONOUS LOGIC

3.1 Basics for Self-timing

Dynamic logic with a complementary structure can be arranged in an asynchronous way. The completion signal of one gate can inform the previous gate to switch to precharge phase. For a dual rail structure with DOMINO or DCVSL a completion signal from a NOR can be directly used as the clock signal for the previous gate. Figure 2 depicts this structure. The calculations start with the first gate and propagate the chain without stopping. A ring structure is examined in [5]. The proper behavior of such self-timed structures is based on the following formulas. The cycle of one gate starts with the evaluation and ends when the next evaluation phase begins. Minimum cycle time occurs when the change of the clock signal to the evaluation phase matches the arrival of the inputs. Therefore, the minimum cycle time starts with the arrival of the inputs with the assumption that the gate is already in the evaluation phase. The following formula calculates this minimal cycle time t_{Cmin}':

$$t_{Cmin}' = t_R' + t_R'' + t_R''' + t_{Ready}''' + t_{Load}'' + t_{Ready}'',$$

where t_R', t_R'', and t_R''' are the time of the logic calculation up to the moment the outputs are settled, t_{Ready}''' and t_{Ready}'' are the slope times of the completion signals from the gates respectively and t_{Load}'' is the time for precharging the output signals of gate two (figure 2). The signal propagates through all three gates. The completion signal of the third gate sets the clock signal of gate two in the precharge phase. After precharge, the completion signal of gate two sets the clock signal of the first gate back to evaluation. Figure 2 shows these signal flows which occur during the evaluation and the precharge phases. The minimum time of the evaluation phase t_{Emin}' results from:

$$t_{Emin}' = t_R' + t_R''.$$

The time t_{Ready}'' of the slope of the completion signal is completely excluded, because the exact finish of evaluation is unpredictable. But calculations with a shorter evaluation phase will not risk the functionality. The time of the precharge phase then results from:

$$t_L' = t_{Cmin}' - t_{Emin}' - t_{Ready}''' - t_{Ready}'' = t_R''' + t_{Load}''.$$

The minimum evaluation phase is subtracted from the minimum cycle time. Again, the slope times of the completion signals are not taken into account. If the inputs arrive after the evaluation phase has started it does not effect the duration of the precharge phase but the cycle time and the duration of the evaluation phase will increase. If the inputs arrive earlier a delay happens. To avoid this problem, a delayed arrival of the inputs at the first gates is advantageous.

3.2 Cooperation with Globally Synchronous Systems

The main goal of this work is to arrange small asynchronous chains of logic in a synchronous design (AC-TSPC) (figure 3). A single phase global clock with same duration of high and low phase clocks the first gate. The following gates are connected via the self-timing scheme. The last gate derives its clock from a completion signal of the following first gates. In case the runtime of the sequence is nearly half the clock cycle time, the end of the precharge phase of this gate will delay. Therefore, the runtime

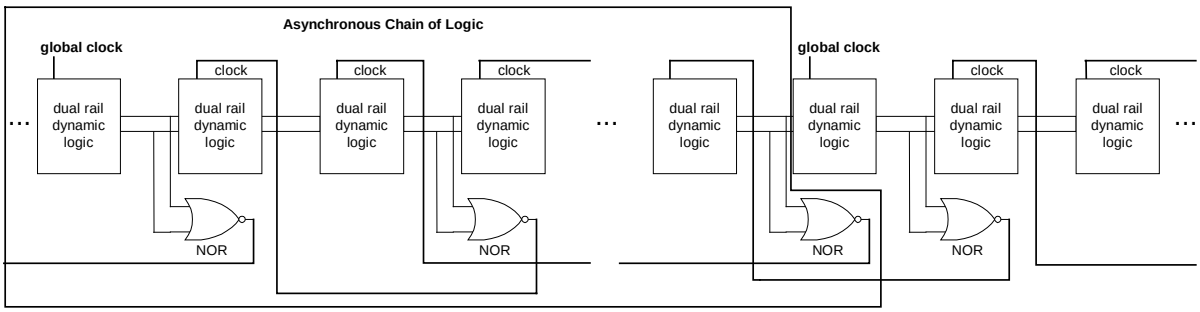

Figure 3. Scheme for implementation of an asynchronous block in a synchronous design; the synchronization happens with the first gate; differences in timing are small due to the short length of the block

will be increased. But this does not corrupt the function. If no logic follows the last gate or another logic style is used a simple delay of the global clock through inverters can be used as a completion signal which clocks the last gate. It has to be assured that the outputs are processed correctly of the following logic. The main boundaries for an implementation are:

a) $t_{minRun} > t_{Clock} / 2$

b) $t_{maxRun} < t_{Clock}$

c) $t_{Cmin} < t_{Clock}$

where t_{Clock} is the cycle time, t_{minRun} and t_{maxRun} are the minimum and maximum propagation delay of the block, respectively (figure 4). The first condition holds because the first gate remains in the evaluation phase for half a clock cycle. If the outputs of the previous chain settle before this gate precharges the inputs are lost. The second condition means that calculation has to be finished in one clock cycle. At least the clock cycle has to be longer than every single minimum cycle time t_{Cmin} of the gates. The minimum number of logic gates in the chain depends on condition c and is about five. To achieve maximum performance it has to be verified that for maximum evaluation time t_{maxRun} of the gates the inputs at all gates arrive after the beginning of the evaluation phases to waste no time.

3.3 Skew Behavior and Inferences for Implementation

In our implementation of asynchronous logic inside a synchronous design the main problems of self-timing can be solved through calculation of the parameters of the logic cells at design time. The values to be considered are the shortest and the longest path that can exist. This includes connections between two or more parallel chains. Due to the very short chains no significant differences between the timing schemes can occur. The last gate of a sequence waits until the next first gate

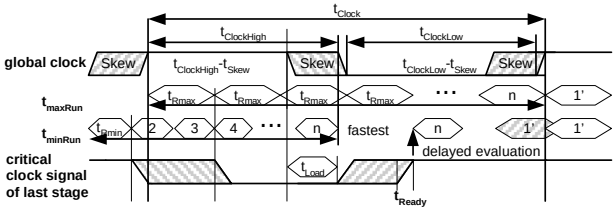

Figure 4. Important signals for the asynchronous chain in a timing scheme with clock skew; the evaluation times are usually directly connected; a delay occurs for the last stage for very short calculations but does not influence the function

generates the completion signal and the first gate waits until the inputs are settled, respectively. Therefore, clock skew is not harmful to some extent. A skew of the falling edge has to be taken into account in condition a), so we get:

a2) $t_{minRun} > t_{ClockHigh} + t_{Skew}$.

An earlier rising edge does not effect the runtime because the logic waits for the arrival of the inputs. Clock skew imposes the following boundaries:

d) $2 \cdot t_{Rmax} < t_{ClockHigh} - t_{Skew}$,

e) $t_{Load} < t_{ClockLow} - t_{Skew}$.

t_{Rmax} is the longest evaluation time for a single logic, $t_{ClockHigh}$ and $t_{ClockLow}$ denote the mean time of the high and low phase of the global clock, respectively. t_{Skew} represents the positive or negative skew of the phases. Because the skew lengthens the low phase and shortens the high phase and vice versa the same value is used to calculate the shortest low and high phases. Here, the skew value covers also the clock slopes. Figure 4 depicts this skew tolerance.

4. IMPLEMENTATION OF THE MULTIPLIER

An 8x8 bit multiplier was designed for verification. The simulations were made for an AMS 0.6µm double metal process with 3,3 V and minimum transistor sizes. The main focus has been put on the use of small and simple standard cells for single rail use. This decision was made as a TSPC design flow from simple standard cells exists and so a comparison between these two realizations can be done easily [10]. The design was made in a dual rail style, i.e. each dynamic cell has one output only but the inputs can be complementary. In step two the same logic functions were structured in the asynchronous style with DOMINO. Note that other logic styles could be used as well.

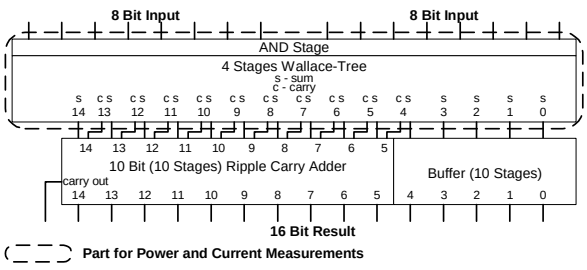

Figure 5. Structure of the implemented multiplier

5 Bit Ripple Carry Adder

A fully pipelined ripple carry adder structure was implemented. Therefore a TSPC realization needs five clock cycles for these adder stage. The asynchronous approach incorporates these five logic stages in one chain and needs only one clock cycle. This adder structure is used later to build the carry propagate adder (CPA) at the end of the multiplier.

8x8 Bit Wallace Tree Unsigned Integer Multiplier

A multiplier was implemented using a Wallace tree described in [1] and [2] (figure 5). The multiplier consists of four stages with carry-save adder (CSA) circuits and one preceding stage with the AND functions of the inputs. Therefore, the TSPC realization again has a latency of five clock cycles. Again, all five stages were merged to one chain in the asynchronous style. Two 5-bit ripple carry adders perform the final addition of the sum and carry bits of the Wallace tree. This realization results in a latency for TSPC of 15 clock cycles and for AC-TSPC of 3 clock cycles. In such pipelined designs output signals have to be stored in registers while others are processed. In the TSPC structure this means one buffer for each clock cycle and each signal. In the AC-TSPC structure the same TSPC buffers can also be used but the number of buffers is extremely reduced.

5. RESULTS

Table 1 gives the simulated results for the maximum frequency of the TSPC and AC-TSPC multipliers, the latency, the power consumptions and the maximum currents. The simulations for current and power consumption are given for the Wallace tree only due to a big difference in buffers for the ripple carry adder (TSPC: 180; AC-TSPC: 20). Therefore, results including these parts would hide the advantages of the asynchronous approach. On the other side, this effect demonstrates the advantage of AC-TSPC to reduce buffer efforts by mixed logic styles.

Table 1. Comparison of TSPC and AC-TSPC Multiplier Stage

	TSPC	***AC-TSPC***
min. cycle time (ns)	1.2	3.6
min. latency (for 3.6 cycle) (ns)	6 (18)	3.6 (3.6)
avg. power in VDD (3.6ns) (mW)	211	126
avg. power in clock (3.6ns) (mW)	183	20
max. current in VDD (3.6ns) (mA)	706	113
max. current in clock (3.6ns) (mA)	1080	123

The minimum clock cycle for TSPC-logic is much faster but the AC-TSPC logic calculates five logic gates in one cycle while TSPC can only calculate one logic gate in that time. Therefore, the overall latency of the AC-TSPC logic is shorter. It should be mentioned that the higher clock rate for the TSPC logic results in much higher power consumption. The simulations for the power consumption were made with a unified cycle time of 3.6 ns for both circuits. Therefore, the latency of AC-TSPC logic is then five times shorter. The reduction of the power consumption through dynamic switching is nearly 40%. The reduction of the clock load results in a 89% reduction of the power consumption. The reduction of the maximum current is also important for the design of the clock tree. The gates of the asynchronous logic switch sequentially. Therefore, the maximum current is extremely reduced (84%). A big advantage is the possibility to mix TSPC style with AC-TSPC chains. A circuit can use the benefits of both logic styles. This is also valid for implementing differential logic styles (e.g. DCVSL) in the asynchronous chains and joining these sequences with TSPC logic.

6. SUMMARY

This paper presents an implementation of asynchronous logic in a globally synchronous dynamic design which offers advantages in speed, power consumption, and in the reduction of the design effort for the clock tree. The simulated 8x8 bit multiplier with AC-TSPC logic exhibits a power reduction of 40% for the logic and of 89% for the clock tree. This is the result of the reduction of the clock tree capacitance due to the asynchronous logic chains. Furthermore, the latency of the circuit is reduced in comparison to a TSPC design because one clock cycle can be used more efficient. A realization of the AC-TSPC style is easy due to the possibility to connect simple cells in a dual rail style but also differential logic styles can be used. However, a combination with normal TSPC logic causes no problems. The advantages in skew tolerance for the clock tree and in reduction of the maximum currents on clock and VDD signals makes the design flow easier and more robust. Nevertheless, a careful design of the dynamic gates is required due to spikes and hazards. The application of this method is for fast design with short latencies and a single clock, e.g. algorithms for cryptology.

7. REFERENCES

[1] N. H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design*. Reading: Addison-Wesley Publishing Company, 1994.

[2] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs: Prentice Hall, 1992.

[3] P. Ng, P. T. Balsara and D. Steiss, *Performance of CMOS Differential Circuits*. IEEE Journal of Solid-State Circuits, Vol. 31, No. 6, June 1996.

[4] D. Harris and M. A. Horowitz, *Skew-Tolerant Domino Circuits*. IEEE Journal of Solid-State Circuits, Vol. 32, No. 11, November 1997.

[5] T. E. Williams and M. A. Horowitz, *A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider*. IEEE Journal of Solid-State Circuits, Vol. 26, No. 11, November 1991.

[6] J. Yuan, I. Karlsson and C. Svensson, *A True Single Phase Clock Dynamic CMOS Circuit Technique*. IEEE Journal of Solid-State Circuits, Vol. SC-22, 1987, pp. 899-901.

[7] L. G. Heller, W. R. Griffin, J. W. Davis and N. G. Thoma, *Cascode Voltage Switch Logic: A Differential CMOS Logic Family*. Proceedings IEEE International Solid-State Circuits Conference, 1984, pp. 16-17.

[8] D. Somasekhar and K. Roy, *LVDCSL: A High Fan-In, High-Performance, Low-Voltage Differential Current Switch Logic Family*. IEEE Transactions on VLSI Systems, Vol. 6, No. 4, December 1998.

[9] J. Park, J. Lee and W. Kim, *Current Sensing Differential Logic: A CMOS Logic for High Reliability and Flexibility*. IEEE Journal of Solid-State Circuits, Vol. 34, No. 6, June 1999.

[10] A. Wassatsch, *DYNAMIC – A Java Based Toolset For Integrating Dynamic Logic Circuits Into A Standard VLSI Design Flow*. International Cadence User Group Conference ICU, San Jose, September 2000, SIG IC - ic6.