Dynamic Single-rail Self-timed Logic Structures for Power Efficient Synchronous Pipelined Designs

Frank Grassert
Institute of Applied Microelectronics & Computer Science
University of Rostock
Richard-Wagner-Str. 31, D-18119 Rostock, Germany
Tel.: 0049 381 498 3534
frank.grassert@e-technik.uni-rostock.de

Dirk Timmermann
Institute of Applied Microelectronics & Computer Science
University of Rostock
Richard-Wagner-Str. 31, D-18119 Rostock, Germany
Tel.: 0049 381 498 3528
dirk.timmermann@e-technik.uni-rostock.de

ABSTRACT

The realization of fast datapaths in signal processing environments requires fastest, power efficient logic styles with synchronous behavior. This paper presents a method to combine improvements on algorithm and logic level. To reduce the power consumption of dynamic logic, a method for using single-rail structures is presented including a new scheme to realize inverting logic functions. It is shown that such structure is most efficient when redundant number systems are utilized. These self-timed logic is integrated in a global clock system using the Asynchronous Chain True Single Phase Clock (AC-TSPC) logic resulting in a latch-free structure. Comparisons with other logic styles show the achievement potential. First simulations for a horizontal redundant adder slice show area and power savings of 40% and 30% compared to complementary Domino logic.

Categories and Subject Descriptors

B.2.1 [**Arithmetic and Logic Structures**]: Design Styles – *Pipeline.*

General Terms

Design, Performance.

Keywords

Dynamic logic, low power, single-rail logic, self-timed logic, redundant numbers.

1. INTRODUCTION

The realization of data-paths with highest performance, i.e., high throughput or low latency, is a main issue in the area of signal processing. Applications, e.g. in wireless environments, emphasize power consumption as an important criterion for comparisons. On the way to more efficient algorithms, redundant number systems like Carry-Save reduce evaluation time by avoiding carry propagation as long as we stay in the redundant number system. Beneath the algorithmic level, there are several possibilities on the logic level to speedup these data-paths. The use of dynamic logic like Domino [1] ensures fastest evaluation because only N-transistors realize the logic function (explained in section 2). The question and the main scope of this paper is how to realize such fast algorithms in dynamic logic in a power efficient way.

As a realization for fastest evaluation with dynamic logic, Horowitz et al presented in [2] a self-timed scheme for a ring divider. In [3], the AC-TSPC logic is presented, an integration methodology for self-timed schemes in synchronous designs. Sechen discusses in [4] a scheme for single-rail dynamic logic. The advantages of redundant calculation are shown in papers like [5]. However, the efficient combination of algorithm and logic level remains an open question.

In this paper, a combination of methods from algorithmic and logic level is presented which results in a scheme to design power efficient pipelines. The advantages of redundant number systems are combined with new ideas on logic level to realize inverting functions in dynamic, single-rail logic styles and to allow single-rail, self-timed structures. This use of single-rail, self-timed logic reduces power consumption in comparison to dual-rail logic structures and does not require netlists with only non-inverting functions. Section 2 describes some basics for understanding the issues of dynamic logic. Section 3 deals with possibilities for the combination of single-rail logic with self-timed schemes when implementing redundant number systems. A completion detection scheme is shown where redundant numbers are used in an advantageous way. Section 4 explains methods for generating inverting outputs in dynamic single rail structures and shows a new robust realization with P-logic. Section 5 shows comparisons and simulation results of a redundant adder row.

2. BASICS

Comprehension of the pros and cons of dynamic logic, distinction between single- and dual-rail and the resulting inference for clocking schemes is fundamental and will be discussed briefly. Figure 1 shows Domino logic [1] and True Single Phase Clock (TSPC) logic [6]. The dynamic principle is based on two phases controlled by a clock signal and can be explained for Domino as follows: during precharge phase (clock is low), the dynamic node is precharged to high and the output is reset to low. During evaluation phase (clock is high), the output node can be discharged to ground, depending on the input values of the logic

GLSVLSI’03, April 28-29, 2003, Washington, DC, USA.

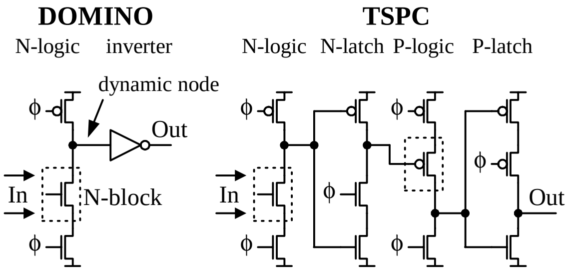

Figure 1. Domino and TSPC logic; Domino needs more than one clock phase, TSPC has alternating N-logic, N-latches and P-logic, P-latches.

tree. Because only N-transistors realize the logic function the evaluation is faster than in static CMOS. Again, because of merely N-transistors in the following logic, the output load is less than in static CMOS.

However, there are two main disadvantages of dynamic logic styles: realization of inverting logic functions is difficult and clock load is extreme as a clock signal is essential for the function of every single gate. In Domino logic, no inverting functions are possible at all (always inverting dynamic stage plus inverter) and, therefore, a netlist with only non-inverting functions is mandatory. Another way is to build up a dual-rail style which generates both the inverted and the non-inverted output independently. But, both inverted and non-inverted inputs are necessary. This results in two logic blocks thus doubling the area. In TSPC logic, an inverting function can be realized with a modified structure of the P-part, called N2-part. Furthermore, for an inversion inverted inputs are always necessary.

In pipelined designs, clocking structure severely affects clock load. TSPC logic results in extreme pipelining with highest clock load because every logic gate realizes also a register function. This ensures highest throughput and clock frequency. For low latency, self-timed structures are better. Here, the evaluation time of the structure can be reduced to only the sum of the single gate evaluation times without additional delays through latches. As a compromise, AC-TSPC logic [3] integrates short chains of self-timed logic into a synchronous design. This approach results in lower clock load and, consequently, lower power consumption. Furthermore, a latch-free structure is possible which reduces evaluation time and, again, power consumption.

3. USING REDUNDANT NUMBERS FOR SINGLE-RAIL SELF-TIMED SCHEMES

Because self-timed schemes are based on dual-rail structures to detect a complete evaluation, a completion detection in single-rail logic using redundant number representations is presented.

3.1 Basics of self-timed structures

The generation of a completion signal in dual-rail structures is simple because the outputs of all gates are on the same level during precharge phase. Only during evaluation phase, the outputs of the independent complementary logic blocks exhibit different values. The evaluation is complete when this change is detected. In [2], a self-timed structure is presented which results in minimum evaluation time. If the evaluated outputs of a gate have settled, a completion signal is generated and this sets the previous gate in the precharge phase (inputs are processed – start precharge). Precharged outputs set the previous gate in the evaluation phase (outputs are precharged – start next evaluation, inputs can be processed). Because the evaluation of the outputs starts only with valid inputs, each gate is waiting for valid inputs during the evaluation phase. Therefore, the evaluation time of the critical path is only the sum of the gate evaluation times without additional delays.

AC-TSPC [3] is a scheme to include such dual-rail self-timed structures in a global clock system. Hereby, it is possible to keep the critical path delay at minimum without additional delays through latches. Therefore, the evaluation time of the critical path is the minimum clock period. However, the structure must be carefully designed and a calculation of the timing behavior is advantageous. Furthermore, such structures reduce the sensibility against clock skew and, consequently, no additional delay for an ensured function is necessary.

3.2 Using redundant number representations

The completion detection differentiates the states of the output nodes after precharge and after evaluation. Using a representation of a redundant number, it is possible to ensure a change of one or more bits which permits completion detection. This assumes that all parallel gates evaluate in a small range of time Δt so that completion generation does not disturb slower or faster paths. This means, that the difference in evaluation time Δt is small enough so that delayed output signals are valid before the input signals are precharged. This assumption seems reasonable in most small blocks with limited logic depth, e.g. in datapaths. As an example, table 1 shows a representation of a radix 2 signed-digit number, but similar coding is possible for other redundant number representations also with higher radix. The redundant digits –1, 0, 1 are represented with two signals, i.e. two bit. The bit representation ensures that at least one bit changes its value during evaluation. Therefore, a completion can be detected if we take both bits into account. Figure 2 shows such a structure. Another approach for a generalization of such completion detection was presented in [7].

Table 1. SD-digit representation.

Digit	Self-timed representation
-1	10
0	01
1	11
free	00 – precharge

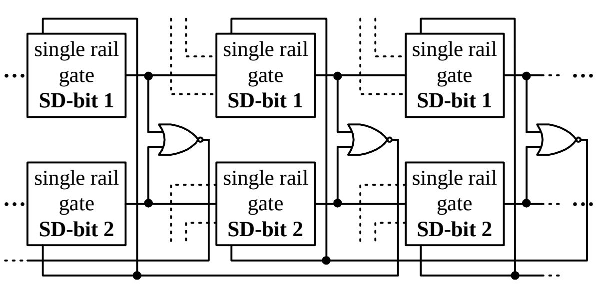

Figure 2. Self-timed structure with dynamic, single-rail logic implementing a redundant number representation.

4. INVERTING FUNCTIONS IN DYNAMIC LOGIC STYLES

To enable the use of simple single-rail logic equivalent to static CMOS, an inversion is necessary. However, this is still one of the main problems as explained above. In the following section, we discuss existing methods and introduce a new solution.

4.1 Existing methods

There are several ideas of dealing with inverting functions. First, a netlist which contains no inverting functions can be used. The generation of such netlists often results in area enlargement and doubled structures. From the power consumption point of view, this is not desirable, too. Second, as described in section 2, the use of differential or dual-rail logic styles solves the inversion problem. It should be noticed that there is no easy way to switch between single- and dual-rail. Consequently, any stage must have two logic blocks to generate both the inverted and the non-inverted output. This nearly doubles area and the clock load is raised. Third, TSPC logic styles like All-N-logic solve the problem but incorporate a big overhead on latches and result in extreme pipelining. These three solutions limit logic synthesis, raise power consumption or fix the logic style. Further, there are solutions to take the dynamic node as an inverted input. In [7] such a scheme is explained, but these method results in additional constraints for the timing between clock and input signals.

4.2 New P-logic scheme

Our new scheme to solve the inversion problem is shown in figure 3. This scheme bases on a completion signal which must be generated for a group of dynamic gates explained in section 3.2. Then, an inverted output signal can be generated with a following P-logic. The completion signal changes to low if one or more gates have already evaluated. This sets the P-logic to evaluation and it inverts the output of the gate. It is essential that the input of the P-logic does not change during evaluation because a high to low transition at the output of the P-logic is not allowed due to the consecutive dynamic gate. During precharge of the gates, the P-logic output can have a glitch because the clock signal of the P-logic is still low while the P-logic input changes to low. To prevent this behavior, an AndNor gate for generation of the completion signal can be used (figure 3). Then, the clock signal of the gates switches the P-logic off during precharge of its input. This inverting scheme is robust even if the inverted outputs are not always driven but a following buffer is possible. However, the evaluation time of the whole structure is affected. All inverted outputs are generated after all other outputs stay stable and the completion signal has settled.

5. COMPARISONS AND EXAMPLE

Because a synchronous design of pipeline structures is essential, the self-timed scheme presented in [3] was adapted to integrate the single-rail self-timed parts into a global synchronous block. For a detailed explanation, we must refer to [3].

Comparisons between different logic styles are very difficult because of the different strategies for clocking. However, we try a comparison between Domino, Clock-Delayed (CD) Domino, AC-TSPC (dual-rail) [3] and the presented self-timed single-rail structure using the scheme of [3]. It is essential to understand the structural differences between these logic styles. Figure 4 depicts the rough sketches of a pipeline comprising six gates. First, Domino logic requires for each gate a complementary gate to realize inverting logic functions. If the netlist consists of only non-inverting functions the dual-rail structure is not necessary. This requirement is hard to fulfill and, therefore, Domino logic belongs for larger netlists to dual-rail logic styles. Second, in figure 4 the pipeline of CD-Domino consists of simple single-rail gates. No complementary structure is necessary because inverting functions can be realized as described in [4]. Third, the AC-TSPC logic consists of only dual-rail gates, a complementary structure is necessary for the generation of self-timed signals [3]. The self-timed signal generation requires extra logic. Fourth, the structure presented in this paper realizes self-timing with single-rail gates, no complementary structure is necessary. With this knowledge, an area comparison is given in table 2.

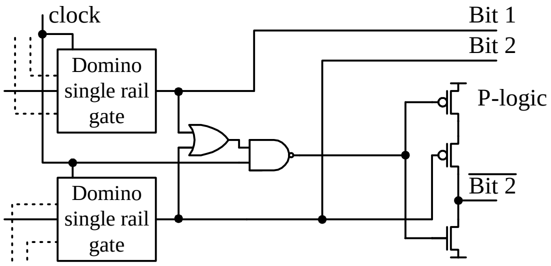

Figure 3. P-logic structure for inverting functions in a dynamic, single-rail logic.

Table 2. Area comparison in transistors per gate depending on gate type and the number N of inputs.

	Domino	CD-Domino	AC-TSPC	Single-rail self-timed
Non-inv. gate	N+4	N+4	dual-rail necessary	N+4+2
Inverting gate	dual-rail necessary	N+7	dual-rail necessary	N+4+11+2
Design with 10% inv. gates	2(N+4)	~(N+4,3)	2(N+4)+4	~(N+7)

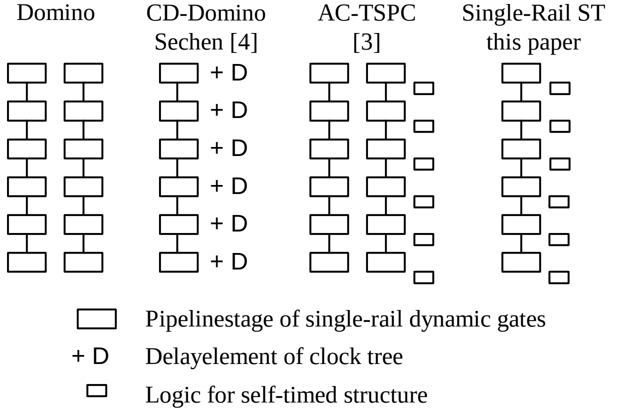

Figure 4. Rough sketch of the pipeline structures to illustrate the differences and the area cost.

For a simple Domino gate the number of transistors is one per input signal (for simple logic) plus 2 clock transistors and 2 transistors for the following inverter. An inverting gate does not exist. CD-Domino uses the same gates for non-inverting functions like Domino but additionally an inverting gate exists, which needs additional logic. AC-TSPC is a dual-rail structure. Therefore, no simple gates exist. Dual-rail gates always generate both output signals. Single-rail self-timed gates base also on simple Domino gates but at least two gate outputs drive a completion logic. Therefore, we need an additional 'half' completion gate with 2 transistors. The inverting gate is realized as described above with additional 11 transistors. For comparison, we generated values for a fictive design with 10% inverting gates after logic optimization, which seems reasonable. Domino and AC-TSPC logic must be build up in dual-rail. The values for CD-Domino and single-rail self-timed logic were approximated. Here, CD-Domino has the lowest area consumption. Note, that, as depicted in figure 4, the additional area for the delay components in the clock tree has not been taken into account.

As a wrap-up, Table 3 shows values for comparison of different logic styles, taken from [3,4] and [7]. Note, the relative values for static CMOS, Domino and CD-Domino were taken from [4]. The values for AC-TSPC are results from the design described in [3]. In contrast to the other designs, a larger pipeline design with register stages has been used in [3]. The static CMOS realization has a high fraction of registers to reach a comparable structure. Therefore, the values for AC-TSPC are not very expressive. The values for the presented single-rail selftimed structure are calculated from the relative results in [7]. A comparison with dual-rail Domino was made and, therefore, these data are comparable to the results from [4].

Table 3 shows that CD-Domino and Single-rail self-timed logic have about the same power-delay-product. CD-Domino has lower delay but needs more area and has a higher power consumption. Both consist of Domino gates but realize different clocking topologies. CD-Domino has a fixed delay clocking which is critical to adjust. Our single-rail self-timed clocking is data-dependent and needs no trimming. Integration of the self-timed structure into a global clock scheme is much easier. However, the design of both CD-Domino and Single-rail self-timed logic needs excessive calculations with the respect to the timing behavior of the design.

Table 3. Comparison of rough relative values from [3,4] and [7] for different logic styles. Note, that the results marked with * are made for a pipeline design with registers.

	Stat. CMOS	Domino	CD-Domino	AC-TSPC	Single-rail self-timed
Delay (D)	1	0,6	0,4	0,3*	0,5
Area	1	1,6	1,5	n.a.	1
Power (P)	1	2,6	2,3	1,5*	1,8
P*D	1	1,6	0,9	0,5*	0,9

To assess our results, we chose an horizontal adder slice for a redundant multiplier using a signed-digit representation and the self-timed scheme presented in [3] to integrate the self-timed parts into a global clock. For testing purposes, 2-bit adder structures with 6 successive gates in a synchronous block were simulated with a typical 0.6 µm CMOS process with 3.3V. Results are shown in figure 5, which have been obtained from a hand-optimized structure.

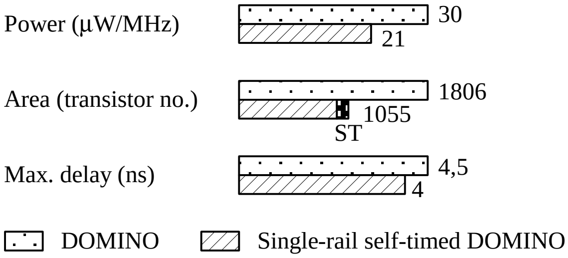

Figure 5. Comparison of the SD-adder structure in single-rail, self-timed Domino logic and complementary Domino; ST marks the area for self-timed logic.

6. SUMMARY

This paper presents a comprehensive approach for combining methods from algorithmic and logic levels of circuit design: with redundant numbers, dynamic single-rail self-timed logic providing inverting functions becomes feasible. Therefore, the approach of redundant number systems to speedup the evaluation can be enhanced with fast and power efficient logic styles because single-rail logic saves area and power. An integration of self-timed structures in synchronous designs lowers clock load and, therefore, power consumption while the speed is maintained through the latch-free scheme. The presented ideas show a way to design fast and power efficient pipeline circuits. As a first example, a redundant number adder was simulated. In comparison to a dual-rail Domino realization, the power and area consumption was reduced to about 70% and 60%, respectively.

7. REFERENCES

[1] R. H. Krambeck, C. M. Lee and H.-F. S. Law, "High-Speed Compact Circuits with CMOS", Journal of Solid-State Circuits, IEEE, Vol. SC-17, No. 3, June 1982.

[2] 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.

[3] F. Grassert, D. Timmermann, "Dynamic Single Phase Logic with Self-timed Stages for Power Reduction in Pipeline Circuit Designs", IEEE International Symposium on Circuits and Systems (ISCAS), May 2001, pp. IV 144-147.

[4] G. Yee, C. Sechen, "Clock-Delayed Domino for Dynamic Circuit Design", IEEE Transactions on VLSI Systems, Vol. 8, No. 4 , August 2000.

[5] S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi, N. Takagi, "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation", Proc. 8th. Symposium on Computer Arithmetic, New York, 1987, pp. 80-86.

[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] F. Grassert, D. Timmermann, "Single-Rail Self-timed Logic Circuits in Synchronous Designs", IEEE MWSCAS Conference, Tulsa, August 2002.