Performance Analysis and Optimization of NCL Self-Timed Rings

W. Kuang, J.S. Yuan*, R. DeMara, [1]M. Hagedorn, and [2]K. Fant
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida 32816
[1]Theseus Logic, Inc., Maitland, Florida 32751
[2]Theseus Research, San Jose, California, 95054

Abstract: A self-timed ring using NULL Convention Logic (NCL) is presented. Analytical method to evaluate the speed of NCL rings has been developed. The analytical predictions are verified by Synopsys simulation. Excellent agreement between the theoretical predictions and simulation results is obtained. The analysis leads to speed optimization of a 24-bit NCL divider to achieve a throughput of 4.21 ns.

Index terms — self-timed ring, division, SRT algorithm, Null Convention Logic

1 Introduction

In recent years, the implementation of floating-point division has received increasing attention. The computation time of a division operation is fairly long compared with other floating-point operations such as multiplication and addition. Moreover, applications such as three-dimensional graphics use division frequently. It has been shown that ignoring its implementation can result in significant system performance degradation for many applications [1].

A self-timed ring is an efficient approach to the implementation of division based on the SRT algorithm [2]-[4], which is the most commonly used algorithm in modern processors [1]. In this ring structure, the computation can progress as fast as the circuit configuration and fabrication technology allow, rather than being slowed down by the worst-case delay or clock skew margins in synchronous counterparts. Furthermore, due to the identity of the iterative operations in division, a self-timed ring with a few stages can be used to perform a floating-point mantissa division operation without speed loss compared with the fully expended pipeline version. One of the early designs was proposed by Williams [2]. Since then several modifications have been suggested in order to improve performance [3],[4]. However, all of these methods employ differential cascode voltage switch logic (DCVSL) [5]. Some conventional CMOS design principles are not applicable to DCVSL. For example, the rise and fall times of output nodes in DCVSL circuits are generally different due to an inherent asymmetry in NMOS tree and PMOS load. Careful sizing of transistors is required for DCVSL to avoid possible problems associated with races [6]. For DCVSL, a timing assumption appears during the pre-charge phase. It is assumed that when request signal goes low, the input data are all reset. This is not always checked [7].

The method for self-timed ring design presented in this paper is based on NULL Convention Logic (NCL). NCL is a new clockless methodology for digital system design [8], [9]. It offers a quasi delay-insensitive logic paradigm where control is inherent with each datum, and follows the so-called "weak conditions" of Seitz's delay-insensitive signaling scheme [10]. The NCL paradigm assumes that forks in wires are isochronic [11]. Besides the common advantages of general asynchronous circuits, such as speed independency, average-case delay, low power, low noise and electromagnetic interference (EMI), a formal and systematic method of design and optimization has been developed to make the design of NCL circuits easy [12] [13]. A complete register-transfer level (RTL) design flow for NCL can be performed using commercial CAD tools [14], such as Synopsys and Modelsim VHDL simulator.

However, the speed of an NCL ring circuit depends not only on the latencies of individual stages in data path, but also on the speeds of handshake circuits and the number of stages. To investigate how these factors affect the ring speed, an analytical method is developed using dependency graphs. Performance analysis of self-timed rings yields important implications for designing NCL rings with optimal speed performance or with an appropriate speed-area tradeoff. These implications are applied to the design of a 24-bit floating-point mantissa division.

The remainder of this paper has four sections. Section 2 briefly presents NCL concepts, using modules known as M-of-N threshold gates and a simple ring structure. Section 3 describes how to determine the overall speed performance of an NCL ring in terms of component delays and ring configurations. In Section 4, rings with different parameters, where each stage is realized as a half adder, are simulated in Synopsys. The analytical results are compared with Synopsys simulation. The design principles are applied to optimization of a 24-bit divider. The conclusion is given in Section 5.

2 Overview of null convention logic

NCL uses symbolic completeness of expression to achieve self-timed behavior [8]. A symbolically complete expression is defined as an expression that only depends on the relationships of the symbols presented in the expression without a reference to the time of evaluation. In general, a multi-rail signal can be used to incorporate data and control information into one mixed signal path to eliminate the time reference, and therefore to form a symbolically complete expression. Typically, a dual-rail signal, D, consists of two wires, D_0 and D_1, which represent a value from the set {Data0, Data1, Null}, shown in Table I. The Data 0 state (D_0 = 1, D_1 = 0) corresponds to a Boolean logic 0, the Data 1 state (D_0 = 0, D_1 = 1) corresponds to a Boolean logic 1, and Null state (D_0 = 0, D_1 = 0) corresponds to the empty set meaning that the value of D is not yet available. The state (D_0 = 1, D_1 = 1) is forbidden.

The primitive element for NCL is an M-of-N threshold gate with hysteresis. This gate has N input signals and one output, shown in Fig. 1 a). The signal values are referred to as DATA and NULL for a high voltage level and a low voltage level respectively. This gate has two important properties, threshold behavior and hysteresis behavior. The threshold behavior requires that the output becomes DATA if at least M of the N inputs has become DATA. The hysteresis behavior requires that the output only changes after a sufficiently complete set of input values have been established. In the case of a transition to DATA, the output remains at NULL until at least M of the N inputs become DATA. In the case of a transition to NULL, the output remains at DATA until all N inputs become NULL. As special examples, an N-of-N gate is an N-input Muller C-element while a 1-of-N gate corresponds to an N-input OR gate. Transistor-level design for M-of-N threshold gates is described in [15]. As an example, a 2-of-3 gate having inputs A, B and C is shown in Fig. 1 b). It includes four blocks (Go to NULL, Go to DATA, Hold NULL, Hold DATA), one inverter and two feedback transistors (pMOS for Hold NULL, nMOS for Hold DATA).

Threshold gates can be used to build combinational circuits. An NCL half adder is shown in Fig. 2, where (*x*_0, *x*_1) and (*y*_0, *y*_1) denote the dual-rail encoded input addends. (*c*_0, *c*_1) and (*s*_0, *s*_1) denote the dual-rail encoded carry and sum outputs respectively.

A typical NCL circuit structure forms a pipeline, in which each stage consists of a register, completion detection circuit and a combinational circuit [8]. An NCL ring is formed when the output of a pipeline is connected to its input as shown in Fig. 3. For sake of simplicity, a half adder is used as the combinational circuit for each stage in the ring. Note that, although one of the three bits in data bus is not involved in the half adder computation, it puts a load on the completion detection block. All of the registers are initialized to Null outputs except that the outputs of register R_N are initialized to Data 0 or 1. Figure 4 illustrates the gate-level structures of register and completion detection circuit used in the ring. Th22nx0 (Th22dx0) is a 2-of-2 threshold gate that is initialized to NULL (DATA). Th12bx0 is a 1-of-2 threshold gate with an inverter following. Th22x0 is a 2-of-2 threshold gate without initialization control.

3 Analysis of rings Using dependency graphs

An efficient analysis method for ring speed performance is based on use of a dependency graph, which was proposed by Williams [16] [17]. Williams' theory on rings shows how to analyze a ring performance in terms of speed and area. This theory will be used below to develop an analytical model for evaluating self-timed rings implemented in NCL.

First, several parameters need to be defined. Local parameters include the forward latency (L_f), the reverse latency (L_r), and the local cycle time (P). The forward latency specifies the delay from valid data outputs at one stage to valid data outputs at the following stage without waiting for request signal. The reverse latency specifies the delay from the request of a stage output to the request of its predecessor's output without waiting for Data or Null in data path. The local cycle time specifies the delays of all the transitions necessary for a stage to pass a token, and become enabled again for the next token. This limits the maximum throughput of a ring. A key global parameter is the total latency (λ), that is the delay between the introduction of a new data token into the ring and the removal of the corresponding processed token after the token has passed through stages necessary for solution of an iterative problem. Since the number of iterations for a given problem is constant and each iteration corresponds to one stage, the total latency is proportional to the average time for a token to pass through one stage. The average propagation delay of a single stage will be used as a metric of ring speed.

3.1 Folded dependency graph

A dependency graph of a pipeline can be constructed from its structure. Figure 5 a) shows the dependency graph of a pipeline corresponding to Fig. 3. The nodes of the graph correspond to specific rising (up arrow) or falling (down arrow) transitions of circuit components (R refers to register, F refers to combinational function block for one stage, D refers to completion detection), and the edges represent the dependencies of each transition on the outputs of other components. The delay of each transition is represented by a value attached to the node in the graph.

Dependency graphs can be used to determine both the stage latencies (forward and reverse) and the local cycle time [17]. When the stages are identical, the folded dependency graph can be used to calculate the local parameters more conveniently as shown in Fig. 5 b). In the folded dependency graph, it is not necessary to subscript the nodes with a particular stage index since the node represents that same transition in all stages. For each edge, there is a stage index offset attached giving the offset in stage index to which the dependency refers.

The local parameters can be calculated by the following Williams' equations [16]:

$$L_f = \max\left(\frac{\sum_{i\in path} t_i}{\sum_{i\in path} w_i}\right) \text{ over all non-repeating cyclic paths with } 0 < \sum_{i\in path} w_i \text{ , } w_i \geq 0$$

$$L_r = \max\left(\frac{\sum_{i\in path} t_i}{-\sum_{i\in path} w_i}\right) \text{ over all non-repeating cyclic paths with } 0 > \sum_{i\in path} w_i \text{ , } w_i \leq 0$$

$$P = \max\left(\sum_{i\in path} t_i\right) \text{ over all non-repeating cyclic paths with } 0 = \sum_{i\in path} w_i$$

where t_i is the ith node transition delay in the cyclic path chosen and w_i is the corresponding stage index offset.

3.2 Local analysis

As a particular example, the latencies and the local cycle time of NCL pipeline can be analyzed based on folded dependency graph. The forward latency is given by

$$L_f = \max(F\uparrow + R\uparrow, F\downarrow + R\downarrow) \tag{1}$$

The reverse latency is given by

$$L_r = \tfrac{1}{2}\left(R\uparrow + D\downarrow + R\downarrow + D\uparrow\right) \tag{2}$$

To determine the local cycle time, the longest cycle with zero offset needs to be found. There are three possibilities for the longest zero offset cycle in Fig. 5 b). as follows

$F\uparrow,R\uparrow,D\downarrow,R\downarrow,D\uparrow,R\uparrow,F\uparrow,R\uparrow;$ $F\uparrow,R\uparrow,D\downarrow,R\downarrow,F\downarrow,R\downarrow,D\uparrow,R\uparrow;$ $F\downarrow,R\downarrow,D\uparrow,R\uparrow,D\downarrow,R\downarrow,F\downarrow,R\downarrow.$

The lengths of the three paths are

$$P1 = 2F\uparrow+2R\uparrow \quad +R\uparrow+R\downarrow+D\uparrow+D\downarrow \tag{3a}$$

$$P2 = F\uparrow+F\downarrow+R\uparrow+R\downarrow +R\uparrow+R\downarrow+D\uparrow+D\downarrow \tag{3b}$$

$$P3 = 2F\downarrow+2R\downarrow \quad +R\uparrow+R\downarrow+D\uparrow+D\downarrow \tag{3c}$$

Since the sum of the first four terms of *P2* is the middle point of the sum of the first two terms of *P1* and the sum of the first two terms of *P3*, the local cycle time is

$$P = \max(P1, P2, P3) = \max(P1, P3) \geq P2. \tag{4}$$

From (1), (2), and (4), the local cycle time is obtained in terms of the forward and reverse latencies:

$$P = 2(L_f+L_r) \tag{5}$$

3.3 Ring performance analysis

The local cycle time of an NCL ring satisfies $P = 2(L_f+L_r)$, which means the token flow rate is limited either by the forward latency (data-limited) or by the reverse latency (bubble-limited) [16].

We consider the case that there is only one token in an *N*-stage ring. There are (*N*-2) bubbles in the *N*-stage ring. When *N* is larger, there are so many bubbles in the ring that the token circulation period will fall in the data-limited region, and

$$P_{ring} = N \cdot L_f \tag{6}$$

When *N* is smaller, there are so few bubbles that the token circulation period will fall in the bubble-limited region [16], and

$$P_{ring} = \frac{2N \cdot L_r}{N - 2} \tag{7}$$

In the data-limited region, the delay of the completion detection is removed from the critical path. If two local parameters are given, the optimal number of stages is defined as the minimum *N* in the data-limited region. It is achieved when (6) and (7) are equal to

$$N_{optimal} = 2(1 + \frac{L_r}{L_f}) \tag{8}$$

The final expression of the token circulation period is given by

$$P_{ring} = \begin{cases} \dfrac{2N \cdot L_r}{N - 2} & , N \le 2(1 + \dfrac{L_r}{L_f}) \\ N \cdot L_f & , N > 2(1 + \dfrac{L_r}{L_f}) \end{cases} \tag{9}$$

Therefore, the average time for a token to pass through one stage is

$$t_{stage} = \begin{cases} \dfrac{2 \cdot L_r}{N - 2} & , N \le 2(1 + \dfrac{L_r}{L_f}) \\ L_f & , N > 2(1 + \dfrac{L_r}{L_f}) \end{cases} \tag{10}$$

We see from (8) that when L_r is decreased by speeding up the register and/or completion detection circuit, the number of stages required for optimal speed will be reduced. On the other hand, speeding up the forward data path will lead to an increase of the number of stages required for a higher optimal speed. It is also implied from (10) that a ring with too few stages will fall in the bubble-limited region, where the ring speed can be improved by speeding up the reverse latency and/or increasing the number of states until the optimal speed is achieved. A ring with more stages will fall in the data-limited region, where the ring speed is the optimal speed that depends only on the forward latency L_f.

3.4 Ring initialization

A token introduction, as well as the characteristics of the threshold gates, requires a correct initialization for NCL rings. Since the threshold gates used in combinational circuits have no reset input for initialization, the validity of the initial output for each gate is guaranteed only by the completeness of input signals. However, dual-rail code (1,1) is forbidden in NCL. Therefore, it is convenient that all inputs of a combinational circuit are initialized as Null so that the circuit initial outputs are validly all Null. In order to realize this idea for the NCL ring initialization, an additional register with initialized Null output is inserted between the register with initialized Data output and the following combinational circuit, shown in Fig. 6.

For a ring initialized in this way, it is not true that each stage is identical. Compared with the analysis in subsection 3.3, the additional register contributes a delay to the token circulation period in data-limited region while an additional bubble will modify the token circulation period in bubble-limited region. The token circulation period for a ring with an initializing stage is given by

$$P'_{ring} = \begin{cases} \dfrac{2(N+1)\cdot L_r}{N-1} & , N \le N'_{optimal} \\ N \cdot L_f + R & , N > N'_{optimal} \end{cases} \qquad (11)$$

where R is the maximum of $R\uparrow$ and $R\downarrow$. The optimal number of stages for a ring with an initializing stage is similarly defined as the value corresponding to the intersection of the bubble-limited region and data-limited region, which is given by

$$N'_{optimal} = \frac{L_f + 2L_r - R + \sqrt{\left(L_f + 2L_r - R\right)^2 + 4L_f\left(R + 2L_r\right)}}{2L_f} \tag{12}$$

Note that $N'_{optimal}$ is slightly smaller than $N_{optimal}$ in (8). The corresponding average time for a token to pass through one stage is given by

$$t'_{stage} = \begin{cases} \dfrac{2 \cdot (1 + \frac{1}{N})L_r}{N - 1} & , N \le N'_{optimal} \\ L_f + \dfrac{R}{N} & , N > N'_{optimal} \end{cases} \tag{13}$$

Noting the fact that R is usually small, Equation (13) shows that the introduction of the initializing register improves the ring speed in bubble-limited region while leading to a negligible degradation of the ring speed in data-limited region.

4 Simulation results and discussion

To verify the effectiveness of the above analysis, the ring circuits are simulated by VHDL code in Synopsys. The delays of threshold gates specified in the VHDL library are listed in Table II. The three local parameters L_f, L_r, P based on the gate delays are calculated in Table II also.

First, the ring performance can be evaluated by Equations in Section 3 without running the VHDL simulation. The token circulation periods as a function of the number of stages are plotted by in Fig. 7. The average times of one stage as a function of the number of stages are plotted by in Fig. 8. In these figures the dashed lines correspond to equation (9) and equation (10), while solid lines correspond to equation (11) and equation (13).

Second, by writing VHDL codes for ring circuits and running simulation, the token circulation period can be measured based on the output waveform and thus the ring performance can be obtained. The corresponding results are plotted by star marks (*) and circle marks (o) in Fig. 7 and Fig. 8.

The analytical and simulation curves agree with each other very well. Although a discrepancy happens due to data-dependent delay and the assumption that the half adder has the same delay for rising and falling transition, it is negligible.

The results in the previous section can be used as a guideline to optimization for SRT division circuit. A Radix-2 SRT division algorithm can be implemented by designing the combinational circuits in the ring structure (Fig. 6). Figure 9 shows the combinational circuit for one stage of a 24-bit division. When the data wavefront arrives, the combinational circuit produces one bit of the quotient and the partial remainder that will be used in the next stage. In order to speed up the calculation of partial remainder, the partial remainder is usually represented by a carry-save redundant binary [18]. The forward latency of the division ring is approximately 4.19 ns calculated from the delays of components. Each register in the ring passes 75-bit data, including 24-bit divisor, 25-bit carry and 24-bit sum for partial remainder, 1-bit for logic “1”, 1-bit for logic “0”. If the completion detection circuit is implemented in serial as shown in Fig. 10 a), the reverse latency of 22.5 ns can be determined by the method in Table II. From Eqs. (12) and (13), a 13-stage ring will achieve the optimal speed 4.21 ns. A fast completion detection circuit can significantly reduce the number of stages for the optimal speed. A parallel implementation of the completion detection circuit, shown in Fig. 10 b), can reduce the reverse latency to 1.865 ns, and thus result in a 3-stage ring for the optimal speed. VHDL simulation results are plotted in Fig. 11.

5 Conclusion

From the analysis, as well as simulation, some important principles for optimal ring design are suggested. When the information of R, F, and D delays is known, the optimal number of stages can be determined by Equation (4), which means that further increase of the number of stages doesn't improve speed, and that a ring with fewer stages will become slower. The optimized speed only depends on the forward latency L_f, independent of the delay of completion detection block.

As for an optimized NCL ring, further speed improvements of components R, F, D provide different opportunities to further improve the ring overall performance. Improving register speed directly leads to increasing the overall ring speed, while the number of stages almost doesn't need to change. Speeding up the combinational circuit doesn't improve the ring overall speed unless the number of stages is increased. Speeding up the completion detector doesn't improve the ring speed, but it allows the number of stages to be reduced without speed loss.

References

1 Oberman,S.F. and Flynn,M.J.: 'Division algorithms and implementations,' *IEEE Trans. Computers*, 1997, **46**, (8), pp. 833-854

2 Williams,T.E. and Horowits,M.A.: 'A zero overhead self-timed 160-ns 54-b CMOS divider,' *IEEE J. Solid-State Circuits*, 1991, **26**, (11), pp. 1651-1661

3 Matsubara,G., Ide,N., Tago,H., Suzuki,S. and Goto,N.: '30-ns 55-b shared radix 2 division and square root using a self-timed circuit,' *Proceedings of the 12th Symposium Computer Arithmetic*, 1995 , pp. 98-105.

4 Matsubara,G. and Ide,N.: 'A low power zero-overhead self-timed division and square root unit combining a single-rail static circuit with a dual-rail dynamic circuit,' *Proceedings Third International Symposium on Advanced Research in Asynchronous Circuits and Systems,* 1997, pp.198-209.

5 Heller,L.G., Griffin,W.R., Davis,J.W., and Thoma,N.G.: 'Cascade voltage switch logic: a differential CMOS logic family,' in *Proc. IEEE International Solid State Circuits Conf.*, Feb. 1984, San Francisco, CA, pp. 16-17.

6 Chu,K.M. and Pulfrey,D.L.: 'A comparison of CMOS circuit techniques: differential cascode voltage switch logic versus conventional logic,' *IEEE J. Solid-State Circuits*, 1987, **22**, (4), pp.528-532

7 Renaudin,M., Hassan,B.E., and Guyot,A.: 'A new asynchronous pipeline scheme application to the design of a self-timed ring divider,' *IEEE J. Solid-State Circuits*, 1996, **31**, (7), pp.1001-1012

8 Fant,K.M. and Brandt,S.A.: 'NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis,' *Proceedings, International Conference on Application-Specific Systems, Architectures and Processors*, 1996, pp. 261-273

9 Fant,K.M. and Brandt,S.A.: 'Null Convention Logic System,' US Patent 5,305,463, April 19, 1994.

10 Seitz,C.L.: 'System timing,' in Introduction to VLSI Systems, Addison-Wesley, 1980 pp.218-262

11 Berkel,K.V.: 'Beware the isochronic fork,' *Integration, the VLSI Journal*, 1992, **13**, (2), pp.103-128

12 Smith,S.C.: 'Gate and throughput optimizations for Null Convention self-timed digital circuits,' PhD Dissertation, Department of electrical and computer engineering, University of Central Florida, 2001

13 Smith,S.C., DeMara,R.F, Yuan,J.S, Hagedorn,M., and Ferguson,D.: 'Delay-insensitive gate-level pipelining,' *Integration, the VLSI Journal,* 2001, **30**, (2), pp.103-131

14 Ligthart,M., Fant,K., Smith,R., Taubin,A., Kondratyev,A.: 'Asynchronous design using commercial HDL synthesis tools,' Proceedings sixth international symposium on advanced research in asynchronous circuits and systems, 2000, pp.114-125

15 Sobelman, G.E. and Fant, K.: ' CMOS circuit design of threshold gates with hysteresis,' in *Proc. 1998 IEEE Int. Symp. Circuits and Systems*, vol. 2, 1998, pp. 61-64

16 Williams,T.E.: 'Self-timed rings and their application to division,' Ph.D. dissertation, Stanford University, 1991

17 Williams,T.E.: 'Performance of iterative computation in self-timed rings,' *Journal of VLSI signal processing*, 1994, **7,** (1-2), pp. 17-31

18 Parhami,B.: 'Computer Arithmetic: Algorithms and Hardware Designs,' Oxford University Press: New York, 2000.

Table I Dual-rail encoding

Logic value	**Encoding**	
	*D*_0	*D*_1
Data 1	0	1
Data 0	1	0
Null	0	0
Invalid	1	1

Table II Gate delays and local parameters

Gate delays / local parameters	**Value (ns)**
Th22nx0↑ (R↑)	0.29
Th22nx0↓ (R↓)	0.16
Th12bx0↑ ($D1$↑)	0.45
Th12bx0↓ ($D1$↓)	0.22
Th22x0↑ ($D2$↑)	0.40
Th22x0↓ ($D2$↓)	0.20
Half adder	0.47
L_f= (Half adder) +R↑	0.76
L_r=0.5(R↑+R↓+$D1$↑+$D1$↓+2$D2$↑+2$D2$↓)	1.16
$P = 2(L_f+L_r)$	3.84

Captions and subcaptions to illustrations

Fig. 1. Threshold gate with hysteresis

a) Symbol of an M-of-N gate

b) Schematic of 2-of-3 gate

Fig. 2. An NCL half adder

Fig. 3. An NCL ring whose each stage is a half adder

Fig. 4. 3-bit register and completion detection

Fig. 5. Dependency graph for NCL pipeline

a) Extended dependency graph

b) Folded dependency graph

Fig. 6. A ring with an initializing stage (R_i, D_i)

Fig. 7. Circulation period versus the number of stages

Fig. 8. Per-stage time versus the number of stages

Fig. 9. Combinational logic for one stage of a Radix-2 SRT division

Fig. 10. Completion detection circuits

a) A serial architecture

b) A parallel architecture

Fig. 11. Simulation results for the 24-bit divisions with different completion detection circuits

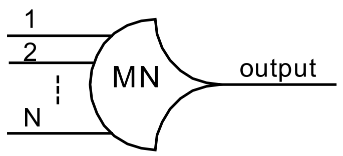

Fig. 1 a)

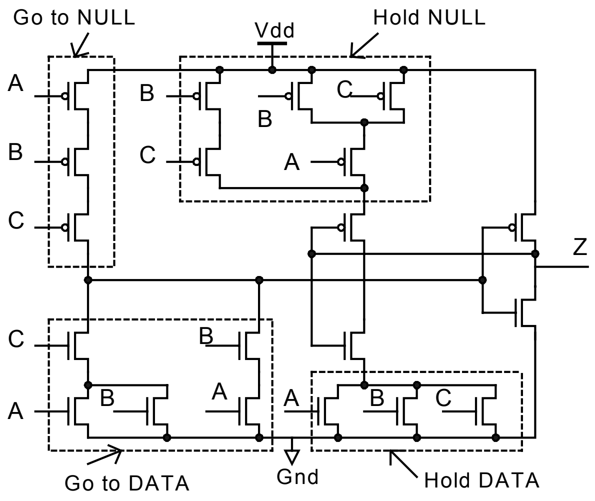

Fig. 1 b)

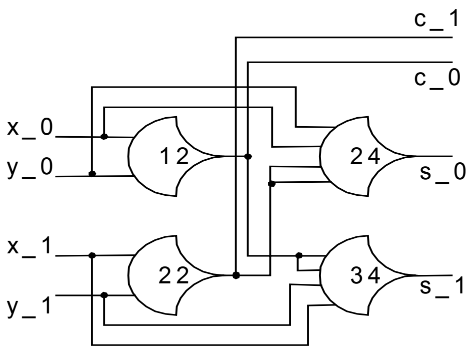

Fig. 2

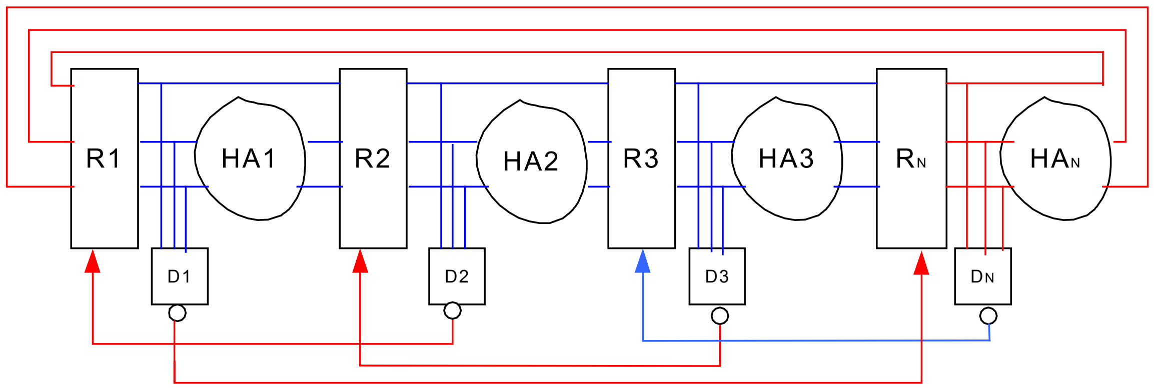
R1
HA1
R2
HA2
R3
HA3
RN
HAN
D1
D2
D3
DN

Fig. 3

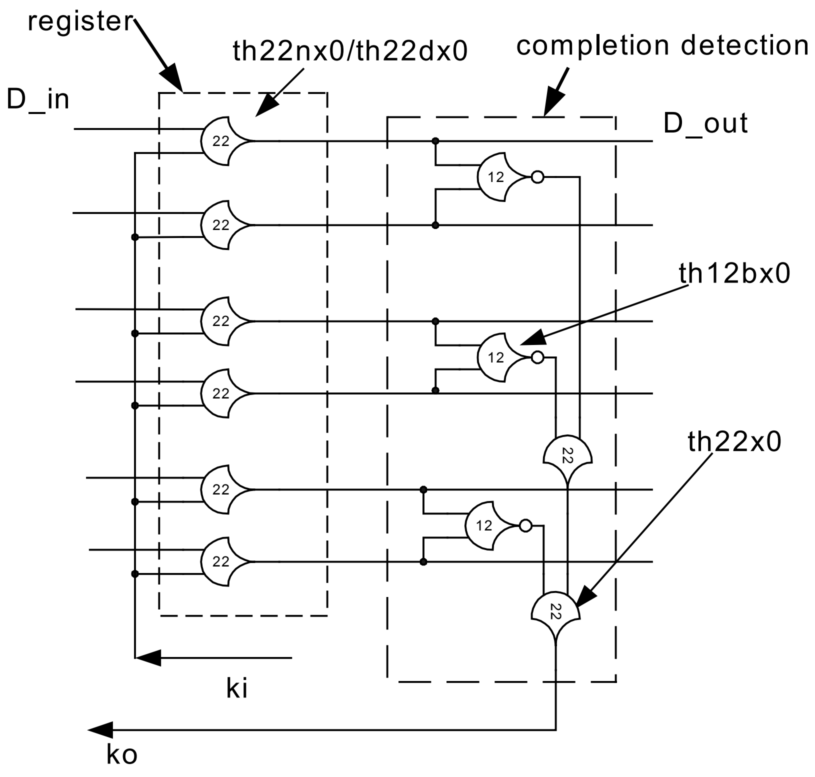
register
th22nx0/th22dx0
completion detection
D_in
D_out
th12bx0
th22x0
22
12
ki
ko

Fig. 4

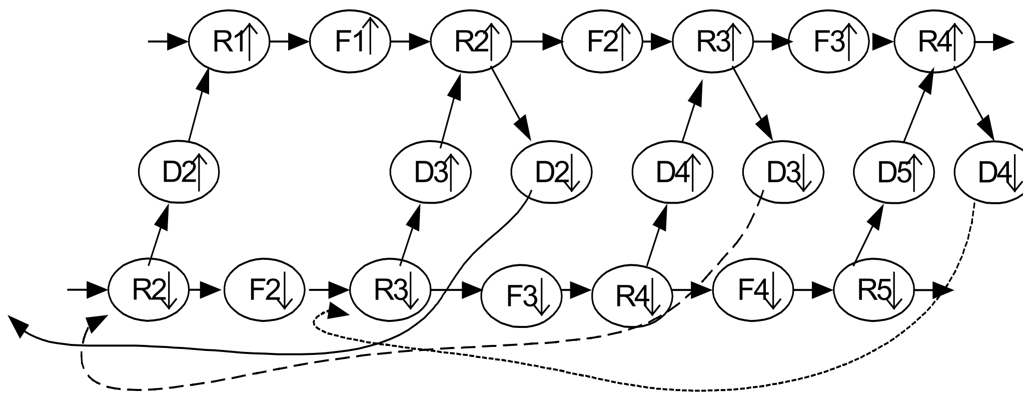

Fig. 5 a)

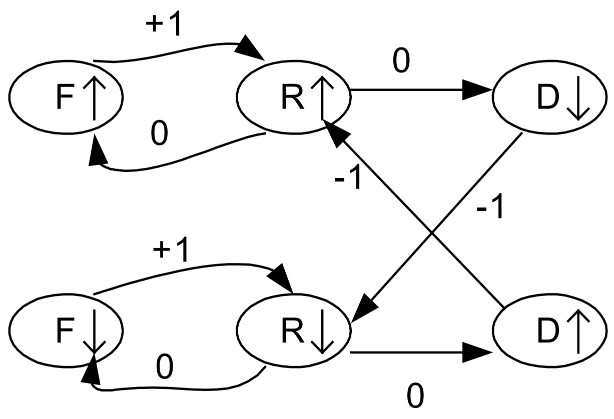

Fig. 5 b)

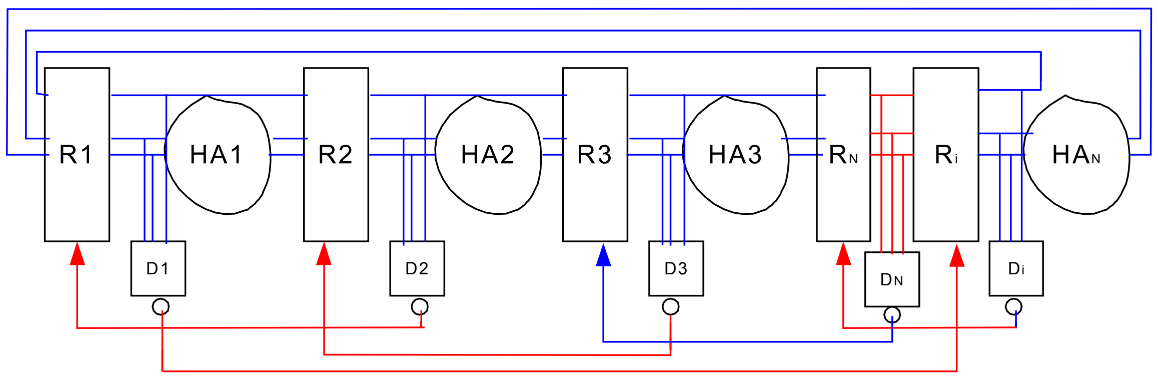

Fig. 6

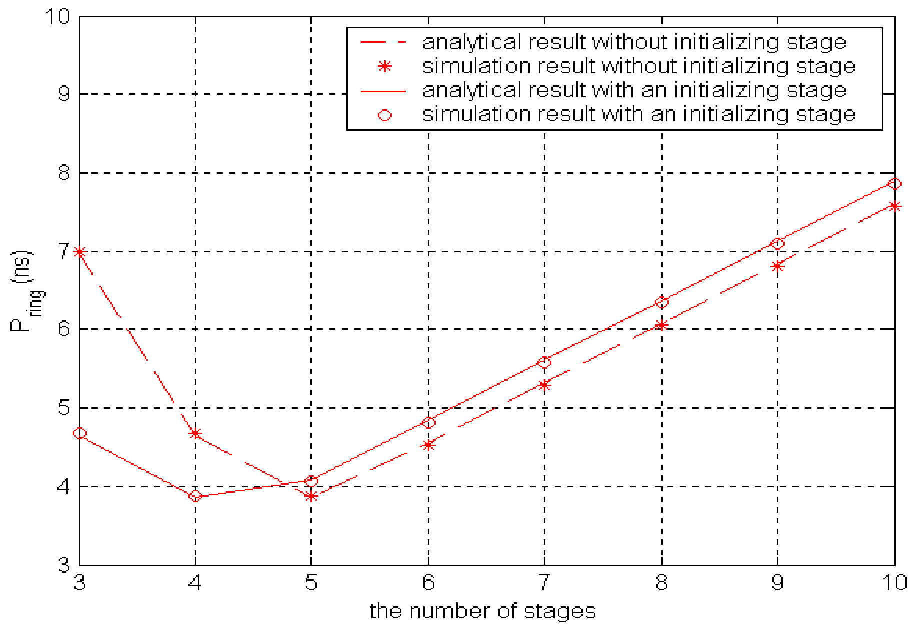
analytical result without initializing stage
simulation result without initializing stage
analytical result with an initializing stage
simulation result with an initializing stage
P_{ring} (ns)
the number of stages
3
4
5
6
7
8
9
10

Fig. 7

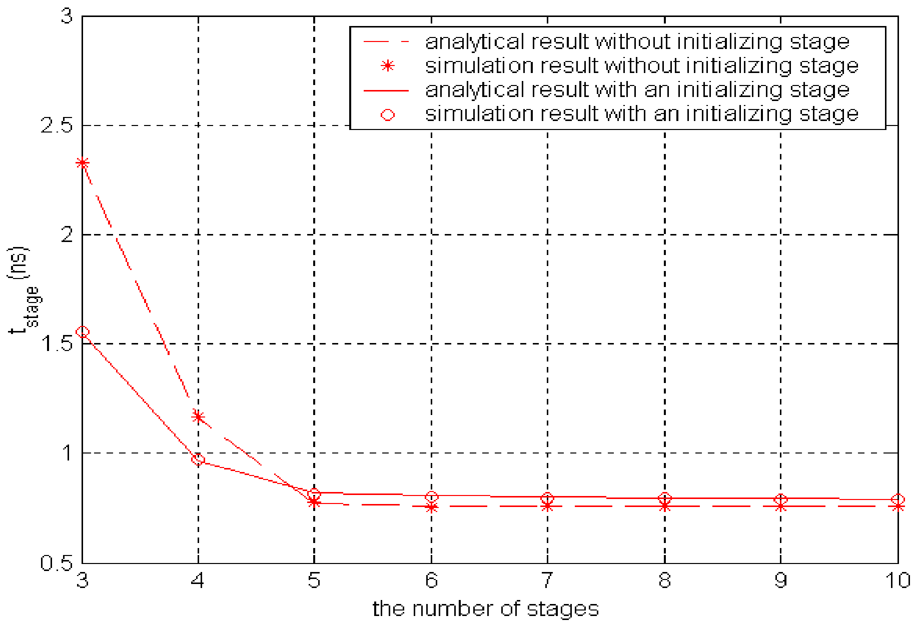
analytical result without initializing stage
simulation result without initializing stage
analytical result with an initializing stage
simulation result with an initializing stage
t_{stage} (ns)
the number of stages
0.5
1
1.5
2
2.5
3

Fig. 8

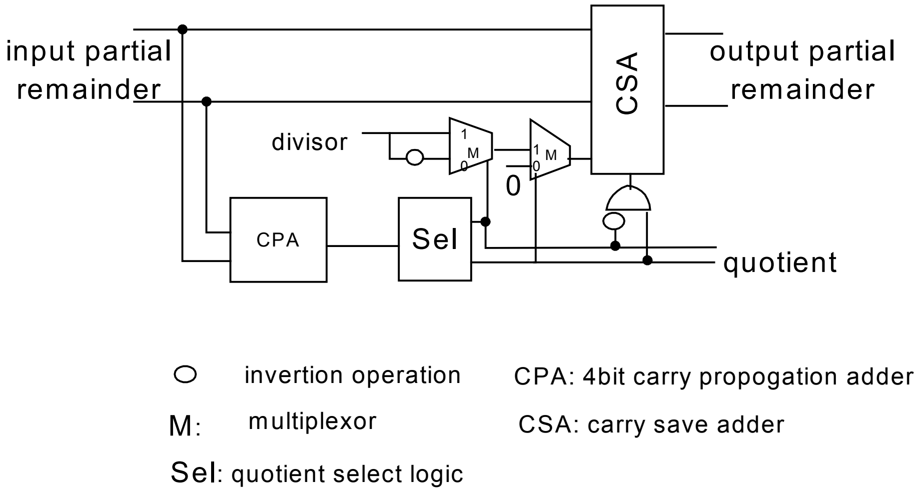
input partial remainder
output partial remainder
CSA
divisor
M
1
0
0
CPA
Sel
quotient
○ invertion operation
M: multiplexor
Sel: quotient select logic
CPA: 4bit carry propogation adder
CSA: carry save adder

Fig. 9

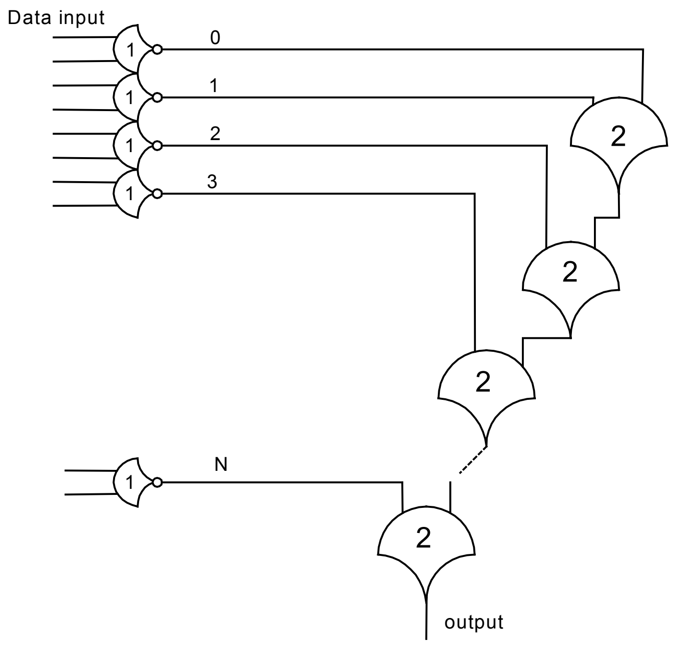
Data input
0
1
2
3
1
2
N
output

Fig. 10 a)

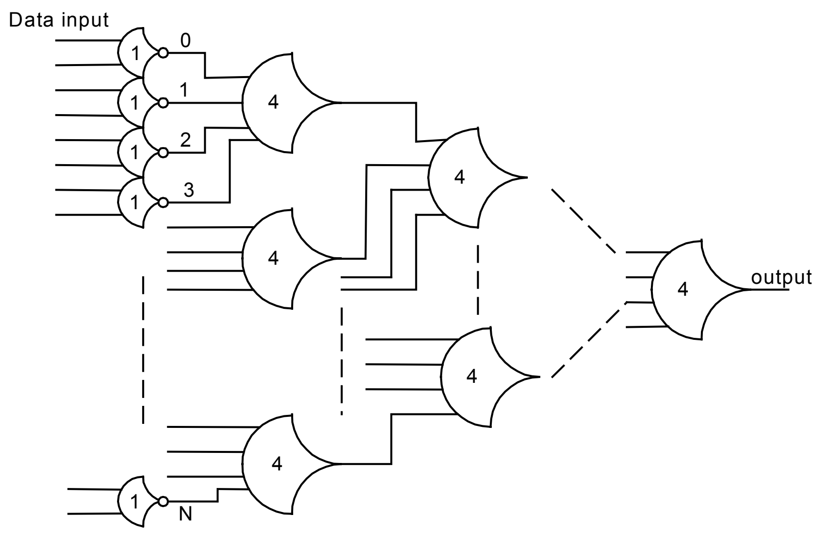

Fig. 10 b)

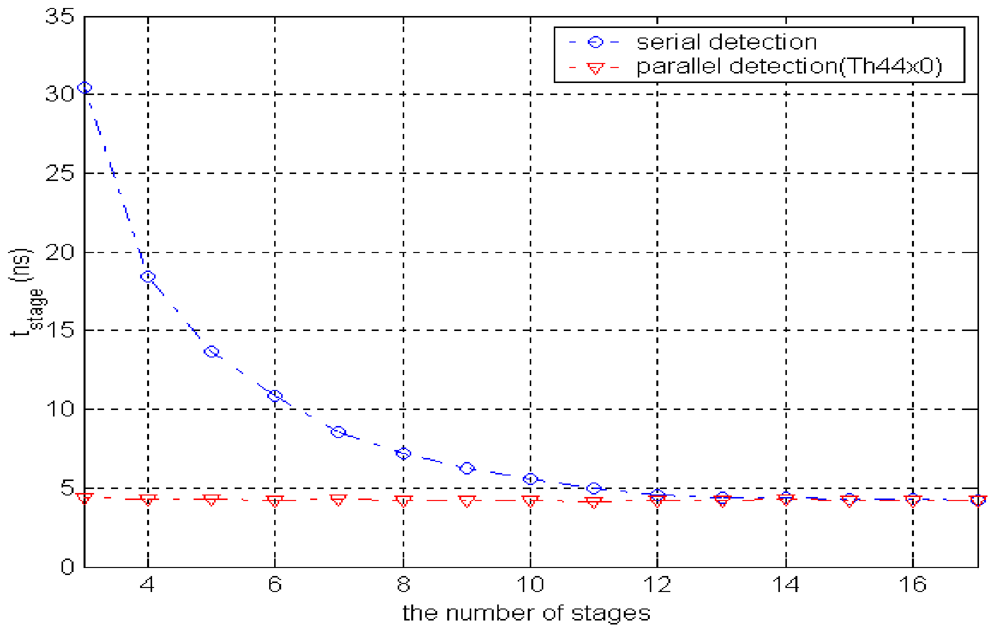

Fig. 11