Design for Self-Checking and Self-Timed Datapath

Jing-ling Yang,
Department of Electrical and Electronic Engineering
The University of Hong Kong
Chiu-sing Choy, Cheong-fat Chan and Kong-pong Pun
Department of Electronic Engineering
The Chinese University of Hong Kong
jlyang@eee.hku.hk, cschoy, cfchan, kppun@ee.cuhk.edu.hk

ABSTRACT

This work examines the inherent self-checking property of a latch-free dynamic asynchronous Datapath (LFDAD) using differential cascode voltage switch logic (DCVSL).

Consequently, a highly efficient self-checking (SC) dynamic asynchronous datapath architecture is presented. In this architecture, no hardware needs to be added to the datapath to achieve self-checking. The presented implementation is efficient in terms of speed and area and represents a new approach to fault-tolerant design.

Keywords: Self-checking, asynchronous datapath, differential cascode voltage switch logic, dynamic circuits

1. INTRODUCTION

A typical self-checking circuit [1] includes a functional circuit and a checker circuit. The output of the functional circuit is encoded using an error-detecting code. The corresponding checker circuit checks for code words at the output. If a non-code word is found, an error is indicated. Additionally, any fault that affects the checker itself should also be indicated. Given this property, a self-checking circuit must meet the requirements of online testing, to detect readily a fault, if the fault causes the circuit to malfunction.

The absence of a global clock signal in asynchronous circuits makes such circuits more difficult to test using current synchronous test techniques [2]. However, the self-synchronous nature of some asynchronous circuits yields self-checking property such that the circuit detects particular types of faults during operation [3].

Martin [4] demonstrated that a single stuck-at fault in a non-redundant delay-insensitive circuit causes a transition either not to occur and/or causing transition on output to become enabled in an illegal state.

Varshavsky [5] showed that semi-modular circuits are totally self-checking, TSC, with respect to output stuck-at faults (OSAFs), if the faults do not cause the circuit to enter an invalid state (exitory faults) or an alternating cycle of valid states (substitutional faults).

Beerel [3] related the semi-modular circuit testability results [5] to speed-independent circuits and extended the study of OSAFs to certain input stuck-at faults (ISAFs).

These results are impressive and have led to new approaches to testing asynchronous circuits, based on the tendency of such circuits to halt in the presence of stuck-at fault. However, such self-checking abilities are limited to within the scope of asynchronous control circuits.

This work, on the other hand, addresses the inherent self-checking property of a latch-free dynamic asynchronous datapath (LFDAD) [6,7,8]. This type of datapath is emphasized for various reasons. To design a self-checking static asynchronous datapath, additional hardware needs to be added to the datapath to achieve the self-checking property for latch/register and computation block [9]. This problem also exists in self-checking latch-based dynamic asynchronous datapath design (LBDAD) [10].

Asynchronous control circuits are known to have some inherent self-checking properties [2,3,4,5], such that circuit activities cease in the presence of single and multiple stuck-at faults at gate inputs and outputs. Some authors have also shown that single stuck-at, stuck-closed and stuck-open faults in most transistors in any DCVSL circuit [11] yields either correct values or loss of complementarity at the outputs. Such a property holds in multi-level DCVSL circuits in addition to single-level DCVSL circuits [12]. This inherent fault-secureness property of DCVSL circuits leads to the design of efficient, strongly self-checking DCVSL circuits [13].

LFDAD, comprised exclusively of asynchronous handshake control circuits and DCVSL computational blocks, thus has the greatest possibility to be self-

Proceedings of the 21st IEEE VLSI Test Symposium (VTS'03)
1093-0167/03 $17.00 © 2003 IEEE

checking among all other asynchronous datapaths. This work focuses on determining the self-checking property of LFDAD and considers this property to simplify the design of self-checking LFDAD.

2. NEW LFDAD DESIGN

Figure 1 shows a new LFDAD design for self-checking. Each pipeline stage consists of a dynamic DCVSL function block and a handshake cell.

The precharge/evaluate control signal, CP, of each stage is controlled by the output of its neighbor stage's completion detector. See Figure 1.

The completion detector indicates validity or absence of data at the outputs of the associated function. The N-pair output data from each function block are connected to the N-pair input dynamic dual-rail code checker (DDCC), which outputs a complementary dual-rail pair when all the input signal pairs are complementary. See Section 2.4. Then, a dynamic single-rail XOR gate is used to generate the final completion signal.

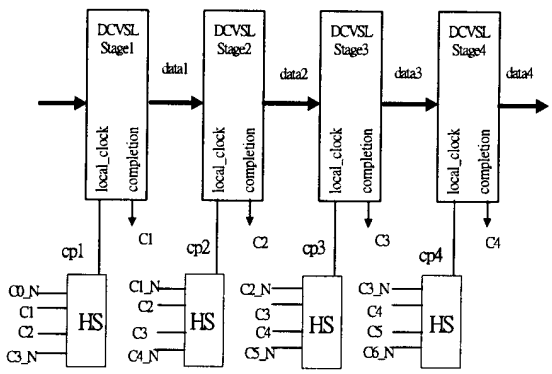


Figure 1: Structure of the New Pipeline

2.1. Timing Protocol and New Handshake Cell

The new LFDAD works in the following way. First, after initialization, a current stage, such as N, is allowed to enter the evaluation phase, which consists of two internal steps, Enable & Evaluation and Evaluation-Hold. During the Enable & Evaluation step, the current stage processes the valid data at the input. After the current stage has finished the evaluation, it enters the Evaluation-Hold step, and this step remains until the following stage has finished its evaluation. Second, the current stage enters the precharge phase when the N-1 stage finishes precharging, the N and N+1 stages finish evaluation, and ends the precharge phase and goes into Evaluation phase when the N-1 stage finishes evaluation, the N stage finishes precharge and the N+2 stage finishes evaluation. See Figure 2.

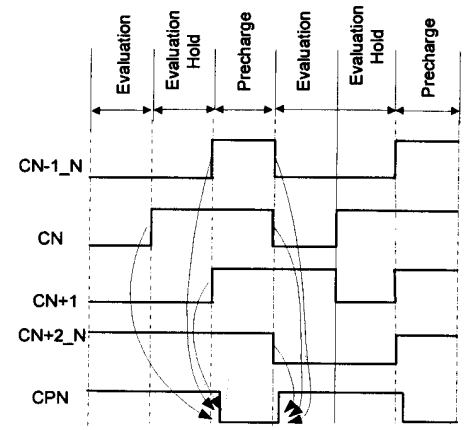


Figure 2: Operating Phase of the Handshake Cell

A new handshake cell is designed to meet the timing requirements. See Figure 3. The handshake cell is a domino-style logic cell.

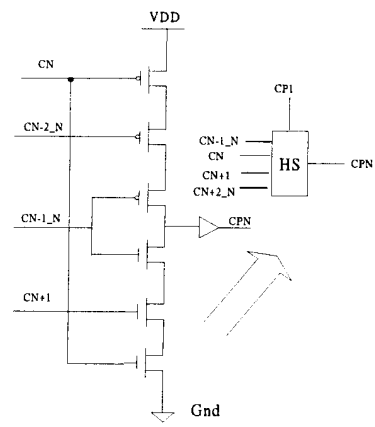


Figure 3: The New Handshake Cell

In Figures 2 and 3, CN and CN+1 are completion signals from stages N, and N+1 of a pipeline, respectively. CN-1_N and CN+2_N are the compliment of CN-1 and CN+2. The operation of a pipeline stage is divided into two phases: (1) Evaluation phase which includes – Enable & Evaluation and Evaluation-Hold two steps; (2) Precharge phase. In the Enable & Evaluation step, the current stage processes the valid data at the input. After the current stage has finished the evaluation, it enters the Evaluation-Hold step. In this step, the input data may become invalid but the output should be held for the use in the evaluation step in the next stage.

Then, the current stage enters the precharge phase as is necessary for DCVSL. After precharging, the current stage enters the Enable & Evualation step again. In this step, the current stage waits for a new valid data to appear at the input.

2.2. Dynamic Dual-Rail Code Checker

DDCC is needed to merge the multiple dual-rail pair into a single dual-rail pair to obtain the completion signal at each pipeline stage. The DDCC functions as follows. Outputs (Z, $\overline{z}$) are complementary if all input pairs are complementary, and if any input pair is 0,0 or 1,1 the outputs (Z, $\overline{z}$) take on that same value. Figure 4 shows the circuit implementation of a 4-pair input DDCC. The 4-pair input DDCC is actually a 4-pair dynamic dual-rail XOR gate.

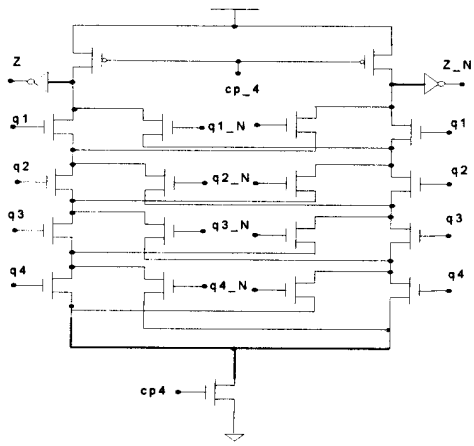
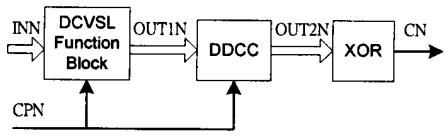
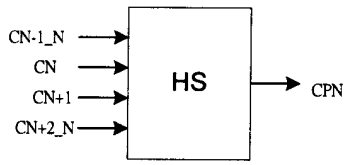


Figure 4: Circuit Implementation of DDCC



(a) Computation Block of Stage N



(b) Handshake Circuits of of Stage N

Figure 5: Building Blocks of Pipeline Stage N

3. INHERENT SELF-CHECKING ABILITY

This section considers the inherent self-checking property of LFDAD. For simplicity, only gate-level input or output stuck-at fault is assumed to occur at a particular time.

Figure 1 shows the implementation of a LFDAD, in which each pipeline stage involves a computation block and a handshake circuit. See Figure 5.

If a one stuck-at fault occurs in stage N, it may occur on INN, OUT1N, OUT2N, CN or CPN. Section 3.1 discusses the fault effect on the data related signals, such as INN, OUT1N, and OUT2N. Section 3.2 discusses the fault effect on the control related signals, such as CN and CNP.

3.1. Fault Effect on Computation Block

Each computation block has three types of element – the DCVSL function block, the DDCC and the XOR gate.

Consider the DCVSL part, which exhibits the following characteristics: (1) During the precharge phase, input data do not affect the output value. Therefore independent of the data at the input lines, the outputs (F and $\overline{F}$) are precharged to low. (2) In the evaluation phase, since the outputs are complimentary, a single stuck–at fault at the inputs or outputs lines of any DCVSL circuit results in either a correct value or in the loss of complementarity at the outputs.

DDCC is a type of DVCSL circuits that has the following special property. The outputs (Z, $\overline{z}$) are complementary if all of the input pairs are complementary; if any input pair is 0,0 or 1,1 then the outputs (Z, $\overline{z}$) take on that same value.

A XOR gate will always map a pair of complementary inputs to output 1, and any non-complementary inputs to output 0.

Figure 7 shows a computation block cascaded by a DCVSL function block, a DDCC and an XOR.

Under fault-free conditions, a computation block alternates between precharge and evaluation. In the precharge phase, OUT1N, OUT2N and CN are suposed to output 00 pairs, 00, and 0, respectively. In the evaluation phase, OUT1N, OUT2N and CN should output 01/10 pairs, 01/10 and 1, respectively.

If a stuck-at fault in the computation block occurs in the precharge phase and (1) if the fault occurs in INN, then, because a DCVSL circuit in the precharge phase does not respond to any input data, such a fault does not affect output. (2) If the fault occurs in OUT1N, one spacer 00 pair cannot be reached. After DDCC, no 00 pair can be generated; after the XOR gate, no 0 can be generated, and so CN is stuck-at 1. (3) If the fault occurs in OUT2N, then no 00 pair can

be reached. After the XOR gate, no 0 can be generated, and so CN is stuck-at 1. (4) If the fault occurs in CN, then 0 cannot be reached, and CN is stuck-at 1.

If a stuck-at fault occurs in the evaluation phase, and (1) if the fault occurs in INN, one data pair of OUT1N loses complementarity. After DDCC, no 01/10 pair can be generated; after the XOR gate, no 1 can be generated, thus CN is stuck-at 0. (2) If the fault occurs in OUT1N, one data pair can not reached 01/10. After DDCC, no 01/10 pair can be generated; after the XOR gate, 1 cannot be generated, thus CN is stuck-at 0. (3) If the fault occurs on OUT2N, 01/10 pair cannot be reached. After XOR gate, 1 cannot be generated, thus CN is stuck-at 0. (4) If the fault occurs on CN, 1 cannot be reached; CN is stuck-at 0.

In summary, for computation block N, (1) a stuck-at fault in INN will result in a CN stuck-at 0 fault in evaluation phase; (2) a stuck-at fault in OUT1N and OUT2N will result in a CN stuck-at 1 fault in prechage phase; and a CN stuck-at 0 fault in evaluation phase.

3.2. Fault Effect on Handshake Gate

Consider the handshake circuit, shown in Figure 4. Its transition diagram can be expressed as:

CPN↓= CN-1_N↑·CN↑· CN+1↑

CPN↑= CN-1_N↓·CN↓·CN+2_N↓

These two expressions show that (1) if CN-1_N is stuck-at 0, CPN↓ will not occur; if CN-1_N is stuck-at 1, CPN↑ will not occur. Therefore, an stuck-at fault in N-1 will stop stage N. (2) If CN is stuck-at 1, CPN↓ will not occur; if CN is stuck-at 0, CPN↑ will not occur. Therefore a stuck-at fault in N will also stop stage N.

Here, we can conclude that a stuck-at fault in stage N will stop the operation of stages N, and N+1, which will further halt the entire pipeline.

3.3. LFDAD is Self-Checking

Asynchronous circuits that halt for all faults are called self-checking [3,5,10]. Here, the notion of self-checking differs from that conventionally used for synchronous systems. Synchronous systems use special codes to ensure that the circuit produces an illegal output in response to a fault. A separate circuit can then detect the illegal output code and generate an error signal. The overhead in terms of area is quite large, often a factor of two.

An asynchronous circuit Y=F(x) is called self-checking (SC) at stuck-at fault with respect to a stuck-at fault p of class P, if for all $p \in P$, the fault will stop the pipeline.

Section 3.1 established that in computation block N, any stuck-at fault results in CN either stuck-at-0 or stuck-at-1 fault.

Section 3.2 established that a stuck-at fault on CN will cause both stage N and N+1 stop working which will further stop the entire pipeline.

For any stuck-at fault in LFDAD, only one result will happen that is the fault will stop the pipeline. SC conditions are met, therefore LFDAD is SC.

4. A DESIGN CASE STUDY

This section presents a highly efficient 8-bit SC asynchronous divider. This design aims to verify the feasibility and the hardware efficiency of the proposed SC scheme.

4.1. SC Divider Architecture

Using the proposed SC scheme, an 8-bit SC asynchronous divider is designed. Figure 6 shows the top-level structure of the SC asynchronous divider. Figure 7 shows the basic configuration of a division stage [14].

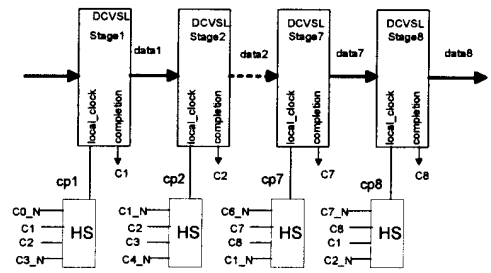


Figure 6: Top Level Structure of the SC Asynchronous Divider

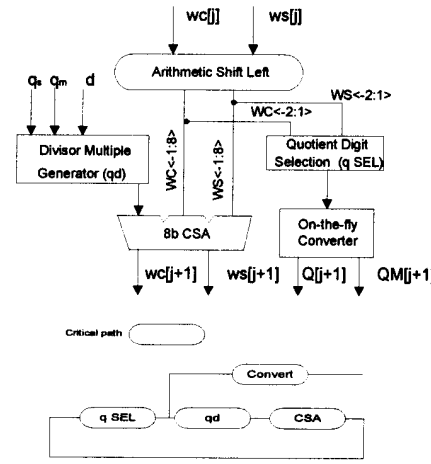


Figure 7: Structure of Radix-2 Division Stage and the Corresponding Timing Diagram

4.2. Implementation and Results

An 8-bit SC asynchronous divider was implemented using AMS CMOS 0.6um technology. The chip size was around 1.66mm x 1.70mm. The design was simulated using SPECTRA under typical conditions. The simulation results show that the SC asynchronous divider works with a clock period of 8.5ns, (approximately 117 MHz), and the latency for 8-bit quotient–digit generation is about 19ns (approximately 52.6 MHZ). Figure 8 shows the simulation results. (a) The local clock signal from eight stages. (b) The completion signal from eight stages. (c) The computed results for the divider. (d) Results obtained by DDCC.

The manufactured chip was tested by the IMS XL-60 Logic Master. The test results reveal that if the acquired quotient is correct, then the DDCC outputs a dual-rail signal; otherwise, if the acquired quotient is not correct, then the DDCC outputs either a 00 signal or a 11 signal. Or, if the SSC divider has stopped, the DDCC will signal a constant data pair.

4.3. Fault Simulation

Two tools are needed to simulate a fault, one is a pattern generation tool, and the other is a fault simulation tool. Most commercial simulators are currently designed for static synchronous circuits, while this work needs a fault simulation tool used to dynamic asynchronous circuits. Accordingly, this work relies primary on the analysis to validate the SC ability of a dynamic asynchronous datapath. Only some selected faults are simulated manually. The selected faults are chosen for their representativeness.

Inserted fault type: stuck-at 0 or 1.

Fault location: (1) for the handshake control: the completion and the clock signal lines of eight stages (2) for the DCVSL computation block: the data signal line of eight stages.

The fault is added to a fault point via a XOR gate, shown in Figure 9.

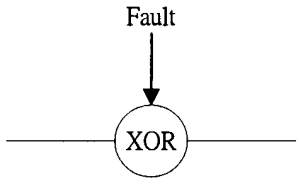


Figure 9: Error Insertion Point

The stuck-at fault is inserted in a simulation cycle operated for 45ns. The selected stuck-at faults yield following responses. (1) The SC divider stops work. (2) DDCC signals a constant data pair.

5. CONCLUSION

This work analyzes the potential SC capacities of LFDAD, and proposes a corresponding SC scheme. The proposed SC structure is validated using a SC 8-bit asynchronous divider design.

The fact that no undetected faults occur in the simulation indicates that the SC abilities of this type of design is satisfactory.

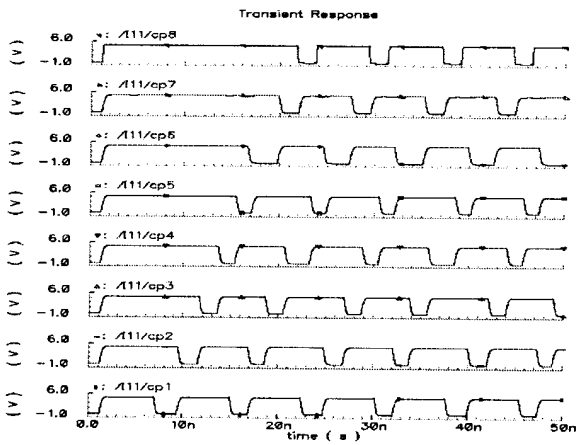
6. ACKNOWLEDGEMENT

The work reported is supported by a Hong Kong SAR government RGC, earmarked grant no. CUHK 4172/97E and a Direct Grant no. 2050246.

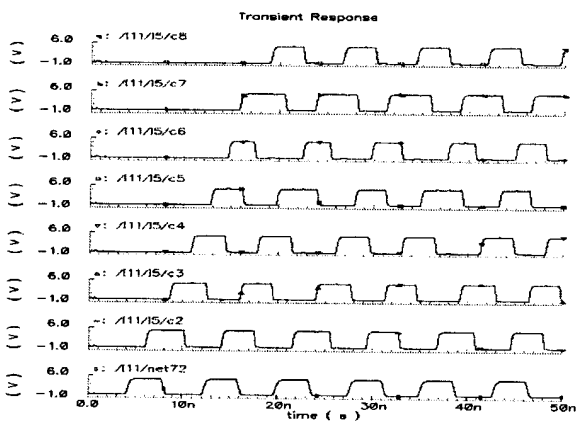
7. REFERENCE

1. John Wakerly, " Error Detecting Codes, Self-Checking Circuits and Applications", Elsevier North-Holland, Inc., 1978.
2. H. Hulgaard, S.M. Burns, and G. Borriello, "Testing Asynchronous Circuits: A Survey", Integration, the VLSI J., vol. 19, pp.111-131, Nov. 1995.
3. P. Beerel and T.Y. Meng, "Semi-Modularity and Testability of speed-independent Circuits", Integration, TheVLSIJ., Vol. 13, Sept, 1992.
4. Alain J. Martin and Pierter J. Hazewindus, "Testing Delay-insensitive Circuits", Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference, pp. 118-132. The MIT Press, Cambridge, MA, 1991.
5. V.I. Varshavky, editor, " Self-timed Control of Concurrent Processes, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
6. Ted E. Williams, Mark A. Horowiz, "A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider", IEEE Journal of Solid-State Circuits, VOL.26, No.11, pp. 1651-1661, November 1991.
7. Gensoh Matsubara, Nobuhiro Ide, "A Low Power Zero-Overhead Self-Timed Division and Square Root Unit Combining a Single-Rail Static with a Dual-Rail Dynamic Circuits", Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and System, 1997, pp.198-209.
8. Montek Singh and Steven M. Nowick, "High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapath", Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and System, 2000, pp.198-209.
9. Ilana David, Ran Ginosar, Michael Yoeli, "Self-Timed is Self-Checking", Journal of Electronic

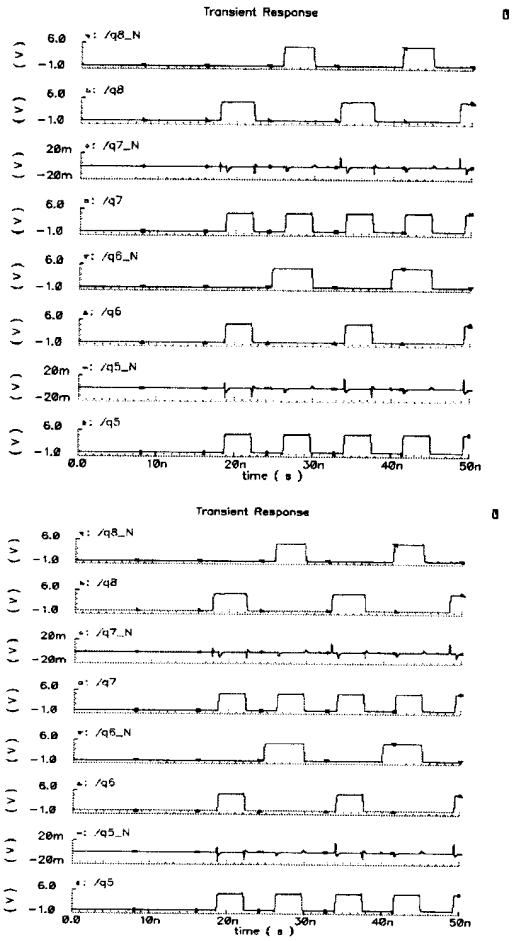
Testing: Theory and Application, 6,219-228, 1995.
10. Divid A. Rennels and Hyeongil Kim, "Concurrent Error Detection in Self-timed VLSI", FTCS-24, Austin, Texas, June 15-17,1994.
11. Chu, LK. M., and Puffrey, D.L. : "Design Procedure for Differential Cascade Voltage Switch Circuits", IEEE J. Solid-State Circuits, 1986, 21, (6), pp. 1082-1087.
12. Niraj K. Jha, "Testing for Multiple Faults in Domino-CMOS Logic Circuits", IEEE Transaction on Computer-aided Design, Vol 7, No. 1, January 1988.
13. Niraj K. Jha, "Strong Fault-Secure and Strong Self-checking Domino-CMOS Implementations of Totally Self-Checking Circuits", IEEE Transactions on Computer-aided Design, Vol., 9, NO. 3. March 1990.
14. Miloš D. Ercegovac, Tomás Lang, Division and Square Root: Digital Recurrence Algorithms and Implementations, Kluwer Academic Publishers, Norwell, Massachusetts, 1994.



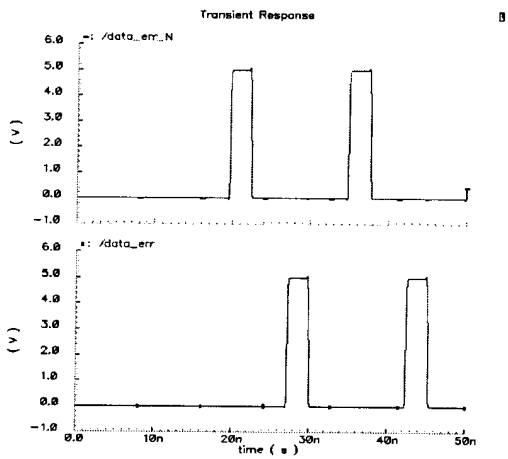
(a) Clock Signal of each of the Eight Stages



(b) Completion Signal from Eight Stages



(c) Computation Results of the Divider



(d) Outputs from Dual-Rail Checker

Fig.8: The Simulation Result of SPECTRA (only part of waveform is shown)