Title	A totally Self-Checking Dynamic Asynchronous Datapath
Author(s)	Yang, JL; Choy, CS; Chan, CF; Pun, KP
Citation	The 11th Asian Test Symposium Proceedings, Guam, USA, 18-20 November 2002, p. 27-32
Issued Date	2002
URL	http://hdl.handle.net/10722/46401
Rights	Creative Commons: Attribution 3.0 Hong Kong License

A Totally Self-Checking Dynamic Asynchronous Datapath

Jing-ling Yang
Department of Electrical and Electronic Engineering,
The University of Hong Kong
jlyang@eee.hku.hk

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

Abstract

This paper investigates the inherent totally self-checking (TSC) property of one type of dynamic asynchronous datapath based on Differential Cascode Voltage Logic (DCVSL). As a result, a totally self-checking dynamic asynchronous datapath architecture is proposed. It is simpler than other similar approaches and represents a new approach to fault tolerant design.

Keywords: Totally self-checking, asynchronous datapath, differential cascade voltage switch logic, divider

1. Introduction

A totally self-checking, TSC, circuit is a circuit that has self-test and fault-secure property for a normal input set N and faulty set F [1]. A TSC circuit should consist of a TSC functional part and a TSC checker. The inputs and outputs of the functional part are encoded using some suitable codes. The checker checks for code words at the outputs. If a non-code word is found, an error is indicated. In addition, a fault produced by a checker itself should also be detected. With the above characteristic, a TSC circuit well meets the requirements of online testing, where a fault, which is either permanent or transient, can be readily detected if the fault causes circuit malfunction.

It is known that delay insensitive circuits, one type of asynchronous control circuits, have the inherent self-check properties, that is all circuit activities cease (dead lock) with respect to single and multiple stuck-at faults at gate outputs [2,3,4]. Some authors have also showed that single stuck-at, stuck-closed, and stuck-open faults in most transistors of any Differential Cascode Voltage Logic (DCVSL) circuits result in either correct values or in loss of complementarity at outputs [5].

Typically a concurrent error detection asynchronous datapath with DCVSL computation block uses a TSC dual-rail code error indicator to check error from DCVSL computation blocks, and a non TSC RC time-out counter at each computation stage to indicate any asynchronous control deadlock error [6]. However this scheme is inefficient because the use of non TSC RC time-out counter in every pipeline stages will lead to (1) large silicon. (2) TSC goal can not be achieved.

This paper, on other hand, proposes a highly efficient TSC asynchronous datapath scheme with only one TSC dual-rail code error indicator at its final stage. In other words, according to the new scheme, a TSC asynchronous datapath with DCVSL computation block can be achieved simply by adding a dual-rail code error indicator at its final computation block.

2. TSC Dynamic Asynchronous Datapath

In a TSC synchronous circuit, input and output are coded as code words. If there is effective fault exists in the circuit, the circuit will output a non-code word that is distinguishable from the fault free circuit that outputs a code word.

Delay insensitive circuits have inherent TSC ability because they will deadlock with respect to single and multiple stuck-at faults at gate outputs. That is the presence of stuck-at faults may cause the circuits leave their normal cycle of states into a deadlock state. Here the conventional code words and non-code words are replaced by the circuits' valid states and deadlock states. Further studies on properties and relations among these valid states and deadlock states, we found that the special deadlock property of a Latch-Free Dynamic Asynchronous Datapath (LFDAD) is well suited to implement TSC design.

2.1. Latch-Free Dynamic Asynchronous Datapath

DCVSL circuits have two interesting behaviors. (1) During the precharge phase, input data has no effect on the output value. That means, no matter what value of data appears at the input lines, the outputs (F and $\overline{F}$) will be precharged to low. (2) During the evaluation phase, DCVSL gate begins evaluation as soon as the input data are valid (01 or 10). When the input data are not valid (00), the DCVSL gate remains at the precharge state.

These two behaviors lead to two advantages of a dynamic asynchronous pipeline over it static counterpart, namely latch free and simpler handshake protocol, and readiness to implement a TSC scheme.

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE

A LFDAD can work in the following fashion. First, after initialization, a current stage is allowed to go into the evaluation phase and remains in the evaluation phase (Hold) until the next stage has finished evaluation. Second, the current stage goes into the precharge phase when the next stage finishes evaluation, and ends the precharge phase when the second next stage finishes evaluation.

The handshake cell is a domino-style logic cell, see Figure 1. Figure 2 shows its timing diagram.

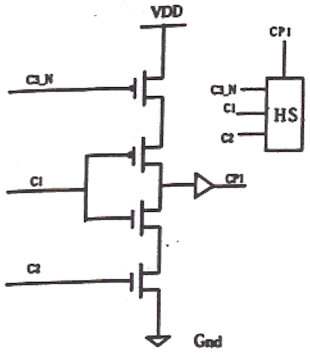


Figure 1: The Handshake Cell

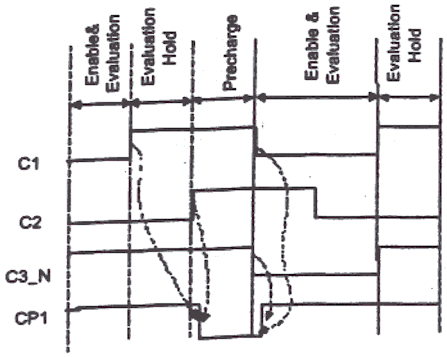
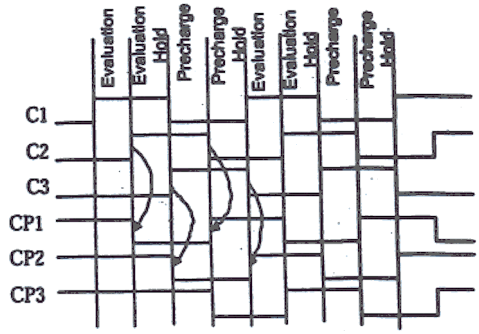


Figure 2: Operation Phase of the Handshake Cell

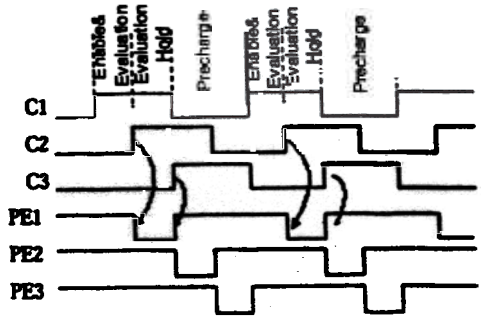
In Figure 1 and Figure 2, C1, C2 and C3 are completion signals from stage1, stage2 and stage3 of a pipeline respectively, C3_N is the compliment of C3.

The operation of a pipeline stage is divided into 2 phase that include 3 steps: Evaluation phase, which includes Enable & Evaluation and Evaluation Hold two internal steps, and 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 will enter the Evaluation Hold step. In this step, the input data may become invalid but the output should be held for the whole Enable & Evaluation step for the next stage. After that, the current stage will enter the precharge phase that is necessary for DCVSL. Up to now the current pipeline stage has completed one complete cycle and goes into Enable and Evaluation step of next cycle waiting for a new valid data to appear at the input.

Comparing with the handshake arrangement of Williams [7], this handshake cell is faster due to its protocol that allows the precharge signal be removed before the arrival of a valid data. Comparing with the handshake arrangement of Matsubara's [8], although both protocols allow the precharge signal be removed before the arrival of a valid data, the handshake cell is faster due to its simplicity. And more importantly, this handshake protocol is robust because it has no timing assumption that make it readiness to TSC implement, while Matsubara's pipeline has to satisfy the precharge width requirement.



(a) Williams' Pipeline Handshake Timing



(b) Matsubara's Pipeline Handshake Timing

Figure 3: Signal Graphs of Different Handshake Cells

Figure 4 shows the implementation of a pipeline using the handshake cell. Each pipeline stage is composed of a dynamic DCVSL function block, completion detector and handshake cell. The completion detector indicates validity or absence of data at the outputs of the associated function. Each pair of data is checked by an XOR gate, and then uses a C-element to generate the final completion signal.

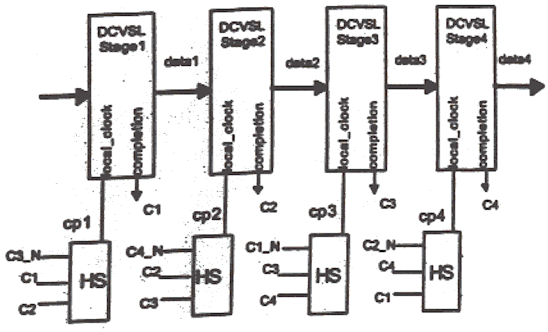


Figure 4: Structure of the Pipeline

2.2. Inherent TSC Abilities on LFDAD

To achieve concurrent error detection in a typical asynchronous datapath with DCVSL computation block, the most conventional method is treating them as two parts, using a TSC dual-rail code error indicator to check error from DCVSL computation blocks, and a non TSC RC time-out counter at handshake circuits of each computation stage to indicate the asynchronous control deadlock error [6]. But the use of non TSC RC time-out counter will lead to both large silicon and TSC goal can not be achieved.

Each pipeline stage of the LFDAD showed in Figure 4 has the following special properties. (1) Work in cycles consisting of three sequential steps, they are Enable & Evaluation, Evaluation Hold and Precharge. Enable & Evaluation carries out computation according to input, Evaluation Hold keeps the output when next stage carries out computation, and then it goes into the precharge step. (2) A Pipeline stage is one step lagging behind its former stage or one step in advance of its later stage. (3) For each pipeline stage an effective error in its completion signal has no influence on its later stages' control signal, see Figure 2.

The combined effect of these behaviors is: if a stage has an effective error, its following stages will continue to work until they reach the same phase as the error stage. This is the very property to be used to simplify the TSC scheme of LFDAD.

Explore in more detail, we found that: (1) If the error stage is in the Precharge phase, because a DCVSL circuit in the precharge phase does not respond to any input data, its following stages will continue to work until they also finish the precharge phase. (2) When the error stage is in the Evaluation phase, its following stages will continue going to the evaluation phase. Because in the evaluation phase, DCVSL gate begin evaluation as soon as the input data are valid, if its former stage cannot finish evaluation, the following stages remaining in the precharge phases.

Thus we can conclude that for those effective errors either in the evaluation phase or in the precharge phase, their ultimately effects are leaving its following stage in the precharge phase. This conclusion applies to stuck-at, stuck-closed, and stuck-open faults in most transistors of any DCVSL circuits and stuck-at fault handshake control signal.

So, to implement a TSC LFDAD, the only thing we need to do is adding a TSC dual-rail code checker at its final stage.

2.3. A TSC Scheme for LFDAD

Figure 5 shows a proposed top-level structure of a TSC LFDAD. The TSC dual-rail checker can be a multiple input dynamic dual-rail XOR gate. The circuit works in this way: if the acquired output is correct, the data-checker outputs a dual-rail signal, otherwise if the acquired calculation output is not correct, data-checker outputs either a 00 or 11 signals. Or, if the pipeline has been stopped, the error-indicator pair will signal 00.

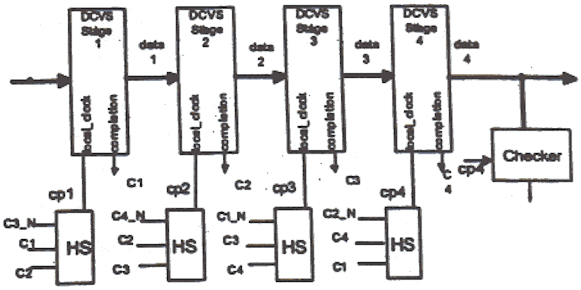


Figure 5: Top Level Structure of the TSC FLDAD

2.4. TSC Analysis of other Similar Approaches

For dynamic asynchronous datapath design, two approaches are popular. One is traditional latch based design [6], and the other is latch free design [7,8,9].

An error detection scheme for a latch based design is demonstrated in [6]. In this scheme the whole datapath is treated as two parts, computation block and control block. A TSC two-rail code error indicator is used to check error from DCVSL computation blocks, and a non TSC RC time-out counter at each computation stage's handshake circuits is used to signal the asynchronous control deadlock error. But the use of non TSC RC time-out counter will lead to both large silicon and TSC goal cannot be achieved.

The TSC LFDAD design introduced in this paper belongs to latch free type design. In this approach, because of the exceptional indication of its deadlock state, the TSC goal can be efficiently realized by adding a TSC dual-rail code checker at its last stage.

For other schemes of latch free approach, no TSC schemes have been proposed. But for those schemes that have made timing assumption in their handshake protocol design, such as [9], the TSC goal may be hard to realize.

3. A Design Case Study

This section presents a highly efficient 8-bit TSC asynchronous divider design. This design aims at verifying the feasibility and the hardware efficiency of the proposed TSC scheme.

Using the TSC scheme proposed, an 8-bit TSC asynchronous divider has been designed. Figure 6 shows the top-level structure of the TSC asynchronous divider with a dual-rail code checker.

To check dual-rail code, a dynamic dual-rail code checker (DDCC) is added at the output of the last stage. Its circuit is showed in Figure 7. The output $(Z, \bar{Z})$ are complementary if all of the input pairs are complementary, and if any input pair is 0,0 or 1,1 the outputs $(Z, \bar{Z})$ take on that same value. If the circuits have stop working, the outputs $(Z, \bar{Z})$ will be kept in 0,0. Note that the DDCC itself is TSC.

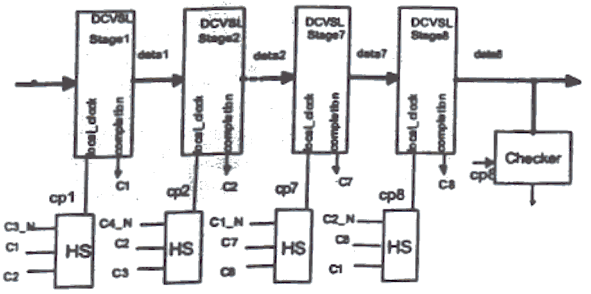


Figure 6: Top Level Structure of the TSC Divider

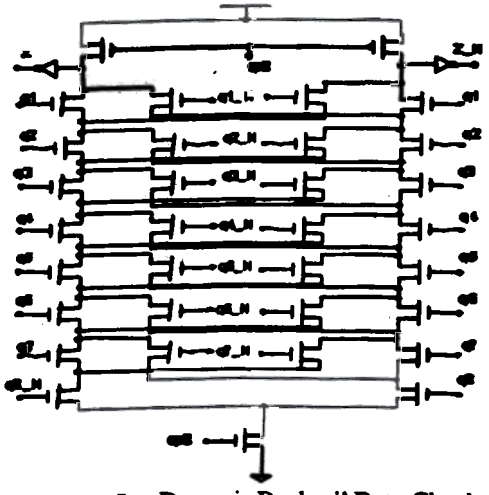


Figure 7: Dynamic Dual-rail Data Checker

The DDCC showed in Figure 7 is actually an 8-bit dynamic dual-rail XOR gate. In layout implementationn, we can compose it by using seven 2-input dynamic dual-rail XOR gates.

4. Results

We implemented an 8-bit TSC asynchronous divider using AMS CMOS 0.6um technology. The chip size is about 1.66mm x 1.70mm. The design was simulated with SPECTRA in typical conditions. Simulation result shows that the TSC asynchronous divider works with a clock period of 8.5ns, (about 117 MHz), and the latency for 8-bit quotient-digit generation is about 19ns (about 52.6 MHZ). Figure 8 shows the simulation result of: (a) Local clock signal from eight stages. (b) Completion signal from eight stages. (c) Computation results of the divider. (d) Results from DDCC.

The manufactured chip was tested with IMS XL-60 Logic Master. The test results show that if the acquired quotient is correct, the DDCC outputs a dual-rail signal, otherwise if the acquired quotient is not correct, DDCC outputs either a 00 signal or 11 signals. Or, if the TSC divider has stopped, the dual-output pair will signal 00.

5. Fault Simulation

To perform fault simulation two tools are needed, one is pattern generation tool, and another is fault simulation tool. Right now, prevailing commercial simulator are designed for static synchronous circuit under permanent fault model, while we want a fault simulation tool used for dynamic asynchronous circuit under transient error model. Because of this reason, this paper mainly relies on the analysis to validate the TSC ability of the dynamic asynchronous datapath, fault simulation is done manually only on some selected error. The selected errors are chosen for their representativeness according to:

1. inserted error type: 0 or 1,
2. error location: control signal or data signal,
3. error instances: before or after completion signal generation.

The error is added to an error point through a XOR gate, showed in Figure 9.

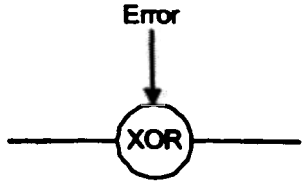


Figure 9: Error Insertion Point

Error with 10ns duration time is inserted in a simulation cycle operated for 45ns. The selected transient errors give four kinds of results. They are: (1) Error has no effect on circuit. (2) Error be tolerated by circuit. (3) Error leads to delay of signal, however the circuits still work correctly because they are delay-insensitive. (4) Error be detected.

With the above experiment, we confirm that the proposed TSC LFDAD scheme is secure against delay errors. It functions correctly under delayed signals caused by the transient errors, whereas in synchronous circuits, the delay faults will lead to circuit malfunction.

6. Conclusion

This paper analyses the inherent TSC abilities of LFDAD, propose a corresponding TSC scheme. In addition, the proposed TSC structure is verified by a TSC 8-bit asynchronous divider design.

The fact that there is no undetected errors be found in the simulation and measurement indicate that the TSC ability of this type of design approach is sound.

7. Acknowledgement

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

8. Reference

[1] John Wakerly, " Error Detecting Codes, Self-Checking Circuits and Applications", Elsevier North-Holland, Inc., 1978.

[2] P. Beerel and T.Y. Meng, "Semi-Modularity and Testability of speed-independent Circuits", Integration, TheVLSIJ., Vol. 13, Sept, 1992.

[3] V.I. Varshavky, editor, " Self-timed Control of Concurrent Processes, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.

[4] Michael J. Liebeit, and Neil Burgess, "Detecting Exitory Stuck-at Faults in Semimodular Asynchronous Circuits", IEEE Transaction on Computers, VOL.48. NO.4. April 1999.

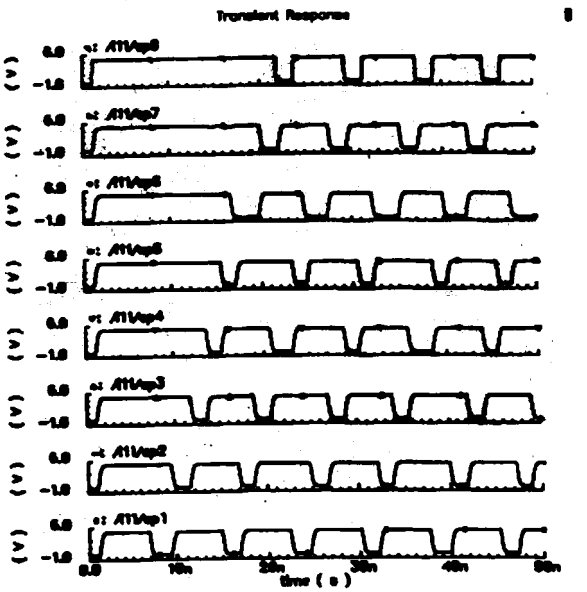
[5] Nick Kanopoulos and Nagesh Vasanthavada, "Testing of Differential Cascode Voltage Switch (DCVS) Circuits", IEEE Journal of Solid-State Circuits, Vol. 25, No. 3, June 1990.

[6] Divid A. Rennels and Hyeongil Kim, "Concurrent Error Detection in Self-timed VLSI", FTCS-24, Austin, Texas, June 15-17,1994.

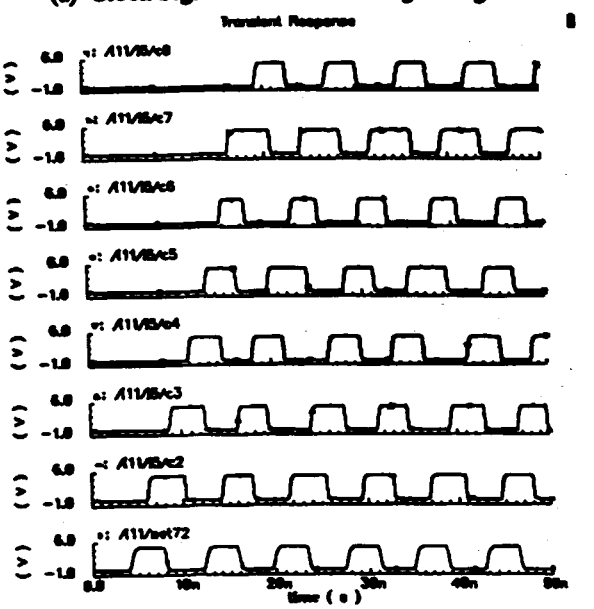
[7] Jing-Ling Yang , Chiu-Sing Choy, Cheong-Fat Cheong, "A Self-Timed Divider Using a New Fast and Robust Pipeline Scheme", IEEE journal of Solid State Circuits. Vol. 36, No. 6, JUNE 2001.

[8] Ted E. Willams, 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.

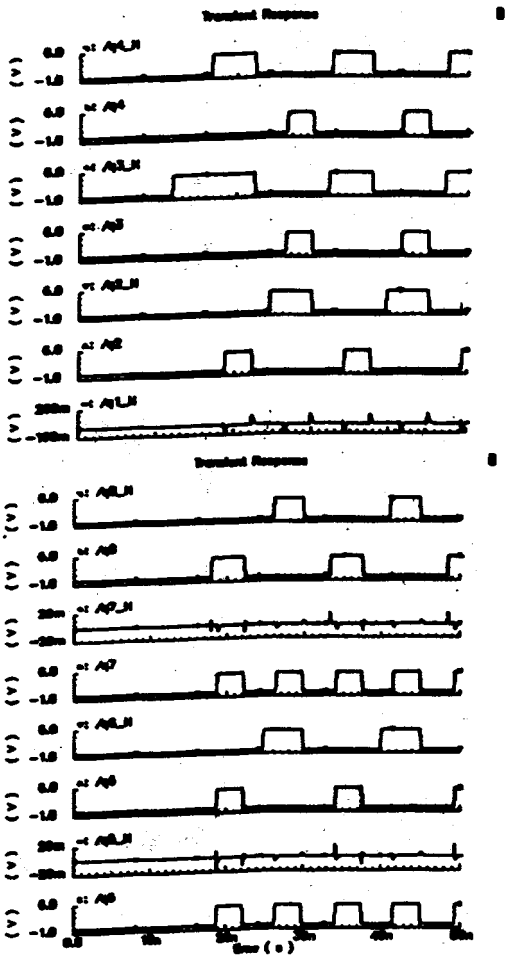
[9] Gensoh Matsubara, Nobuhiro Ide, "A Low Power Zero-Overhead Self-Timed Division and Square Root Unit Combining a Single-Rail Static with a Dual-Rail Dynamic Circuits", Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and System,1997, pp.198-209



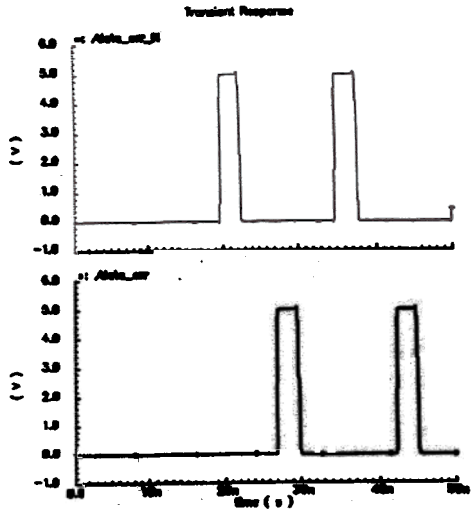
(a) Clock Signal of each of the Eight Stages



(b) Completion Signal from Eight Stages



(c) Computation Results of the Divider



(d) Outputs from DDCC

Figure 8: The Simulation Result of SPECTRA (only part of waveform is showed)