Dual-Rail Asynchronous Pipeline Based on Stochastic Resonance Logic Gates

Gonzalez-Carabarin Lizeth† Tetsuya Asai † and Masato Motomura †

†Graduate School of Information, Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo 060-0814, Japan
Email: lizeth@lalsie.ist.hokudai.ac.jp, asai@ist.hokudai.ac.jp

Abstract— This study proposes an application of stochastic resonance logic gates (SR logic gates), proposed by Asai *etal*. These gates generate basic logic functions in which noise plays an important role in recovering logic operations in the presence of mismatches among devices. These gates work at low power consumption regardless mismatches in comparison with conventional gates. However, one limitation is the timing response, because it is dependent on stochastic processes. Therefore, we propose the design of asynchronous circuits as a suitable application for SR logic gates. In this study, we demonstrate the performance of a three-stage asynchronous pipeline with a dual-rail data encoding. Simulations were performed for a 0.18-μm CMOS technology in SPICE. The simulations show that even in the presence of unpredictable delays, the pipeline successfully accomplishes data transmission. Moreover, experimental results are also shown employing a macro system.

1. Introduction

Cutting-edge technologies for device miniaturization are reducing device size to molecular or atomic scales. However, as such dimensions are reached, side effects are inevitable, such as uncontrolled noise and parameter mismatches. Therefore it has become essential for designers to find new strategies to avoid these drawbacks. Asai *et.al*. has proposed the design of logic gates assisted by noise to recover logic functions in the presence of mismatches (Fig. 1). These gates combine nonlinear systems with hysteresis and the stochastic resonance effect to generate mismatch-insensitive logic circuits that consume low power regardless of mismatches in comparison with conventional logic gates [1]. Simulations and experimental results show that with the introduction of an intentional mismatch, SR logic gates correctly generate the four basic logic functions with low power supply (0.35 V) [2]. However, because the response of the SR logic gates depends on noise, there is a unpredictable delay in their response. Consequently synchronization problems are evident in synchronous circuits based on SR logic gates. Therefore, proper applications for the SR logic gates are found in the field of asynchronous circuit design, in which performance does not depend on a central clock, but on handshake protocols [3]. Asynchronous circuits allow the design of delay-insensitive circuits (DI circuits) because they rely on signals that indicate when a task is accomplished. In this study, we propose the design of an asynchronous pipeline to demonstrate the performance of SR logic gates regardless of the presence of mismatches and unpredictable delays in the response. Here, we demonstrate its performance through simulations and experimental results. This paper is divided as follows: section 2 introduces basic concepts of asynchronous circuit design, as well as the data encoding and protocols used to design the asynchronous pipeline; section 3 presents simulation and experimental results of the basic asynchronous elements, as well as the asynchronous pipeline; and finally section 4 presents the general performance (power consumption and error rate) of the pipeline.

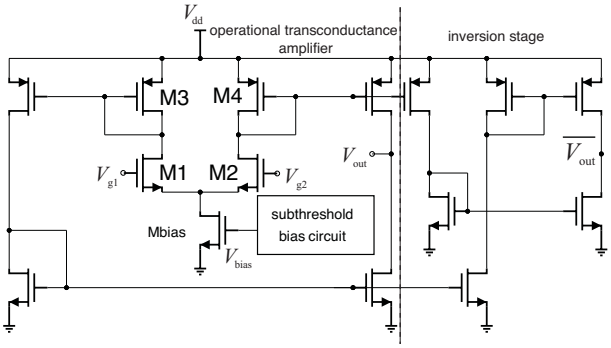

Figure 1: Internal configuration of the SR logic gates

2. Basic concepts of asynchronous circuit design

Although there is no established methodology to design asynchronous circuits, these types of circuits are based on requirement (*req*) and acknowledgment (*ack*) signals between stages. These signals indicate when new data are required (indicated by a change of state of *req* signal) to perform a certain task and when the current task is accomplished (indicated by a change of state of the *ack* signal). Execution of instructions among stages relies on the exchange of these signals.

Further, it is necessary to choose the appropriate data encoding and protocols according to the desired application. Asynchronous circuits are divided into two categories based on their data encoding: single-rail (or bounded) data and dual-rail data encoding. The former is used when the

delays are known precisely, and the latter represents a more robust data encoding because it allows the design of DI circuits and that is the reason it is used for our purposes. Dual-rail data encoding refers to converting one bit to two-bit encoding (Fig. 2), where valid data are given by {0, 1} and {1, 0} which represents 0 and 1 logic respectively. {0, 0} is a spacer between valid data and {1, 1} is not used. Another advantage of the dual-rail data encoding is that it contains the *req* signal implicit in the data (where {0, 1} or {1, 0} represents a *req* signal equal to one); therefore, an additional channel for *req* signal is not necessary.

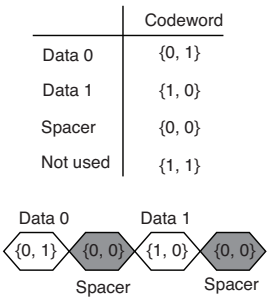

Figure 2: Dual-rail data encoding

Dual-rail data encoding requires a four-phase protocol for data transmission. The four-phase protocol resets the signals *req* and *ack* when the current task has been accomplished, in contrast with the two-phase protocol, where a reset it is not necessary. To implement dual-rail encoding, data are necessary to design dual-rail logic gates based on their single-rail counterparts. In this framework, we show an example of a dual-rail NAND gate based on SR logic gates. The basic diagram is shown in Fig. 3, where the invert operation consists simply of the inversion of output bits. The circuit consists of SR NAND and SR AND gates previoulsy proposed in [2].

On the other hand, in order to design high performance asynchronous circuits, the asynchronous pipeline is one of the main elements [4]. Thus, we present the implementation of an asynchronous pipeline based on a dual-rail data encoding. Figure 4 shows the basic block diagram for a dual-rail based asynchronous pipeline for n stages. Here, data transmission relies on validation of input data. If $L_{0,0}$ and $L_{0,1}$ are either {0, 1} or {1, 0} (*req* equals to one), then data are valid, consequently the acknowledge signal $L_{\text{ack},0}$ changes its state, resetting both channels. Once, reset has occurred, $L_{\text{ack},0}$ changes again its state, sending new data to both channels. This process is repeated in the next stages to transmit data. Further, choosing the pipeline configuration depends on the desired application. In this study, we present the implementation of the Weak Condition Half Buffer pipeline (WCHB pipeline) [5]. Owing to the fact that the WCHB pipeline is a gate-based configuration with dual-rail data encoding, it can be easily implemented with the current SR logic gates to design a delay insensitive pipeline. Figure 5 shows a basic configuration of the WCHB pipeline. In this configuration, NAND gates validate input signals. When data are valid, $L_{\text{ack},0}$ is equal to zero, resetting the signals. Data capture will be done only when $L_{\text{ack,n+1}}$ is equal to one (the next stage has accomplished data transmission successfully). Here, the basic function of the C-element is maintaining the current state until a reset is done; the basic performance of the C-element is as follows: only when both inputs are set to zero or one the output will be zero and one respectively, otherwise, the output will be held with its previous state. A possible implementation of the C-element has already been proposed in [3], where the same internal configuration of the SR logic gates is used. The next section will show the simulation results of these examples, designed with the SR logic gates.

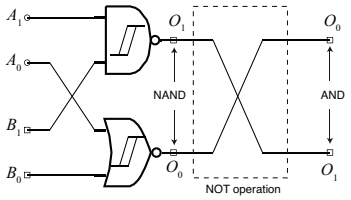

Figure 3: Dual-rail SR AND/NAND gate

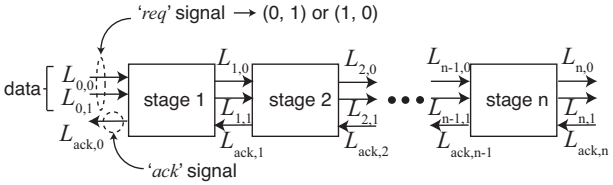

Figure 4: Asynchronous pipeline

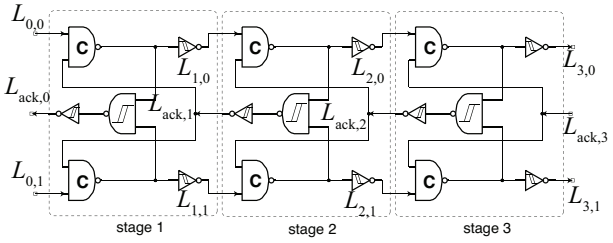

Figure 5: Three-stage WCHB asynchronous pipeline

3. Simulations and experimental results

In this section, simulation and experimental results are presented. Simulations were performed in SPICE for a 0.18-μm CMOS technology (subsection 3.1). Experimental results for the WCHB pipeline were obtained with a macro system and they are presented in subsection 3.2. Subsection 3.3 contains an analysis of the WCHB pipeline.

3.1. Dual-rail NAND/AND gate

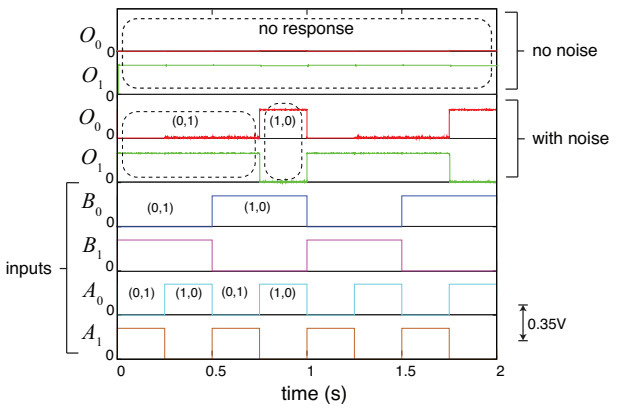

Figure 6: SPICE simulations for the dual-rail SR AND/NAND gate (case with noise)

Here we show electrical simulations of the dual-rail NAND/AND gate, where the power supply was set to V_{dd} = 0.35 V and the standard deviation of noise to $\sigma_{V_{noise}}$ = 0.27 mV. Figure 6 shows the simulation results, where two cases are present: without noise and with noise. In the first case, owing to the presence of mismatches, the logic functions are not generated. In the second case, when noise is applied, the logic functions are recovered, demonstrating the beneficial effect of noise.

3.2. WCHB pipeline based on SR logic gates

In this subsection we present the main application of this study, that is the implementation of a three-stage WCHB pipeline. The simulation parameters are the same as those used in the previous subsection. Figure 7 shows the simulation results for the three stages.

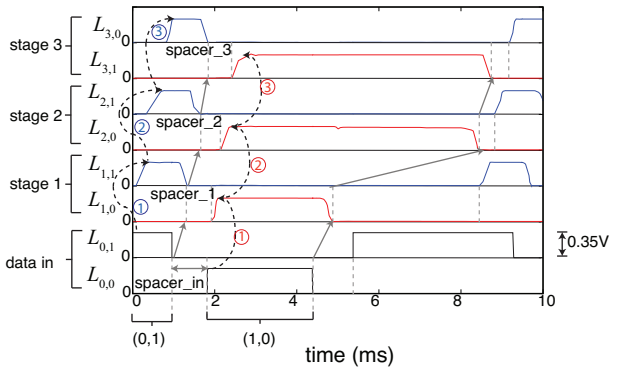

Figure 7: SPICE simulations for the WCHB asynchronous pipeline based on SR logic gates (case with noise)

The signals at the bottom of the figure represent input data, in which every time data have been passed to the next stage (*ack* signal = 1), is generated new data, alternating between {0, 1} and {1, 0} (0 and 1) with the reset (spacer)

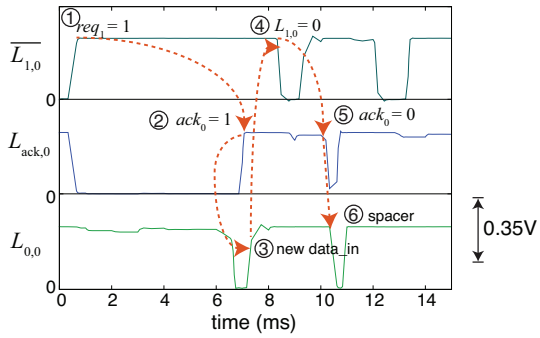

Figure 8: Control signals between stages 1 and 2 of the WCHB asynchronous pipeline

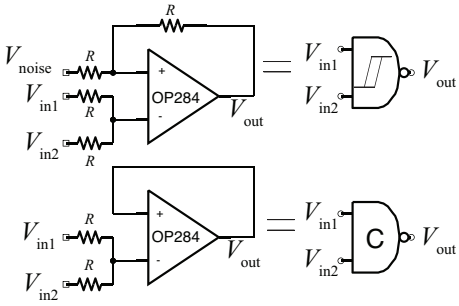

Figure 9: OP284 configuration to implement the SR NAND gate and the C-element

between valid data. The red lines represent data of channel 0 ($L_{n,0}$), and the blue ones represent data of channel 1 ($L_{n,1}$). Note that data transmission is accomplished regardless different values of delays among stages. Figure 8 shows in greater detail the control signals *ack* and *req* that help accomplish data transmission between stages 1 and 2.

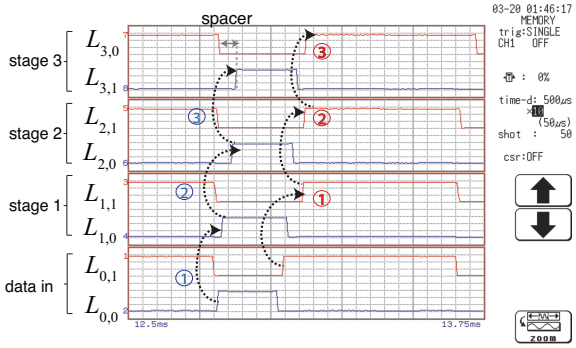

Figure 10: Experimental results for the three-stage WCHB asynchronous pipeline (data transmission), obtained from a Memory Hicorder HIOKI 8826

In addition to the simulations, a three-stage WCHB pipeline was built using an operational amplifier OP284 connected with a positive feedback to generate a hysteresis characteristic. SR logic gates were built using the configuration shown in Fig. 9. Their performance and parameters

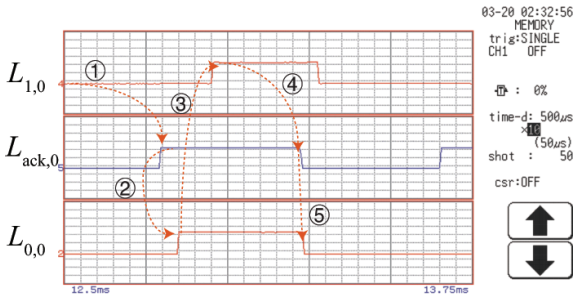

Figure 11: Experimental results for the three-stage WCHB asynchronous pipeline (acknowledge signal between two stages)

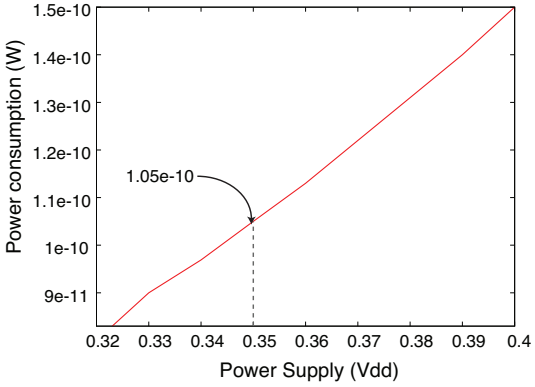

Figure 12: Power consumption of the three stage WCHB asynchronous pipeline

can be reviewed in more detail in [2]. Note that the circuit used for experimental results does not correspond to the practical implementation of the circuit for SR logic gates of Fig. 1, but it is used to demonstrate how the noise can be utilized in nonlinear circuits to recover logic functions in the presence of mismatches. Therefore, parameters such as power consumption and delay time vary from the simulation results. Figure 10 shows the experimental results of data transmission between the three stages, and it is seen that data are transmitted independently of the presence of delays. Figure 11 shows the control signals between stages 1 and 2.

3.3. Power consumption and error rate of the three-stage WCHB asynchronous pipeline

In this subsection, the general performance of the three-stage WCHB pipeline is described in greater detail in terms of the power consumption and error rate. Figure 12 shows the simulation results of the power consumption (P) versus standard deviation of noise ($\sigma_{V_{\text{noise}}}$), where a linear relationship between the parameters is observed. The power consumption is 105 pA for our current setting (V_{dd} = 0.35 V), demonstrating that the pipeline is consuming low power. Additional simulations were performed for the error rate

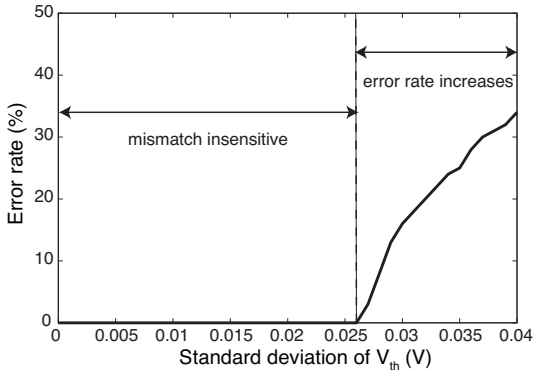

Figure 13: Error rate versus $\sigma_{V_{\text{th}}}$ (V)

(%) versus ($\sigma_{V_{\text{noise}}}$), where a zero tolerance is demonstrated for a $\sigma_{V_{\text{noise}}}$ value less than 0.25 V(Fig. 13).

4. Conclusions

SR logic gates offer three main advantages: a mismatch insensitive design over a certain range of standard deviation of the threshold voltage, a low-power design, and a stable response regardless of the noise input. However, the dependence on noise also generates unpredictable delays in their response. Although these delays represent a challenge to SR logic gate synchronization, an alternative solution is found by employing asynchronous circuit design, where the possibility of designing DI circuits allows a reliable synchronization of the SR logic gates. Although the current configuration is expensive in terms of VLSI area, it provides a solution for future devices beyond MOSFET technology which exhibits large internal fluctuations leading to system-level malfunctions.

References

[1] Asai T., "Logical operations based on stochastic resonance for coarse-grained devices," *The 1st RIEC International Symposium on Brain Functions and Brain Computer*, 2012.

[2] Gonzalez-Carabrin L., Asai T., and Motomura M., "Asynchronous digital circuit design using noise-driven stochastic gates," *International Symposium on Nonlinear Theory and its Applications*, 2013.

[3] C. Mayer, "Asynchronous Circuit Design," John Wiley & Sons, 2001.

[4] S. Nowick, and M. Singh "High-Performance Asynchronous Pipelines: An Overview," *J. IEEE Design AND Test of Computers* , vol. 28, pp. 8-22, 2011.

[5] P. A. Beerel, R.O. Ozdag, and M. Ferretti "A Designer's Guide to Asynchronous VLSI," Cambridge University Press, 2010.