Self-Locked Asynchronous Controller for RISC-V Architecture on FPGA

Florian Deeg, Sebastian M. Sattler
Chair of Reliable Circuits and Systems
Friedrich-Alexander-University Erlangen-Nuremberg
Paul-Gordan-Str. 5, 91052 Erlangen, Germany
Email: {florian.deeg,sebastian.sattler}@fau.de

***Abstract*—We present a new approach for designing an asynchronous control unit for a RISC-V processor using dual-rail domino logic and a self-locking mechanism. The proposed method is based on the observation that dual-rail domino logic can be mapped to look-up tables in FPGAs. This allows for the design of a self-locking asynchronous control unit that is both inherently structurally safe and efficient. First we discuss the concept of dual-rail domino logic and its advantages for asynchronous circuits. A self-locking mechanism is presented that can be used to prevent asynchronous circuits from entering erroneous states. The mechanism is based on the use of a pulse circuit that locks the input, triggers a precharge and then an evaluate phase until it acknowledges the outputs and unlocks the input. This ensures that the circuit is in a stable state before it starts the computation. Afterwards, we apply the proposed approach to the design of an asynchronous control unit for a RISC-V processor. The control unit is implemented using look-up tables and function stable circuits. The result is a control unit that is both safe and efficient.**

I. Introduction

Digital circuitry distinguishes between synchronous and asynchronous circuits. Synchronous circuits use a common clock signal to control circuit functions, while asynchronous circuits use alternative synchronization methods instead of a global clock. This alternative synchronization method results in increased resistance to noise and other disturbances compared to other circuit types. Synchronous circuits are commonly used in microprocessor technology because they are easier to implement and debug than asynchronous circuits. However, in certain cases, such as when performance or power consumption is a concern, asynchronous circuits may offer advantages. RISC-V is a novel Instruction Set Architecture (ISA), proposed and designed by Berkeley, that is rapidly gaining prominence on the scene. The key concept behind this ISA is to enable the implementation of a Reduced Instruction Set Computer (RISC) processor with a load-store architecture without the need to pay royalties for its use. [6] In addition, the ISA was designed with modularity in mind. It is therefore possible to enable more features of the processor architecture by including different ISA extensions. Modularity and royalty-free have proven to be key concepts for hardware developers, who are now more inclined to provide solutions specifically tailored to a niche problem. RISC-V is still in its infancy, and clearly behind the current market dominance of the x86 and ARM ISA for the high-end embedded market in System-on-a-chip (SoC) [1] with automotive applications, for example. However, we are likely to see a growth of RISC-V ISA.

In this paper, we show how an asynchronous controller is designed which will control a RISC-V multicycle processor. To achieve high switching speed with low power consumption, dual-rail domino logic (DRDL) design will be used [4]. DRDL stages consist of two output paths, called rails, with each a Pin for F and $\overline{F}$, which are organized to complement each other. Dual-rail domino circuits are highly immune to noise and other interference, making them suitable for various applications such as microprocessor and field-programmable gate array (FPGA) technology. This paper investigates the capabilities of domino logic circuits in an FPGA and presents a technique for their implementation in the overstated platform.

In order to guarantee the feasibility and safety of the design, a low-level implementation is required. Our asynchronous design requires the specification of design rules that deviate from the standard approaches adopted clocked design. This automaton is self-clocking and fault-tolerant due to the dual-rail approach. A comparison with the clocked version highlights the advantages of the self-clocking version. The complexity of the automaton is manageable, using the necessary z-variables to define all the states specified by the ISA. In addition, the design concept is transferable to more complex applications. The pipeline has been successfully tested and is confirmed to be free of glitches, hazards and races.

II. Structure Observations of Domino Logic Circuits

Domino logic circuits are created using domino gates that have precharge PMOS for charging, n-complexes for function realization, and NMOS transistors for evaluation. To prevent the cascading follower stage from opening prematurely during precharge, additional inverters are placed at the output since function F generates a 0. A clock pulse controls the domino inverter, which outputs a signal that can be 0 or 1 and is then passed on to the next domino gate. In this section, we will briefly review the individual structures on transistor level (TL), starting with single-rail domino logic. We will then demonstrate how to implement dual-rail domino logic circuits in a single look-up table (LUT).

A. Single-Rail Domino Logic in FPGA

First, we will examine a single-rail domino logic (SRDL) circuit with a keeper on TL, as displayed in Figure 1. This is now

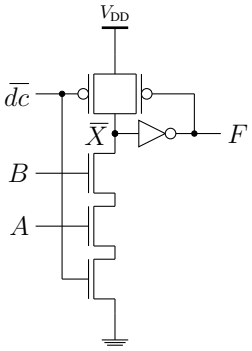


Figure 1: Single Rail Domino Logic

mapped onto a multiplexer (MUX) structure of pass transistors, which realizes a LUT, as shown in Figure 2. Because the lower

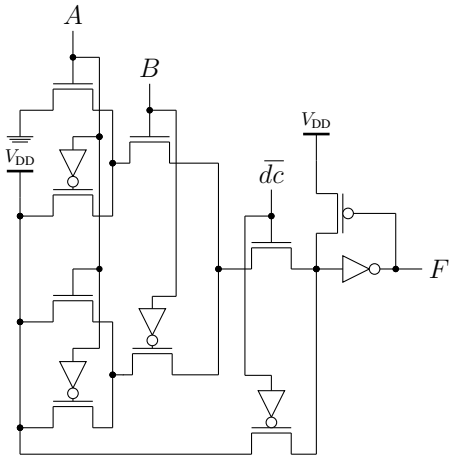


Figure 2: Single-Rail Domino Logic mapped to LUT3

path for $\overline{dc} = 0$ is to charge the inverter, all assignments are mapped to 1. The structure was not drawn to include this path for simplicity, but only the connection to V_{DD} is included. The node preceding the NMOS, which is controlled by $\overline{dc}$, can only be charged to V_{DD} or go high-Z and hold the charge. Therefore, this simplification accurately reflects the structure. Moreover, the node $\overline{X}$ can solely be pulled over AB to GND, meaning that the top path is the only one capable of triggering the transition from 1 to 0. Hence, the bottom path loads from 0 to 1 (Precharge), and the upper path discharges from 1 to 0 (Evaluate). By implementing this simplification and demonstrating solely the paths for the transitions, we achieve the structure depicted in Figure 2 with the exception that the transistor for Evaluate is closer to the output, as shown in Figure 3. This structure can be replicated by exchanging the control inputs of the LUT.

B. Dual-Rail-Domino Logic

If two complementary SRDL circuits are used, they can be merged to create a dual-rail domino logic circuit. An illustration

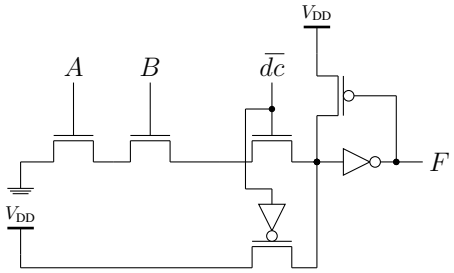
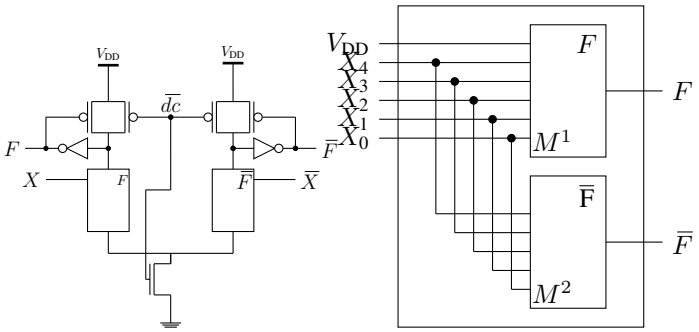


Figure 3: Transitions of SRDL mapped to LUT3

of a DRDL circuit at gate level as applied in the FPGA can be viewed in Figure 4a. Here, we have constructed two SRDL



(a) Dual Rail Domino Logic TL (b) DRDL as LUT6_2

Figure 4: Dual Rail Domino Logic

circuits with their corresponding functions F and $\overline{F}$. These circuits are built in parallel and governed by the same $\overline{dc}$. It is possible to implement the DRDL circuit in a XILINX ARTIX-7 LUT6_2 with input X_5 connected to V_{DD}, which contains two outputs $F = O_6$ and $\overline{F} = O_5$, see Figure 4b [7]. A pulse circuit is now connected to the input of the domino logic module to implement a self-locking circuit (self-X), see Figure 5. The

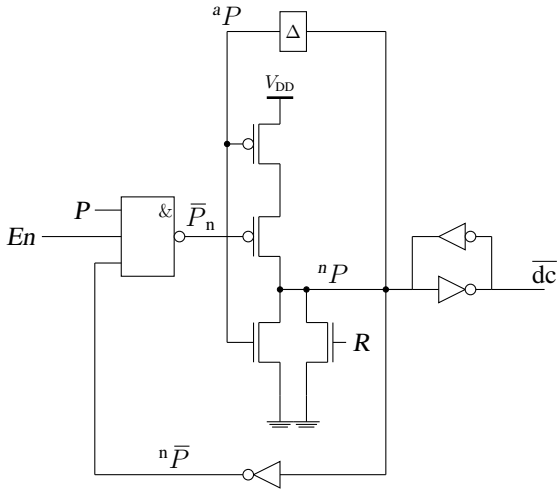


Figure 5: Pulse Circuit

self-X circuit self-locks when a pulse passes through, directly

locking the input. It is only released again by the En pin once the pulse line has passed once. If the pulse lacks sufficient energy, no switching operation is triggered. To solve this issue in FPGAs, a function stable circuit is utilized [2]. This circuit freezes at 1 in the internal state and then sets the output of the pulse circuit to HIGH (Evaluate) once a precharge delay of Δ has passed. This remains the case until the pipeline is completed, after which the input is unlocked again by the en signal. Refer to Figure 6 for the implementation in the FPGA on Gate Level (GL). The circuit self-locks once function stability

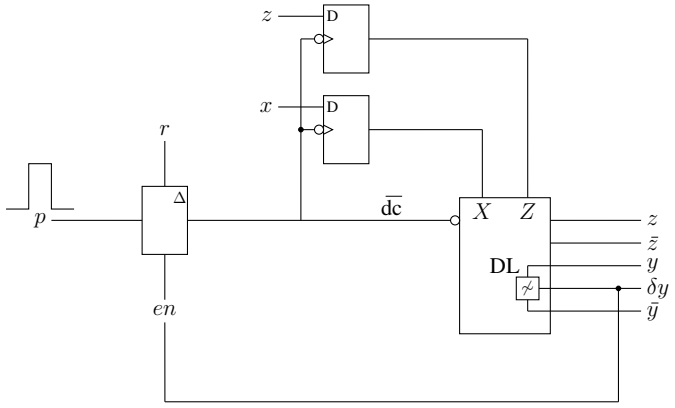


Figure 6: Self-Locked Dual-Rail Domino Logic on FPGA

is ensured, which is the case when $\tau_{\text{inv}} < \tau_{\text{LUT}}$ of the LUT [2]. This stable self-locking design principle can be now applied to pipelines in general. Function stability is ensured by limiting the input combinations to the LUT for stable feedback. To accomplish this, a Design Rule Check (DRC) was developed. The DRC ensures function stability by comparing the verified LUT assignments for feedback with that used in the current design.

III. Control Unit for RISC-V Architecture

The simplified instruction set at the core of RISC-V is small and orthogonal, allowing for a thriving ecosystem of innovation. This simplified approach reduces the hardware requirements and improves overall performance by eliminating the complexity and overhead associated with complex instruction sets. The paper presents the design of a control unit for a 32-bit RISC-V architecture.

A. Control Unit Design

The implementation of an asynchronous multicycle control unit in the FPGA is demonstrated below. A multicycle unit was chosen to fully utilize the benefits of asynchronicity by dividing cycles and handling particularly long cycles separately. The multicycle controller from [3] serves as our reference point for the design. The GL representation of the automaton can be seen in Figure 7. The state transfer function takes op-code variables 6:3 as inputs. The output function takes funct3, funct7_5, and the zero flag as additional inputs. The corresponding state diagram with the states of the multicycle processor control unit are shown in Figure 8. [3]

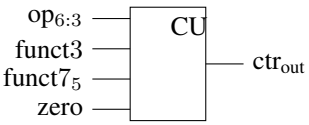


Figure 7: Block Diagram of the Control Unit

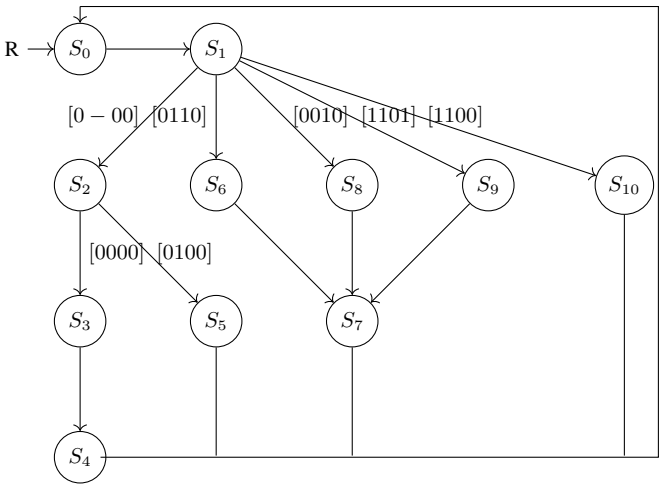


Figure 8: Automaton Graph

B. Control Unit

The control unit has 11 states, each of which generates different outputs for different instructions. To understand the different states and their function in the central processing unit (CPU), please refer to [3]. It is designed as a Moore machine, which needs more states than mealy machines in general. The machine is coded as a multi-cycle machine, which means that there are different cycles (different number of states to get back to S_0) for different instructions, e.g. the load word instruction goes through 5 states (needs 5 cycles), while the branch-equal (beq) instruction needs only 3 cycles. This results in different path delays and especially for asynchronous architectures in better performance. We have used only bits 6 to 3 of the opcode ($op_{6:3}$) as input, because they are all disjoint to each other, and therefore each event can be controlled individually by four input variables. The automaton is now designed as a dual-rail domino logic pipeline circuit.

C. Pipelined Automaton

The automaton with constant instructions through all states can be designed with a single self-X pipeline. The automaton is coded one-hot (except for state [00000]) to make the pipeline as easy to design as possible. Since the automaton has a maximum of five runs, a pipeline with five states and five z-variables is designed, see Figure 9.
Edge A and B from the opcode are $A = \begin{bmatrix}1\,1\,0\,0\end{bmatrix}$ and $B = \begin{bmatrix}0\,1\,\text{-}\,0\end{bmatrix} \vee \begin{bmatrix}\text{-}\,1\,0\end{bmatrix} \vee \begin{bmatrix}1\,1\,0\,1\end{bmatrix}$.
The domino logic then simply passes the 1 for each state until the final state is reached. The pipeline is designed to be function

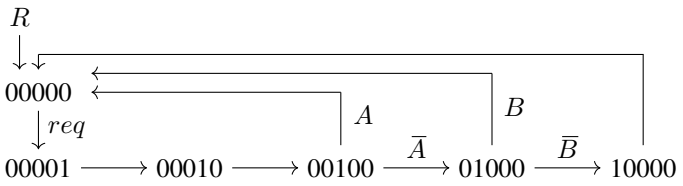


Figure 9: One-Hot encoded Pipeline

stable and self-locking, i.e. each incoming pulse locks the input and sets the next stage only when $en = 1$ and an incoming pulse P arrives. The stabilization is valid for each node and is therefore not drawn. To simplify the pipeline, the automaton was coded as a mealy machine, which means that the output function also depends on the input. This is not a problem in this case because the input is also locked by the self-locking mechanism and will be stable until the automaton has run through the pipeline once. Equation 1 to Equation 11 depict the equations for the implemented pipeline.

$$f = (f_4, f_3, f_2, f_1, f_0)$$

$$= ((F_4, \overline{F}_4), (F_3, \overline{F}_3), (F_2, \overline{F}_2), (F_1, \overline{F}_1), (F_0, \overline{F}_0)) \quad (1)$$

$$F_4 = \overline{X}_2\overline{X}_1 \vee \overline{X}_3X_0 \vee X_3X_1 \vee X_3X_2\overline{X}_0 \quad (2)$$

$$\overline{F}_4 = \overline{X}_3X_1\overline{X}_0 \vee \overline{X}_3X_2\overline{X}_0 \vee X_3X_2\overline{X}_1X_0 \quad (3)$$

$$F_3 = \overline{X}_3 \vee \overline{X}_2 \vee X_1 \vee X_0 \quad (4)$$

$$\overline{F}_3 = X_3X_2\overline{X}_1\overline{X}_0 \quad (5)$$

$$F_2 = F_1 \quad (6)$$

$$\overline{F}_2 = \overline{F}_1 \quad (7)$$

$$F_1 = F_0 \quad (8)$$

$$\overline{F}_1 = \overline{F_0} \quad (9)$$

$$F_0 = REQ \quad (10)$$

$$\overline{F}_0 = \overline{REQ} \quad (11)$$

The resulting pipeline can be seen in Figure 10. Each stage i has outputs $f_i = (F_i, \overline{F}_i)$ which feed the next stage.

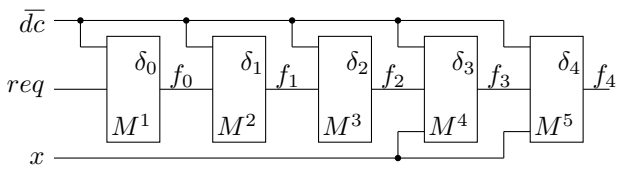


Figure 10: Pipeline in GL

IV. Implementation

The complete control unit consists of the pulse circuit, the DRDL pipeline and the completion detection to generate the en-signal to unlock the input. The completion detection consists of an exclusive-or (XOR) gate for each stage and evaluates after each transition to a new state whether the outputs of the dual-rail LUTs are complementary to each other. When all XOR gates are 1, means all dual-rail stages have disjoint outputs, enable is set to 1 and the input is unlocked. It exhibits the same GL representation as in Figure 6. To realize our structures, we have integrated the control unit in a given synchronized RISC-V CPU and used an ARTY-A7 board with an ARTIX-7 FPGA (XC7A35TICSG324-1L). The control unit was programmed in VHDL at low level. and the device realization in Xilinx Vivado can be seen in Figure 11.

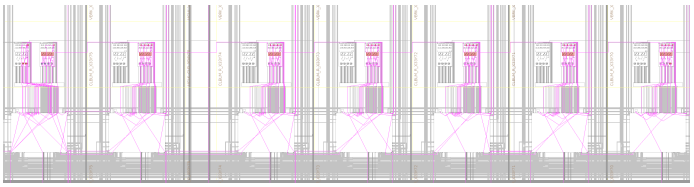


Figure 11: Control Unit after Implementation step in Vivado

A. Implementation Results

Table I depicts the utilization report of the FPGA. The

Unit	LUTs	Registers	Slice
Synchronous	15	16	6
Asynchronous	15	7	5

Table I: Used Ressources in FPGA

structure that was implemented was expected to occupy a larger area due to the dual-rail approach [5]. However, the dual-rail stages were successfully implemented within a single LUT, resulting in no change in the number of LUTs. Furthermore, the self-synchronization technique reduced the number of registers. When dealing with larger projects, it is important to consider the trade-off between area and other factors, such as clock skew issues. Self-clocked pipelines reduce the need for complex clock distribution networks and do not suffer from clock skew. Another benefit of asynchronous circuits is performance and power consumption. To understand the benefits of performance different modules of the CPU can be fed by different clocks and the overall performance can be optimized. Since our synchronous CPU is clocked with one global clock line there is no change in performance as of now. This will be a future project. The dynamic power consumption of the asynchronous automaton was approximately one-third that of the synchronous automaton.

V. Conclusion and Future Work

This paper presents a new approach for designing an asynchronous control unit for a RISC-V processor using DRDL and a self-locking mechanism. The approach offers several advantages over traditional synchronous approaches, including

safety and reliability, as the self-locking mechanism prevents the circuit from hazards or races and the use of dual-rail technology makes it resistant to glitches. Another advantage is constituted by high switching speeds and low power consumption. Additionally, it is scalable and can easily be adapted to different instruction sets and clocks. The feasibility of the proposed approach was demonstrated by implementing the control unit in an FPGA. The pipelined automaton was programmed in VHDL at a low level, and its implementation in Vivado was successful. The results indicate that the pipeline operates correctly and efficiently. The proposed approach has the potential to be used in a wide range of applications.

In future work, we plan to investigate the use of more asynchronous components in the CPU. Request and acknowledge signals will be used to communicate between different components, further improving safety, reliability, and performance of the architecture. We consider dividing the pipeline into smaller steps to increase parallelism and improve performance. To expand the use of asynchronous design, a computer-aided design tool must be accessible to convert functionality into secure structures.

REFERENCES

[1] Enfang Cui, Tianzheng Li, and Qian Wei. Risc-v instruction set architecture extensions: A survey. *IEEE Access*, 11:24696–24711, 2023.

[2] Florian Deeg, Florian Eiermann, and Sebastian M. Sattler. Verification of function stable muller c-element in fpga. In *AmE 2023 âĂŞ Automotive meets Electronics; 14. GMM Symposium*, pages 62–67, 2023.

[3] S. Harris and D. Harris. *Digital Design and Computer Architecture, RISC-V Edition*. Elsevier Science, 2021.

[4] David Hodges, Horace Jackson, and Resve Saleh. Analysis and design of digital integrated circuits : In deep submicron technology / d.a. hodges, h. g. jackson, r.a. saleh. 01 2004.

[5] Hossein Rezaei and Soodeh Aghli Moghaddam. Implementation of low-power and high-performance asynchronous dual-rail join using domino logic gates in 16-nm technology. In *2016 24th Iranian Conference on Electrical Engineering (ICEE)*, pages 142–147, 2016.

[6] Andrew Waterman. Design of the risc-v instruction set architecture. 2016.

[7] Xilinx. *Xilinx Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*, 2 edition, July 2012.