Athens Journal of Technology & Engineering 2024, 11: 1-18
https://doi.org/10.30958/ajte.X-Y-Z

Self-locking Domino Logic Pipelined Controller for RISC-V in FPGA

By Florian Deeg[*]*, Xiangyuan Wu*[±] *& Sebastian M. Sattler*[°]

This paper proposes an asynchronous RISC-V CPU design based on self-locking domino logic. The asynchronous approach offers advantages over traditional synchronous designs, including improved performance, lower power consumption, and greater modularity. The paper details the design and implementation of the asynchronous control unit using domino logic on an FPGA development board. The control unit is designed for a Turing-complete 32-bit RISC-V architecture. A significant aspect of the design is the self-locking mechanism, which ensures that the circuit only unlocks after all processing stages have been completed. This eliminates the need for a global clock and simplifies hazard-free operation. Furthermore, the paper discusses the potential for parallelizing the ALU using domino logic to improve performance further. The implementation of the asynchronous CPU has been analyzed in terms of power, performance, and area using the Vivado Design Suite. The power analysis indicates that the asynchronous processor consumes considerably less power in the clock network compared to its synchronous counterpart, thereby underscoring its energy efficiency. A performance analysis using the SPECint2000 benchmark suite demonstrates a 10% increase in performance, while only using slightly more area. These findings illustrate the asynchronous processor's potential for performance-critical applications while maintaining energy and area efficiency.

Keywords: *domino logic, asynchronous design, self-locking, RISC-V, GALS*

Introduction

Synchronous circuits represent the state of the art in circuit design. Still, asynchronous circuits are becoming increasingly important as they offer numerous advantages over synchronous circuits (performance, power consumption, modularity, no single-point-of-failure, no clock skew, etc.) (Sparsø 2001). Asynchronous circuits are also more resilient to fluctuations in the supply voltage and temperature. Local faults in asynchronous designs are often limited to the affected area, which increases fault tolerance. Furthermore, they generate less electromagnetic interference and are therefore more suitable for applications in which electromagnetic compatibility (EMC) plays an important role (Bouesse et al. 2007). However, the advantages of this approach are also offset by disadvantages, including the necessity for more complex design methods and an associated lack of design tools.

Field Programmable Gate Arrays (FPGAs) are a special hardware component distinguished by their high performance, flexibility, and energy efficiency. In

[*]PhD Student, Friedrich-Alexander-University Erlangen, Germany.
[±]Student, Friedrich-Alexander-University Erlangen, Germany
[°]Professor, Friedrich-Alexander-University Erlangen, Germany.

contrast to conventional integrated circuits (ICs), which are pre-programmed for a specific function, FPGAs can be reconfigured after manufacture to undertake new tasks or optimize performance. This feature renders them an optimal platform for the development of demanding applications that require high computing power, low latency, and customizability.

For a considerable period, the market for processors was divided between two architectures: x86 and ARM, which are mainly used in mobile devices. In recent years, however, a new player has joined them and is providing a breath of fresh air in the form of RISC-V. RISC-V is a license-free instruction set architecture (ISA) that originated at the University of California, Berkeley (Waterman 2016). In contrast to x86, which has grown historically and is complex, RISC-V was developed from scratch. The principle of simplicity was prioritized. This simplicity is intended to minimize hardware costs on the one hand and increase flexibility on the other. RISC-V is becoming increasingly important in the processor world. One decisive factor is that it is license-free. This enables various companies and research groups to develop and utilize processors based on RISC-V. This has led to a wide variety of RISC-V processors that are used in different areas. The spectrum ranges from energy-efficient devices in the Internet of Things (IoT) to high-performance computers. Although RISC-V has not yet reached the market share of x86 and ARM, its growth potential should not be underestimated. The simplicity, flexibility, and license-free nature of RISC-V make it an attractive option for many developers. Other positive aspects of RISC-V include its energy efficiency, scalability, and security, as the basic architecture of RISC-V is so simple and offers little scope for attack.

Structure of the Paper

A brief literature review is conducted to distinguish this paper from others in the field. The following section presents the circuit structure, which comprises the self-locking pulse circuit, the dual-rail domino logic circuit, and the entire pipeline with completion detection and its realization in the FPGA. Subsequently, an existing synchronous multicycle RISC-V processor is briefly introduced, after which a control automaton for this Turing-complete processor is realized as a domino logic pipeline. It demonstrates how the pipeline can be utilized to control a globally asynchronous locally synchronous (GALS) system that can be arbitrarily divided into subcircuits to achieve the highest possible speed and safety. The subsequent chapter deals with the results and a comparison with synchronous automata. Finally, a conclusion and future work are presented.

Related Work

Dooply and Yun (1999) presented a method for optimizing clocking in self-resetting domino pipelines. This method employs soft synchronizers and roadblocks to allow time borrowing, thereby maximizing throughput and eliminating latch

overhead. The authors introduced a high-performance clocking methodology for self-resetting domino pipelines that optimizes the clock rate through time borrowing and robust handling of clock skew while eliminating latch overhead. However, their approach does not adequately simplify the complex clocking and synchronization management or provide a robust precharge management system, nor does it adequately simplify the complex clocking and synchronization management or provide a streamlined implementation and testing methodology. Jung et al. (2002) presented a high-speed add-compare-select (ACS) unit for Viterbi decoders using locally self-resetting CMOS (SRCMOS), which achieves significantly higher data rates compared to static and domino CMOS designs. This approach is associated with higher power consumption and increased design complexity due to the need for careful device sizing and additional components. In contrast, Jung et al. (2003) introduced a dual keeper structure and delay logic gates to enhance the performance and noise margin of domino logic gates, ensuring high-speed switching and robustness to noise and timing variations. However, their approach introduces additional design complexity and lacks a focus on scalability issues

In (Litvin and Mourad 2005) they presented the development of dual-rail self-resetting logic gates with input disable (DRSRLID) for fast and power-efficient arithmetic operations. They also demonstrated the application of these gates in a 16-bit parallel adder. However, their work primarily focuses on arithmetic circuits and does not extensively validate the logic in broader applications or address implementation complexity.

Alsharqawi and Einioui (2006) proposed two novel synchronization approaches for clockless pipelining of coarse-grain datapaths using self-resetting stage logic (SRSL) to achieve high throughput. The approach suffers from scalability issues and increased implementation complexity. Ramadass et al. (2014) introduced the Self Resetting Logic with Gate Diffusion Input (SRLGDI) technique to create low-power, high-speed logic circuits and demonstrated its effectiveness through the design and simulation of various adders. However, their approach increases transistor count and design complexity. The method of designing high throughput and ultra-low power asynchronous domino logic pipelines based on a constructed critical data path was introduced in (Xia et al. 2015). However, their approach does not fully address the challenges of design automation, placement, routing optimization, and timing verification. The implementation of low-power and high-performance asynchronous dual-rail interconnect using domino logic gates in 16-nm technology was proposed in (Rezaei and Moghaddam 2016). The integration of self-locking mechanisms or the detailed implementation of a complete RISC-V pipeline controller remains an issue. Sokolov et al. (2020) introduced a novel framework for automating the design of asynchronous logic control in AMS electronics, integrating formal verification and specialized analog-to-asynchronous interface components for handling non-persistent signals. It does not fully address the challenges of comprehensive design automation and efficient handling of nonpersistent signals within the FPGA implementation. Li et al. (2021) presents a methodology for implementing asynchronous phase-decoupled circuits using traditional electronic design automation (EDA) tools. The authors present an

asynchronous RISC-V processor implemented on the Xilinx ZCU102 FPGA, achieving a threefold improvement in dynamic power efficiency compared to its synchronous counterpart, while maintaining similar resource utilization. The approach demonstrates the potential of asynchronous design in reducing power consumption for IoT and neuromorphic applications, despite challenges in commercial tool support.

This work builds on the work in (Deeg and Sattler 2024), which focused on structural feasibility in the FPGA. In this paper, the design of the automaton and in particular the low-level design is described in more detail, with results of the asynchronous implementation.

Self-locking Domino Logic

This section presents the structure and realization of the self-locking domino logic in the FPGA. The delay-insensitive domino logic was selected to minimize constraints in the design process while maintaining hazard-free and race-free operations. This approach contrasts with one-step designs (Deeg et al. 2020), where complex algorithms are employed to construct the automaton without clocking. The programming in the FPGA occurs at the lowest level of abstraction to ensure that the structure is built in the same way, without the software attempting to optimize the structure. This is because the synchronous optimization process is used to build the structure. The asynchronous design cannot be simulated, so it must be built in accordance with the structure and verified with tests. This is to ensure that any known error models are excluded. The structural comparison of domino logic on the FPGA was conducted in (Deeg and Sattler 2024). This section will subsequently discuss the individual realizations in the FPGA at the low level.

Globally Asynchronous Locally Synchronous (GALS)

GALS is a design methodology for electronic circuits. It addresses the challenge of ensuring safe and reliable data transfer between independent clock domains within a system. A GALS system breaks down the circuit into independent blocks, each with its own local clock. These blocks communicate with each other asynchronously using handshaking protocols (Krstic et al. 2007). This allows for flexibility because blocks can operate at different speeds based on their needs, and scalability because the system can be easily expanded without worrying about the global clock. Furthermore, the GALS methodology results in reduced power consumption, as only active blocks are clocked, thereby increasing the system's power efficiency.

Asynchronous Handshake Protocol:

An asynchronous handshake protocol represents a communication agreement between two or more entities that allows them to exchange data without the necessity of a common clock (Chapiro 1984). In contrast to synchronous protocols, which rely on the timing of a common clock to regulate communication,

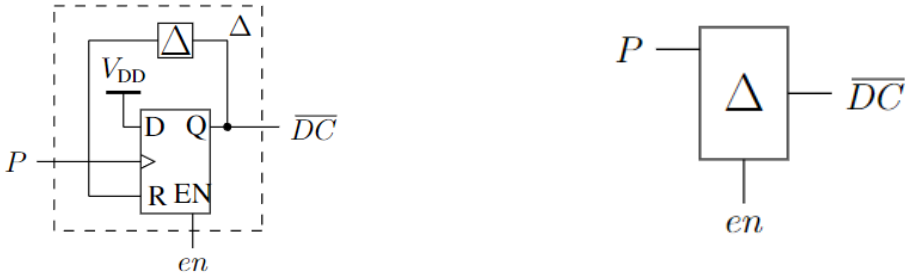
asynchronous handshake protocols employ a pair of signals to regulate data transmission. The initial signal is used to initiate the transmission of data (REQ), while the subsequent signal is utilized to confirm the successful completion of the data transmission (ACK).

Pulse Circuit

The purpose of self-locking is to enable the system to be unlocked again only once the circuit branches have been run through once and brought into a valid state. The input pulse circuit, which locks the input, can be seen in Figure 1.

Figure 1. *Pulse Circuit for Self-locking and Duty Cycle*

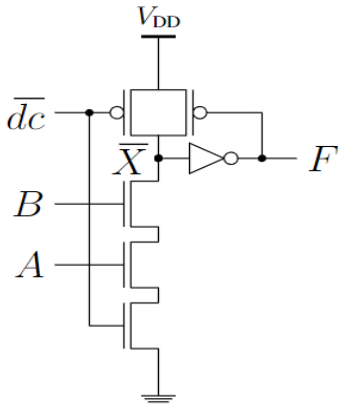
(a) Self-resetting D-FF (b) Symbol



The self-resetting input pulse circuit is employed for self-locking, whereby the input is directly locked following an initial pulse (REQ), and a precharge phase for the domino logic is initiated by the circuit's self- resetting feedback, subsequently disabling the input. The duration of the self-reset determines the length of the precharge phase. It is therefore necessary to ensure that the precharge phase is long enough for all internal nodes to be pulled to V_{DD}. Once this has been achieved, the dual-rail domino logic gates (DRDL) have no disjunctive outputs and then trigger the evaluation phase after the system has self-reset. The rising edge of dc then initiates the transfer of states and input signals to a D-FF at the input, where they are stabilized until the next evaluation phase. The circuit thus blocks the input, generates a duty cycle, and ensures stable signals during the evaluation step. Once the following block is done it will set an enable signal to 1 (ACK) and unlock the input again.

Domino Logic

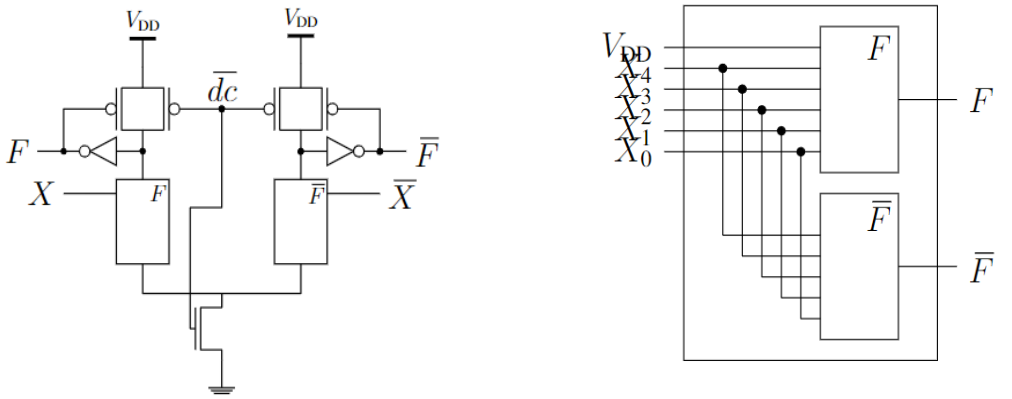
Figure 2. *Single Rail Domino Logic on Transistor Level (TL)*



The domino logic is an asynchronous logic family based on the principle of the domino effect (Hodges et al. 2004). The domino effect describes the chain reaction that occurs when one domino falls and knocks over the next in a row. In domino logic, these effects are used to transmit data through a switching network. Domino logic offers the aforementioned advantages of asynchronous circuits over traditional synchronous logic families. The general mode of operation of a domino logic gate can be divided into two phases: precharge and evaluate. A domino gate represents the fundamental unit of construction in domino logic. It is composed of two transistor circuits, one for the pull-up phase and one for the pull-down phase, which is composed into a single unit, see Figure 2, where an example for an AND2 domino gate is given. In the precharge phase, the inner node is charged to V_{DD}, and the logic state after the inverter is 0. If we then switch to the evaluate phase, i.e. our duty cycle switches from 0 to 1, the node is pulled to GND when the Pull-Down is active (i.e., the equation is fulfilled) and logic 1 is present at the output. Domino logic gates can now be connected in series and propagate through the pipeline. As the goal is to design asynchronously and recognize the transition through the gate, DRDL gates are employed, see Figure 3. These have an output F and the complementary output $\overline{F}$. The same principle applies here: first comes the precharge phase and then the evaluation phase. In the PC phase, both inner nodes are pulled to V_{DD}, the outputs are equivalent in their output value of logical 0, and then in the evaluation phase, one output becomes 1, while the other remains 0 due to the disjointness. This allows for the direct recognition of whether the domino gate has finished switching or not by linking both complementary outputs with an XOR. The dual-rail circuit thus provides a means of determining whether the circuit is in a valid state (i.e., the switched state) or an invalid state (i.e., the switching process is still underway). This information is always available, allowing the user to ascertain whether the circuit is currently occupied or ready for new data.

Figure 3. *Dual Rail Domino Logic*

(a) DRDL on TL (b) DRDL as LUT6 2 on FPGA



Pipeline with Completion Detection

Domino gates can now be composed serially in such a way that a pipeline is created, which is operated sequentially, i.e. not pipelined. This is achieved by assigning a separate state for each transfer of a 1, i.e. each domino effect to the next stage. In principle, however, real pipelining can also be used with the corresponding holding elements between the stages. However, this would not be a viable approach for the processing of instructions and the multicycle processor in question. The serial composition is performed by setting up the dominoes from the first stage f_0 to the last stage f_{n-1} to form $f = f_{n-1}(f_{n-2}(\ldots(f_1(f_0))\ldots)) = f_0 \circ f_1 \circ \ldots f_{n-2} \circ f_{n-1}$. Firstly, all domino gates are subjected to a preliminary charge which is designed to energize the internal nodes to a voltage of V_{DD} and set the outputs to a value of 0. The evaluation phase then pulls each DRDL gate in a path to 0, thus producing a 1 at one output of F and F. The system is then complete as soon as all DRDL gates are complementary to each other, which in turn unlocks the input. The input pulse therefore serves as a request signal and the en signal as an acknowledgment, so this is the asynchronous handshaking protocol. In the pipeline circuit, it is generally sufficient to check only the last stage for disjunctivity, as the last stage can only switch as soon as the previous one has switched. However, we have conducted a comprehensive analysis of all stages for disjunctivity, i.e. we have applied an XOR operation to each stage and rounded the results to ensure that each individual gate has switched and thus enhance safety.

Low-Level Primitives Design

The realization of our circuits is accomplished through the use of the Arty A7 Artix-7 FPGA Development Board, which is provided by Digilent and contains an FPGA manufactured by Xilinx Inc. The FPGA is programmed with the Vivado Design Suite (VDS) at a low-level in order to precisely define how the structures

are generated within the FPGA (what you see is what you get). The primitive libraries from ARTIX-7 (UG953 2012) are employed for this purpose. Two commands have emerged as pivotal: firstly, the ability to incorporate combinatorial loops into the constraints, and secondly, the don't touch commands to prevent the VDS from modifying any settings. The design is currently still completed manually but will be automated in the future. The logical design is based on look-up tables (LUTs). These LUTs are typically multiplexers that switch exactly one path to the output, depending on the input assignment. They are constructed in the shelf from NMOS pass transistors or transmission gates (Chiasson and Betz 2013). These low-level primitives can now be initialized as shown in the code snippet below.

Listing 1. *Low-Level LUT6_2 for AND2 DRDL Gate*

```
1.  LUT6_2_inst : LUT6_2 generic map (
2.  INIT => X"800000007FFF0000") -port map (
3.  O6 => f_int, -- 6/5-LUT output (1-bit)
4.  O5 => fbar_int, -- 5-LUT output (1-bit)
5.  I0 => '1', -- LUT input (1-bit)
6.  I1 => '1', -- LUT input (1-bit)
7.  I2 => x_int(0), -- LUT input (1-bit)
8.  I3 => x_int(1), -- LUT input (1-bit)
9.  I4 => _dc, -- LUT input (1-bit)
10. I5 => '1'-- LUT input (1-bit)
11. );
```

To realize dual-rail domino logic circuits, it has been decided that the LUT6_2 will be employed, as this structure allows for two disjoint outputs when input 5 is clamped to V_{DD}. However, this does entail a tradeoff in that one input is no longer available for use, and the number of table entries is reduced from 26 to 25. For designs with a maximum of five inputs, however, this has no negative effects. The LUT is initialized with a hexadecimal number, in this case, the realized function is $F = I_4 \wedge I_3 \wedge I_2 \wedge I_1 \wedge I_0$ for positive Pin F and $\overline{F} = \overline{I_4} \vee \overline{I_3} \vee \overline{I_2} \vee \overline{I_1} \vee \overline{I_0}$ for the complementary $\overline{F}$. To generate the duty cycle for our self-locking input pulse circuit, a D-FlipFlop is used that is permanently connected with one at the input and briefly goes to one on the positive edge of P and resets itself asynchronously after a duration τ_Δ. The low-level primitive of an FDCE, which is a D-FlipFlop with Clock Enable and Asynchronous Clear, is employed for this purpose. A code snippet for our feedback pulse circuit is provided in reference to the FDCE.

Listing 2. *Low-Level Self-Resetting Pulse Circuit*

```
1. INIT => '0') -- Initial value port map (
2. Q => dc, -- Data output
3. C=> P, -- Clock input
4. CE => '1', -- Clock enable input
5. CLR => dc, -- Asynchronous clear input
6. D=> en -- Data input
7. );
```

Parallelization of Domino Gates

As previously stated, switching can also occur in parallel, rather than in a cascaded manner. This is because the switching processes have a clear direction and a clear end, due to the disjunctivity of the components and the self-clocking enables hazard-free operation. This is because the circuit only unlocks as soon as all the switching parts are disjoint to each other. It is therefore also conceivable, for example, to design the arithmetic logic unit (ALU) in parallel as a dual-rail domino gate to maintain the minimum processing delay by communicating with the controller using the handshake protocol and switching the individual gates in parallel until they are all disjoint. In this instance, the individual domino gates are composed in parallel to form $f = f_0(x) + f_1(x) + ... + f_{n-2}(x) + f_{n-1}(x)$

Implementation for RISC-V Processor

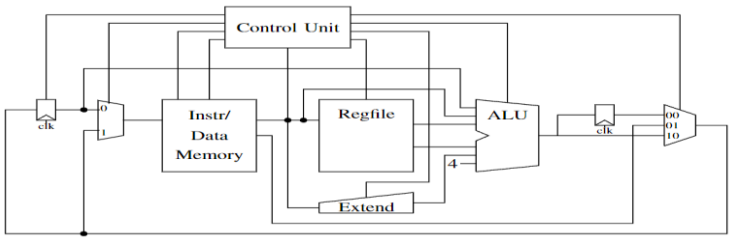
RISC-V represents a flexible and energy-efficient alternative to the dominant Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC) architectures. The simplified instruction set at the core of RISC-V is small and orthogonal, allowing for a thriving ecosystem of innovation. 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 Turing-complete RISC-V architecture.

Synchronous Multicycle Central Processing Unit (CPU)

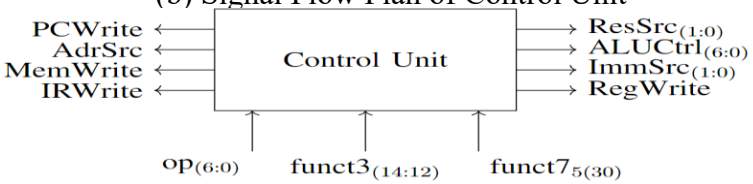
We will now briefly introduce the initial processor (Harris and Harris 2021), see Figure 4. It is a synchronous processor that is Turing complete, which means that it can calculate all Turing-computable functions. The processor is realized as a multicycle processor in order to design the different access times for different instructions in a way that allows for the division of an instruction into different individual steps. This is in contrast to a single-cycle processor, where the worst-case path for the entire instruction is considered. Instead, in this case, the worst case for the individual processing steps is considered. However, the processor employs a Harvard architecture, which is evident from the fact that it has separate data and instruction registers in a block random access memory (BRAM) (i.e., with two different addresses).

Figure 4. *Synchronous CPU and the Synchronous State Transfer Function*

(a) Architecture Synchronous CPU



(b) Signal Flow Plan of Control Unit



Synchronous Control Unit

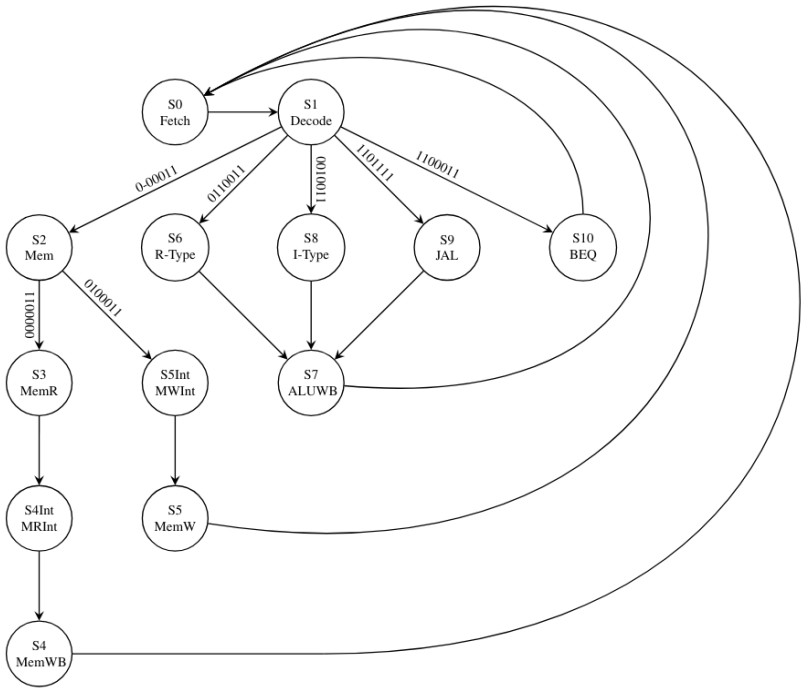
To be Turing-complete, the instructions given with the opcode in Table 1 are implemented. In order to process the instructions, an automaton is generated that was derived from RISC-V and uses the input opcode 6:3 to decode the individual states. In contrast to Harris and Harris (2021), a few states were added because the memory access requires two clock cycles, for example. Consequently, the multicycle processor is divided into the branches load, store, r-type, i-type, b-type, and jal, with the clock cycles split up in order to achieve shorter access times, see Figure 5.

Table 1. *Instructions*

Branch	**OPCode**	**Name**
L	0000011	Load
I	0010011	I-Type
S	0100011	Store
R	0110011	R-Type
B	1100011	B-Type
J	1101111	JAL

This results in the longest instruction load and the shortest branch-if-equal (BEQ). A single-cycle processor would now cover the worst case for all instructions, whereas the multicycle processor saves three cycles for NEQ. It is evident that the differing access times of the instructions justify the necessity of the multicycle processor. The automaton was then constructed as a Moore machine, with each state having a single output. The synchronous automaton was then designed at the high-level with optimizations from VDS.

Figure 5. *Automaton Graph of synchronous Moore Machine*



Design of the Asynchronous Controller

The circuit is then implemented in the synchronous RISC-V processor. The asynchronous handshaking protocol can significantly enhance the processor's performance, as it allows for different access times for different process steps (e.g., writing memory is much slower than addressing the reg file). Since the domino logic is realized as a pipeline, there is also a direct transfer to pipelines, but this application is not addressed further in this paper. In order to facilitate the design of a pipeline cascade, which is a more complex process in domino logic, we will implement a Mealy automaton that can generate different output values for its states depending on the input signal. In contrast to the synchronous Moore automaton, which has predefined processing times for the various instructions, the Mealy automaton is clocked externally by REQ and ACK. This means that the same states for different instructions can have different access times. Consequently, the states are superimposed and the edges are retained. Furthermore, self-locking can be applied to the output function, which negates the need for hazards and other potential issues. This results in a reduction in hardware requirements compared to the Moore Machine. The individual states were then encoded one-hot to enhance clarity in the domino output in the z-variables. The automaton graph for the pipeline can be seen in Figure 6 and its structure in Figure 7. Edge *A* is the opcode of the BEQ branch, since it only needs three states, *B* is given by R-type, I-type and JAL branches, *C* is the edge for the Store instruction, and the Load instruction gets to the last state [100000].

Figure 6. *Automaton Graph Pipeline*

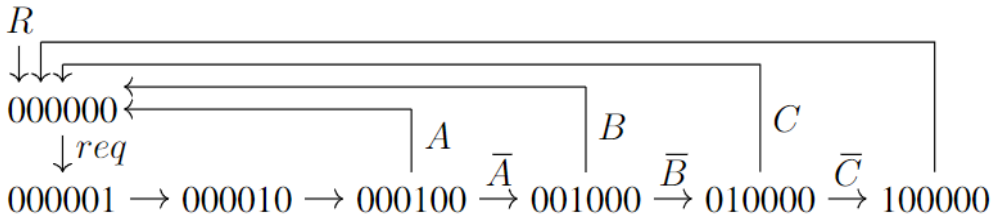
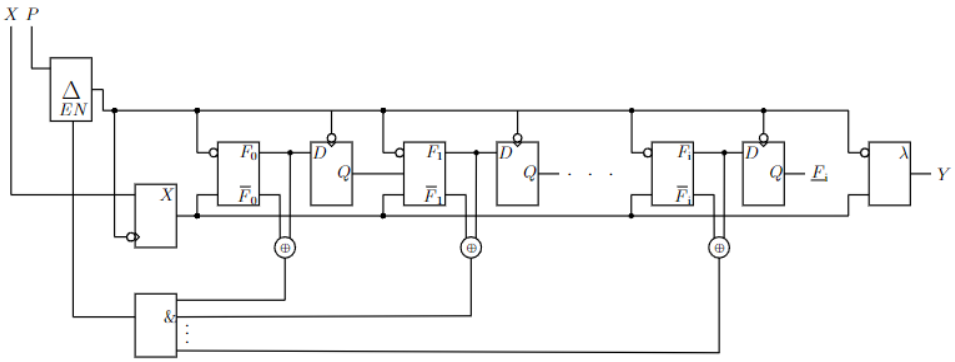


Figure 7. *Structure of realized Pipeline in FPGA*



Design of an Asynchronous ALU

To more effectively illustrate the advantages of the asynchronous handshake protocol, we have elected to configure the ALU as a parallelized, self-locking domino logic. Given the considerable time required to access the ALU, the self-clocked variant represents a promising improvement. In the following section, we will utilize the AND instruction as a case study in domino logic, with the objective of elucidating the design process. The starting point is an asynchronous ALU, which we wish to convert into a domino logic. The bitwise AND can be implemented straightforwardly by utilizing the AND structure of a domino gate and combining each position of the 32-bit word in dual-rail. This can be accomplished entirely in parallel. The independence of all dual-rail gates then indicates whether the ALU has undergone rounding, allowing the input to be unlocked by setting the *en* signal (ACK) to 1. The code snippet for the ALU's AND function as DRDL AND2 is listed below.

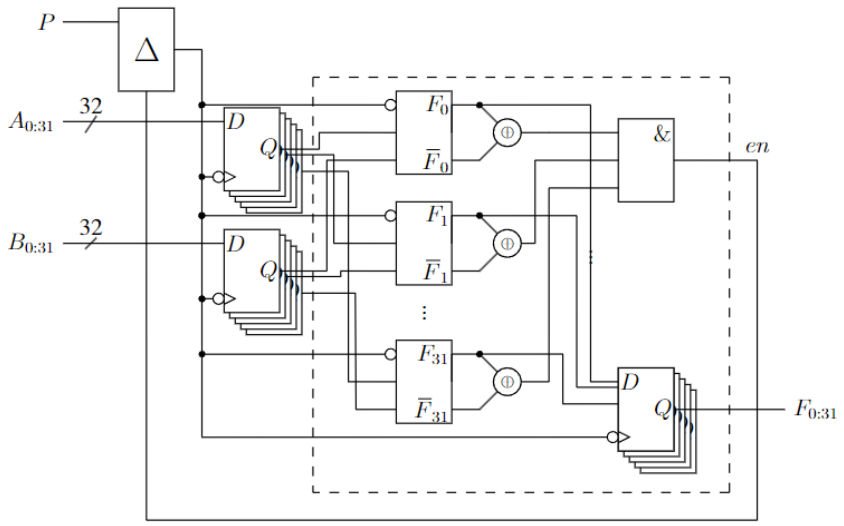
Listing 3. *Low-Level 32-Bit DRDL AND*

```
1.  MY_GEN : for i in 0 to 31 generate
2.  DominoGate: dualRail port
3.  map( dcbar => dc, x(0)=>'1',
4.   x(1)=>'1', x(2)=>reg_b(i),
5.  x(3)=>reg_a(i), f_out=>f_int(i),
6.  fbar_out=>fbar_int(i) );
7.  CompletionDetection:  xor_LUT port
8.  map( A => f_int(i),
9.  B => fbar_int(i),
```

```
10. Y => xor(i)
11. );
12. process(dc)
13. begin
14. if(falling_edge(dc)) then
15. f_out(i)<= f_int(i);
16. fbar_out(i)<= fbar_int(i);
17. end if;
18. end process;
19. end generate;
20. process(xor)
21. begin
22. if(xor=x"FFFFFFFF")
23. then
24. en_int<='1';
25. else
26. en_int<='0';
27. end if;
28. end process;
```

The resulting structure of the AND for ALU in Self-Locking Domino Logic can be seen in Figure 8. Furthermore, the input incorporates a self-locking pulse circuit that generates a duty cycle, thereby initiating the precharge phase. This is followed by a scan of the source registers of the ALU multiplexers, after which the input is unlocked when each of the 32 DRDL gates has disjoint outputs.

Figure 8. *Domino Logic ALU*



Integration in the CPU

The self-locking machine can now be readily incorporated into the existing CPU and controlled via the clock, for instance. While this does not immediately enhance performance if the other processor components do not exhibit GALS

behavior, it demonstrates the simplicity of integrating self-timed circuits. Furthermore, self-timing necessitates fewer FFs, which consequently results in reduced power consumption. To illustrate the advantages of asynchronous handshaking, the DRDL ALU was also integrated. The controller oversees the operation of other components in a synchronous manner, while simultaneously initiating the asynchronous handshaking process with the ALU, thereby accelerating the execution of instructions that utilize the ALU.

Power Performance and Area (PPA) Results

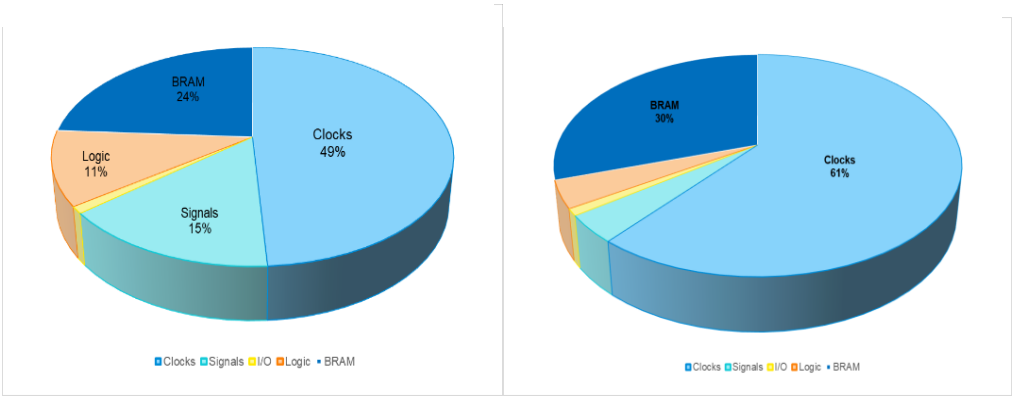
The PPA analysis of the implemented asynchronous processor on an FPGA definitively shows its efficiency and viability for various applications. This section presents the findings from the synthesis, implementation, and simulation processes using the Vivado Design Suite. The results are presented in three subsections: Power Analysis, Performance Analysis, and Area Analysis.

Power Consumption

The power consumption shares of the individual processors were obtained from the Vivado Power Analysis Tool, see Figure 9.

Figure 9. *Power Consumption of the CPUs*

(a) Asynchronous CPU (b) Synchronous CPU



The total dynamic power consumption of both the asynchronous and synchronous processors is nearly similar, which can be attributed to the parallel operation of the asynchronous processor. However, the asynchronous design achieves substantial power savings in the clock network, as evidenced by the lower percentage of power consumption dedicated to clocks in the asynchronous processor compared to the synchronous one. This highlights the efficiency of the asynchronous processor in minimizing clock-related power, which is a critical factor in overall power management and energy efficiency.

Performance Analysis

In this section, we assess the efficacy of our asynchronous CPU design by contrasting it with a synchronous CPU using the SPECint2000 benchmark suite. SPECint2000 is a well-established benchmark that measures the performance of CPUs with integer-heavy workloads, providing a comprehensive assessment through a variety of real-world applications. The benchmark is composed of approximately 25% loads, 10% stores, 11% branches, 2% jumps, and 52% R- or I-type ALU instructions. The diverse range of operations included in the SPECint2000 benchmark makes it an optimal tool for evaluating and comparing the performance characteristics of different CPU architectures. The objective of this analysis is to highlight the advantages and potential trade-offs of the asynchronous CPU design in comparison to its synchronous counterpart. A loop was constructed around the test code with a branch-if-equal (BEQ) instruction using a clock frequency of 100MHz, and the throughput of the asynchronous and synchronous CPUs, as well as the average duration per instruction were determined (see Table 2).

Table 2. *Performance Metrics*

Parameter	**Asynchronous CPU**	**Synchronous CPU**
Throughput (MIPS)	25.64	22.73
Average Latency/Instruction	39.5 ns	44 ns

The analysis shows, that the CPUs Performance for the SPECint2000 benchmark increased by around 10% simply by letting the control unit and the ALU do handshaking using DRDL Gates.

Area Analysis

The area analysis focuses on the utilization of FPGA resources, including LUTs, Slice Registers, F7 Multiplexers (F7 Muxes), and Slices. The asynchronous design utilized 6.67% of the available LUTs, indicating a moderate complexity in logic implementation. The design also employed 4.85% of the available slice registers, and leveraged 4.63% Slices. As anticipated, the area utilized exhibited an increase, yet remained within the anticipated range due to the implementation of DRDL gates within a single LUT.

Table 3. *FPGA Resource Utilization*

Resource Type	**Utilization in (%) Async**	**Utilization in (%) Sync**
LUTs	6.67%	6.32%
Slice Registers	4.85%	4.9%
Slices	9.27%	8.9%

Discussion

The PPA results indicate that the asynchronous processor demonstrates significant potential in terms of performance, which is crucial for performance-critical applications. While there was a narrow change in power consumption, the area analysis shows a balanced utilization of FPGA resources, thereby highlighting the feasibility of implementing such designs within reasonable silicon area constraints.

Conclusion and Future Work

This work proposes an asynchronous RISC-V CPU design based on self-locking domino control. The asynchronous approach offers advantages over traditional synchronous designs in terms of performance, power consumption, and modularity. The paper describes in detail the design and implementation of the asynchronous control unit using domino control on an FPGA development board. The control unit is designed for a Turing-complete 32-bit RISC-V architecture. A significant aspect of the design is the self-locking mechanism, which ensures that the circuit is not released until all processing stages have been completed. This eliminates the need for a global clock and simplifies error-free operation. Furthermore, the paper discusses the possibility of parallelizing the ALU using domino control to further improve performance. Subsequently, the paper illustrates the straightforward integration of the asynchronous control unit into an existing synchronous central processing unit (CPU), thereby demonstrating the potential benefits of self-timed circuits. Ultimately, the PPA analysis of the implemented asynchronous processor on an FPGA substantiates its considerable potential for diverse applications. The power analysis indicates that while the total dynamic power consumption of the asynchronous processor is comparable to that of the synchronous processor, it achieves a significant reduction in power consumption within the clock network. This underscores the asynchronous processor's efficacy in curbing clock-related power consumption, a pivotal aspect of comprehensive power management and energy efficiency. A performance analysis conducted using the SPECint2000 benchmark suite revealed that the asynchronous processor exhibited superior performance compared to the synchronous processor, demonstrating a 10% increase in throughput and a reduction in average latency per instruction. This performance enhancement is achieved through the use of handshaking with DRDL gates in the control unit and ALU. The area analysis indicates that the asynchronous design employs FPGA resources in a moderate manner, exhibiting a slight increase in LUT, slice register, and slice utilization in comparison to the synchronous design. Despite this increase, the resource utilization remains within acceptable limits, thereby substantiating the feasibility of implementing the asynchronous processor within reasonable silicon area constraints. In conclusion, the asynchronous processor exhibits notable advantages in terms of power efficiency, performance, and area utilization, thereby establishing its viability as a potential solution for performance-critical applications.

References

Alsharqawi A Einioui A (2006) Clockless pipelining for coarse grain datapaths. In *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*.

Bouesse F, Ninon N, Sicard G, Renaudin M, Boyer A, Sicard E (2007) *Asynchronous logic vs synchronous logic: Concrete results on electromagnetic emissions and conducted susceptibility*. Available at: https://hal.univ-grenoble-alpes.fr/hal-00222916/.

Chapiro D (1984) *Globally-asynchronous locally-synchronous systems*. *IEEE Design & Test of Computers* 24(5): 430–441.

Chiasson C, Betz V (2013) Should fpgas abandon the pass-gate? In *23rd International Conference on Field programmable Logic and Applications*, 1–8.

Deeg F, Sattler SM (2024) Self-locked asynchronous controller for risc-v architecture on fpga. In *AmEC 2024 - Automotive meets Electronics & Control; 15. GMM-GMA-Symposium*, 1–5.

Deeg F, Zhu J, Sattler SM (2020) Asynchronous design. In *AmE 2020 - Automotive meets Electronics; 11th GMM-Symposium*, 1–5.

Dooply A, Yun K (1999) Optimal clocking and enhanced testability for high-performance self-resetting domino pipelines. In *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*, 200–214.

Harris S, Harris D (2021) *Digital Design and Computer Architecture*. RISC-V Edition. Elsevier Science.

Hodges D, Jackson H, Saleh R (2004) Analysis and design of digital integrated circuits. In *deep submicron technology / d.a. hodges, h. g. jackson, r.a. saleh*.

Jung G, Kong JJ, Sobelman G, Parhi K (2002) High-speed add-compare-select units using locally self-resetting cmos. In 2002 *IEEE International Symposium on Circuits and Systems (ISCAS)* 1: I–I.

Jung S-O, Kim K-W, Kang S-M (2003) Timing constraints for domino logic gates with timing-dependent keepers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22(1): 96–103.

Krstic M, Grass E, GÃijrkaynak FK, Vivet P (2007) Globally asynchronous, locally synchronous circuits: Overview and outlook. *IEEE Design & Test of Computers* 24(5): 430–441.

Li Z, Huang Y, Tian L, Zhu R, Xiao S, Yu Z (2021) A low-power asynchronous risc-v processor with propagated timing constraints method. *IEEE Transactions on Circuits and Systems II: Express Briefs* 68(9): 3153–3157.

Litvin M, Mourad S (2005) Self-reset logic for fast arithmetic applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13(4): 462–475.

Ramadass U, Ponnian J, Dhavachelvan P (2014) New low power adders in self resetting logic with gate diffusion input technique. *Journal of King Saud University - Engineering Sciences* 29(2): 118–134.

Rezaei H, Moghaddam SA (2016) 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*), 142–147.

Sokolov D, Khomenko V, Mokhov A, Dubikhin V, Lloyd D, Yakovlev A (2020) Automating the design of asynchronous logic control for ams electronics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39(5): 952–965.

Sparsø J (2001) *Asynchronous circuit design - a tutorial*. Available at: https://www2.imm.dtu.dk/pubdb/edoc/imm855.pdf.

UG953 (v 2012.2) (2012, July 25). *Vivado Design Suite 7 Series FPGA Libraries Guide*. XILINX.

Waterman A (2016) *Design of the risc-v instruction set architecture*. Available at: https://people.eecs.berkeley.edu/~krste/papers/EECS-2016-1.pdf.

Xia Z, Hariyama M, Kameyama M (2015) Asynchronous domino logic pipeline design based on constructed critical data path. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23(4): 619–630.

Yun K, Dooply A (1999) Optimal evaluation clocking of self-resetting domino pipelines. In *Proceedings of the ASP-DAC '99 Asia and South Pacific Design Automation Conference 1999 (Cat. No.99EX198)*, 121–124.