µSystems Research Group

School of Electrical and Electronic Engineering

Self – Timed Register Bank with Completion Detection

Marios Elia

Contact: m.elia@ncl.ac.uk

Self-Timed Register Bank with Completion Detection

Marios Elia

August 2014

Abstract

As the need for low-power electronics increases steadily, so does the demand of asynchronous systems that can function under low and variable power conditions. In designing asynchronous logic, hand-shaking protocols are required in order to preserve a form of synchronization. The single most common method, for acknowledgment (ACK) generation, is the use of delay blocks but this can sometimes prove both unreliable and incompetent in terms of performance. The following paper proposes the design of a Register Bank with completion detection (CD) for both Read and Write operations. Simulations performed on the synthesised design provided solid results.

1. Introduction

The following design emerged after an industrial placement in the Microsystems Research Group of Newcastle University. The objective was to enable the low-voltage operation of an asynchronous version of the Intel 8051 microcontroller. As most RAM chips fail to properly operate at low voltages, the use of a self-timed Register bank was proposed. The entire design is very simple and consists of a Register Bank, coupled with a module responsible for generating ACK signals. The use of an equality comparator enables the module to generate an acknowledgment for each write operation, once the data is correctly saved by the register. Given that the Register Bank bears more than one register, it was deemed necessary to provide an ACK for each read request (REQ) too. A second comparator was also used, as the means for providing the read ACK using a completion detection method.

2. Register Bank

The Register Bank, designed for this project, is a collection of four 16 – bit registers, namely the Accumulator register (A), the Register B, the Data Pointer register (DPTR) and the Register T. It has three inputs and one output which are described below.

INPUTS
Address: 2-bit address bus selecting any of the four internal registers
Write_req: The register will save the data present at its input (Data_in) at every rising edge of this signal
Data_in: Receives the data to be stored

OUTPUTS
Data_out: It forwards the data stored in the Register pointed by the address (ADR)
adr_out: It holds the address of the currently selected internal register

Its operation is fairly simple and it is as follows: While the write_request signal is low, it will present any data stored in the memory location specified by the address (ADR). At every rising edge of the write_req signal, it will save the data present on its data_in input, into the location specified by the

ADR input. The data_output, constantly forwards the data stored in the register pointed by the address, while the current register's address is displayed on the adr_out pin. The write_req signal is generated by the two gates shown on Figure 1. The read/write signal is set by the processor, while the register_req signal is generated by an address decoder when the processor wants to access the Register Bank.
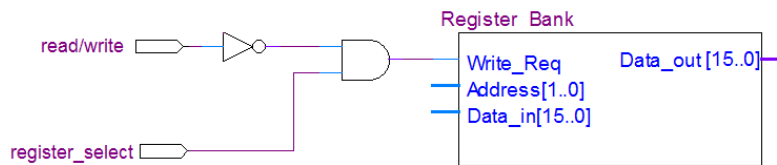

Figure 1: Register Bank

The adr_out pin was specifically added to enable the CD of read operations. The internal architecture of a Register Bank containing only two registers is shown in Figure 2.
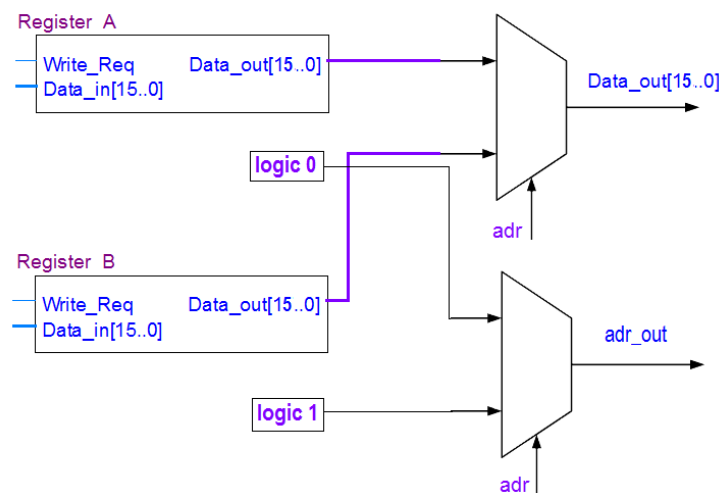

Figure 2: Register bank internal arrangement

For the sake of clarity, the schematic demonstrates the addition of a second multiplexer which drives the adr_out. In reality, the multiplexer responsible for the data output was extended to accommodate two more bits and all partial components that make up the entire 18-bit multiplexer are to be grouped and placed together on the silicon chip. This is to make sure that the variability between the wire delays and the components carrying the address and data is kept to a minimum. The inputs of the multiplexer's MSB are connected either to V_{DD} or ground (logic 1 or logic 0), depending on each register's address. For example, the register's A address in Figure 2 is logic 0, while the register's B is logic 1. Depending on the address coming from the processor, the two MSB will forward the appropriate register address.

3. Read Completion Detection

As explained before, this particular Register Bank contains 4 registers. From the moment that a read REQ is issued until the moment that the required register's data is actually presented on the output, some time period has elapsed. The simplest method for a read ACK generation, would have been to implement a delay block, with a delay equal to the time required by the output multiplexer to switch its output. The REQ signal would then be fed in the delay block and be redirected back as an ACK signal as demonstrated in Figure 3.
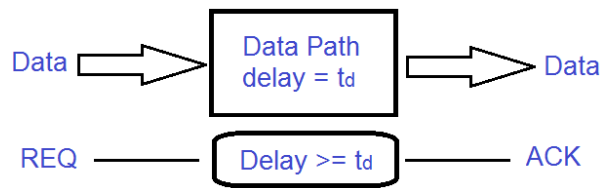
3: Bundled delay solution

The proposed solution, is to implement another two bits as part of the Data_out bus, which demonstrate the address of the current register. Given that the multiplexer elements for the data and ADR bits are implemented and fabricated in exactly the same fashion, it can be considered that both data and address will be available on the output at the same instant. An equality comparator (made up of two X-NOR gates) compares the input and output ADR, raising a logic 1 once both are equal. The comparator's output is then redirected as an ACK signal. The arrangement implemented is shown below.
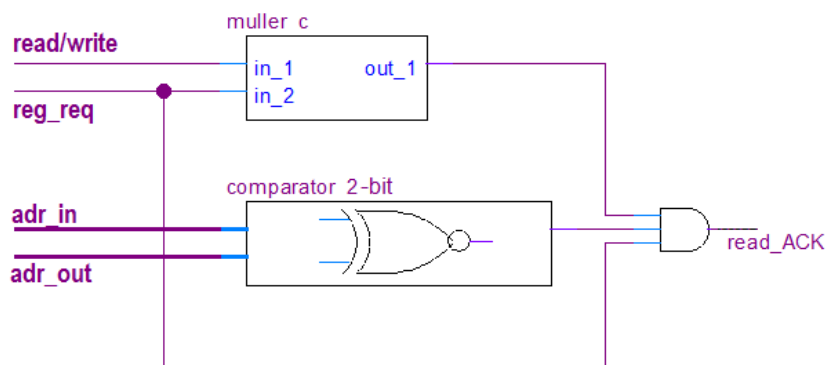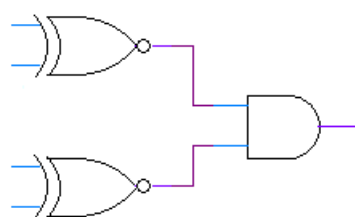

Figure 4: Read ACK arrangement

Figure 5: Comparator's internal implementation

A Muller-C element evaluates and holds the value of read/write signal (coming from the processor) on every REQ rising edge ensuring that a read operation is selected (if the read/write signal is also high). Otherwise, if a Write operation is selected, the value of the read/write signal is evaluated while REQ is low. The comparator's output is allowed to propagate as an ACK signal only if both a read operation is selected and the REQ signal is high. This ensures that an ACK is generated *only* for read operations and it does not interfere with other blocks.

The arrangement used, offers two main benefits over the delay block solution. Firstly, it can be used under a variety of voltage supplies, always generating an ACK once the data is available on the output; the data and address go through identical Mux elements thus their delay is matched. In addition to that, in case that the read request is for the register already selected, no delay will occur (the only delay is the time required to propagate through the AND gate) and an ACK will be generated instantly. For this particular register bank, there is a 25 percent chance of reading from the same register twice, while it can climb to a maximum of 50 percent in case of two registers allowing for faster read cycles.

4. Write Completion Detection

In generating a write ACK signal, the instant when all bits have been saved correctly must be logged. The solution proposed is based on the fact that, this Register Bank's data output is constantly forwarding the data stored in the location pointed by the address. During a write operation, the register's data output changes its output bits. For the time interval between the moment that the Write Request is issued and the instant that all the bits have changed to the appropriate ones, the data available on the output is invalid. The write operation is regarded as completed when the data is valid; that is the moment that the ACK signal should be issued. For detecting the instant when the data becomes valid, it is proposed to use sixteen X-OR gates that compare the Register's input and output data-buses bit-by-bit. While the data on the output differs from the data on the Register's input, the comparator's output is set to logic 0. Once the data is valid the comparator's output will change to logic 1 indicating that an ACK should be produced. The signal coming from the comparator is allowed to propagate to the processor only if the request signal is high as well. This is to ensure that an ACK is generated *only* and for *every* Write request. The proposed solution is displayed in Figure 3.
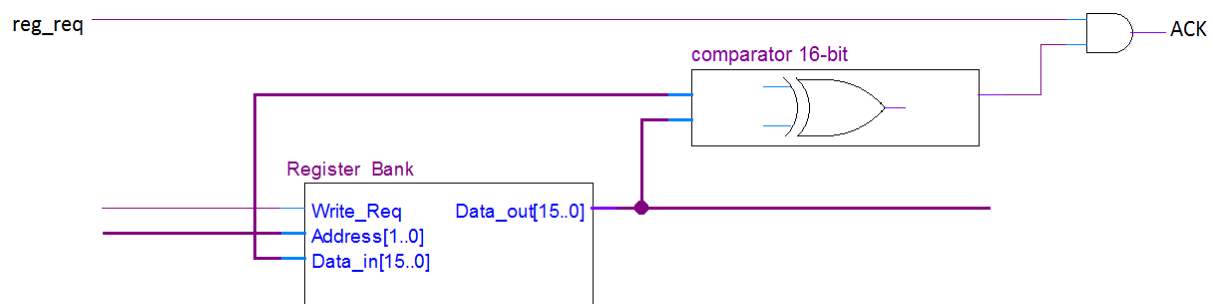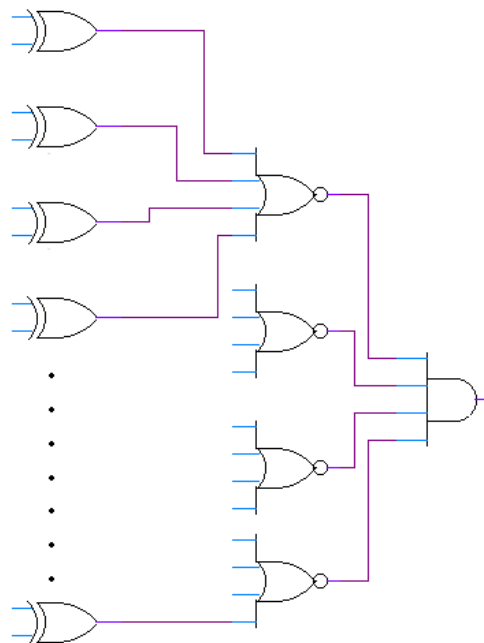

Figure 6: Write ACK generation

The comparator's internal implementation is shown in Figure 5. Each X-OR gate receives as inputs a bit from the data_in bus and the corresponding bit by the data_out bus.

7: 16 - bit comparator's internal implementation

This method demonstrates an exceptional solution for CD and it can be considered far superior than a delay block for a number of reasons. To start with, the 16-bit comparator evaluates the output bit-by-bit making it similar to a speed independent solution and able to operate under a variety of voltage supplies. In addition to that, it also validates that all bits have been saved correctly and error free. On the other hand, the ACK produced by a delay block is not a true indicator of the operation competition, but an approximation of the time it takes for the operation to complete. Even if the operation is not completed, for any reason, the ACK will still be generated (falsely).

5. Timing Requirements

In order to prevent glitches and false ACKs to be generated, some timing requirements must be met regarding the input signalling. Let any of the two comparators to have a previous state of logic 1, and its next state should be logic 0. If the REQ signal arrives before the comparator evaluates its next state, a glitch will occur since the two signals are the inputs to an AND gate. This is demonstrated in Figure 4. It can be clearly seen that if the comparator is not allowed enough time to evaluate its output (on a new read/write cycle) a glitch is very likely to occur.
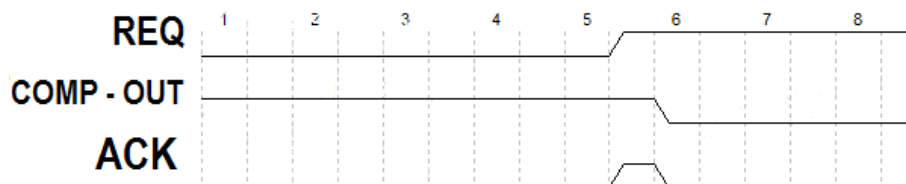

Figure 8: False ACK generation

In avoiding this, it must be ensured that both comparators evaluate their state before the rising edge of the REQ signal. The timing constraints for both the CD methods are considered below.

Read ACK: The address signal should be stable before the REQ signal, for a time period *[t]* which is equal to the propagation delay of the 2-bit comparator. In addition to that, the read/write signal should be stable for a period equal to the propagation delay of the Muller-C element.

Write ACK: The data_input signal should be stable for a time period equal to the propagation delay of the 16-bit comparator to allow for evaluation of its output.

Given that the conditions described above are met, no glitches or false ACKs will occur, as all sub-blocks (comparators, Muller-C) are given enough time to evaluate their next state before the rising edge of the REQ signal.

6. Results

This section contains simulation results performed on the Register's netlist which was produced by the Synopsys Design Compiler logic synthesis tool. All the screenshots captured can be found in the Appendix.

Simulation 1: The simulation performed included a series of both write and read operations, demonstrating the Register's flawless operation. It can be seen that both CD modules functioned exactly as expected generating ACKs for their respective operations.
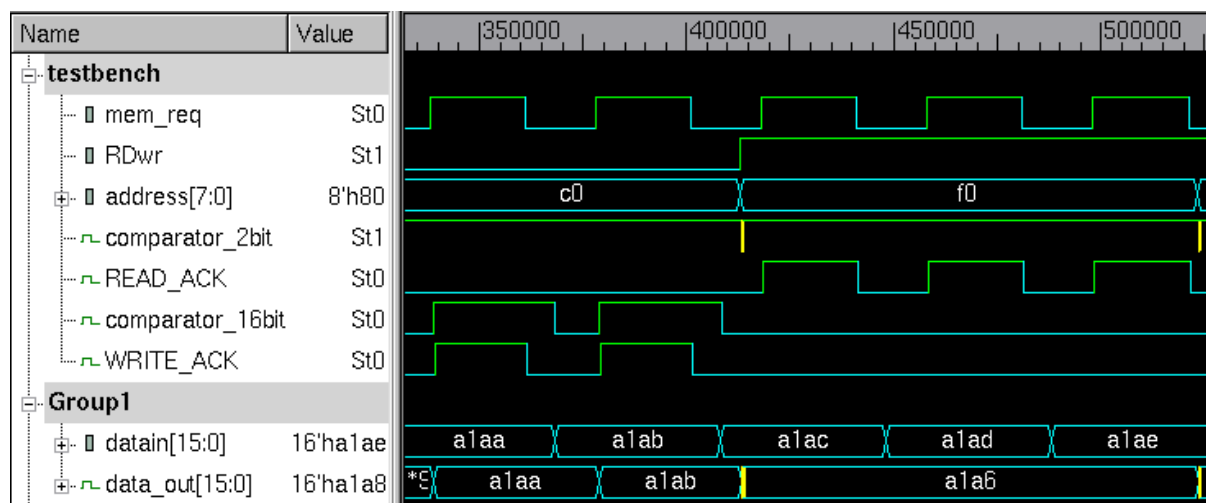

Figure 9: Complete read/write cycles

Simulation 2: Figure 6 demonstrates the operation of the 2-bit comparator and records the time taken by the module to evaluate its output. At time $\tau = 608$, the ADR has changed. The comparator reset its output to 0 after 0.5 nS, and at time $\tau = 608.7$ it raised a logic 1 indicating that the Register Bank has selected the correct register. At first glance it is seen that the comparator's propagation delay is 0.5nS, but it should be noted that most of the time period was spent by an address decoder, decoding an 8-bit ADR to the 2-bit ADR for the internal register. Other simulations performed clearly demonstrated that the comparator's actual propagation delay is 0.2nS.
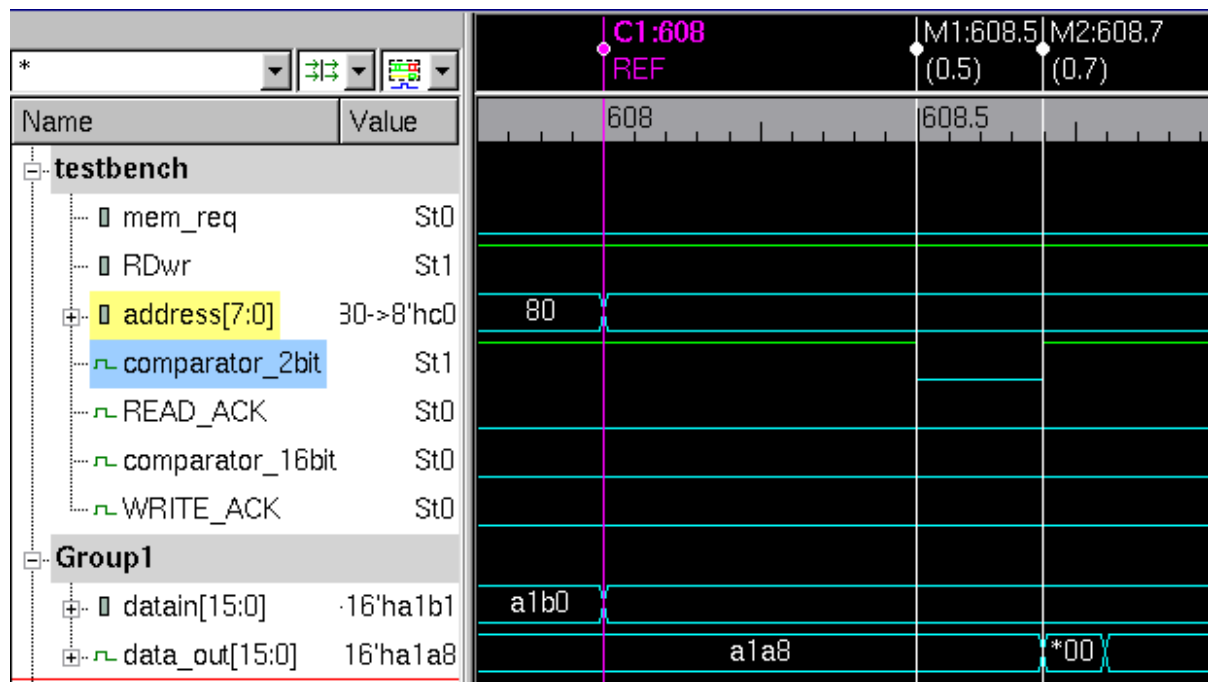
Figure 9: 2-bit comparator operation

Simulation 3: A read ACK is correctly generated by the module within 0.4nS from the instant that the REQ signal was issued.
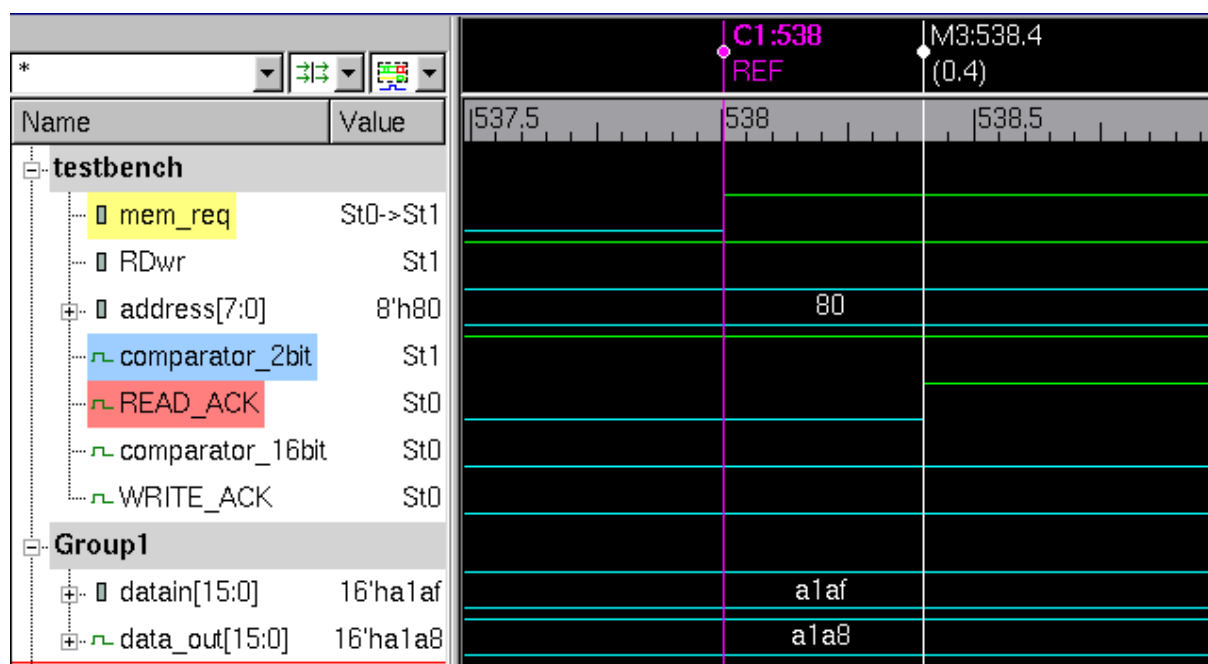

Figure 10: Read ACK generation

Simulation 4: The following figure determines the 16-bit comparator's propagation delay to be 0.3nS. That is the time taken form the instant that the data_input bus has changed value, to the instant that the comparator set its output to logic 0.
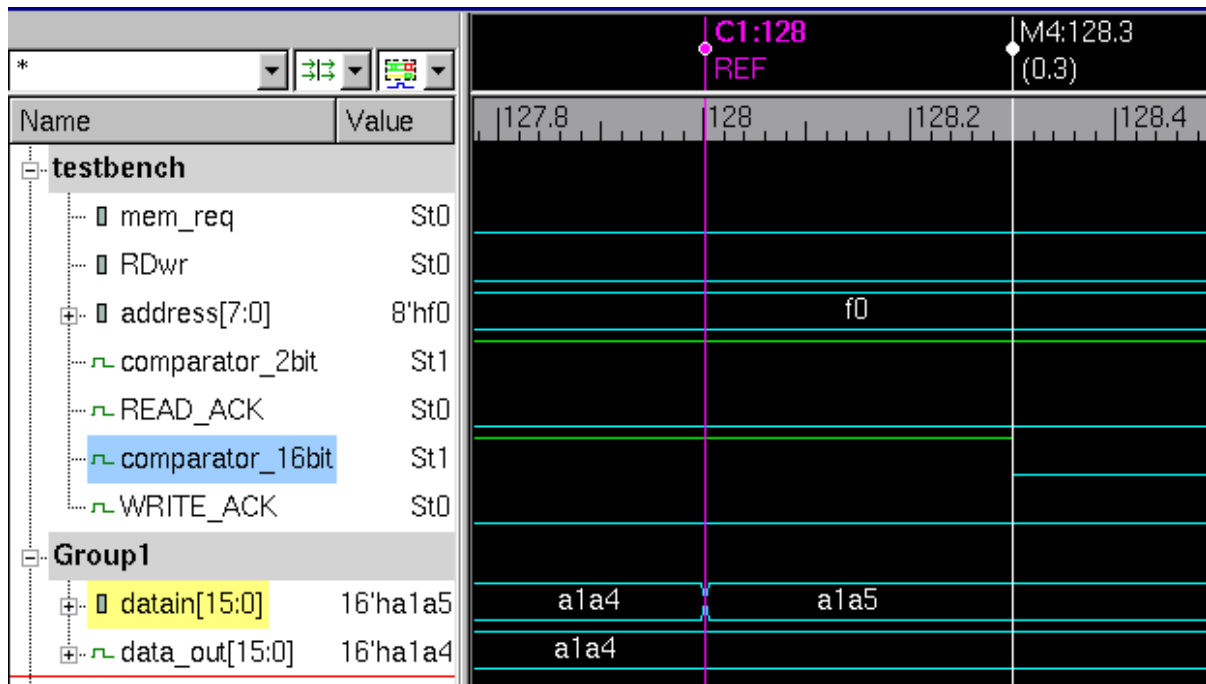
Figure 12: 16-bit comparator's propagation delay

Simulation 5: Figure 9 clearly displays all the signal transitions during a Write operation. At τ = 138nS a memory request is issued. Then, at time τ = 138.8nS (0.8nS after the REQ signal) all the bits have been correctly saved by the register and within the next 0.2nS the 16-bit comparator has raised its output. Finally, 0.1nS later a Write ACK is produced.
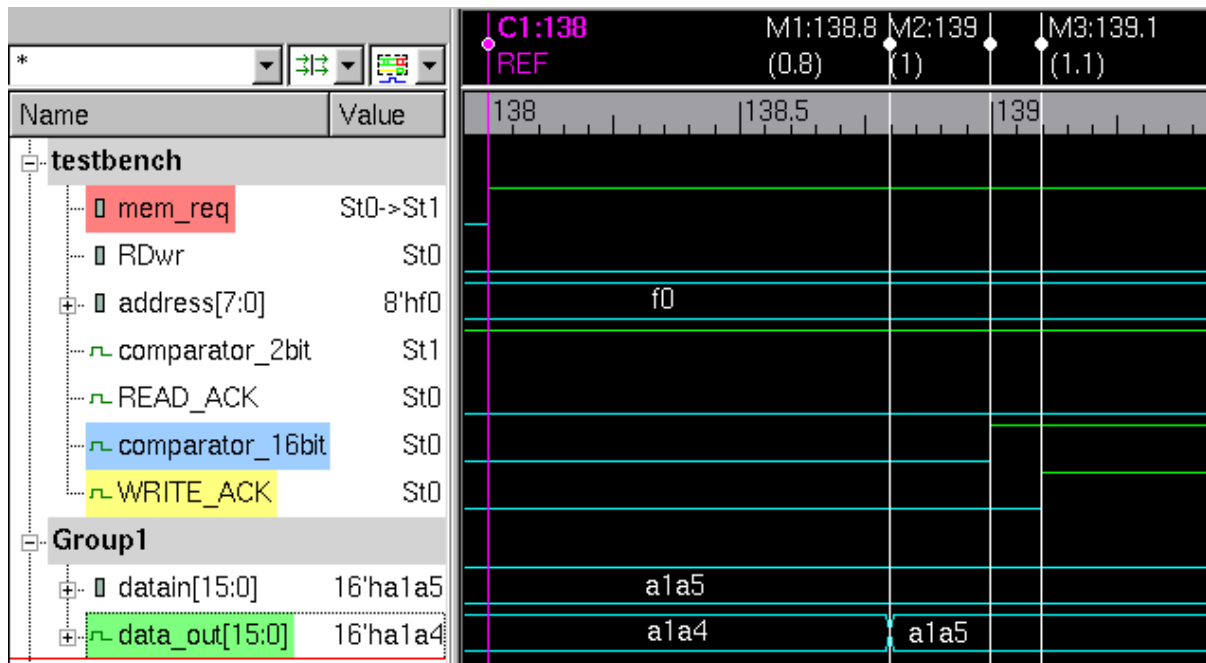

Figure 13: Write operation signal transitions

7. Discussion

All tests conducted on the Register Bank demonstrated positive results for both CD methods in terms of performance and reliability given their fault-free operation. Having said that, it is worth mentioning that the Read CD method used is less reliable than the Write CD one. The reason for this is the fact that while the multiplexers used for both the data and the ADR signals are identical, fabrication imperfections could lead to different propagation delays between the various Mux elements. Given the fact that the ADR bits go through a set of other gates (comparator, AND gate etc.) before an ACK it is raised, the latter's only chance of being produced falsely, is if the path followed by the data is faster than address path, by an amount equal to the propagation delay of the 2-bit comparator and the AND gate. It is vastly unlikely that the multiplexer elements would suffer than much variation. In addition to that, the CD method used here outperforms a delay block implementation from almost any point of view for the following reasons

1. It can be used under a wide variety of voltages and will always detect the instant that the data bits are available – no delay approximations
2. Does not need to be tuned for different supply voltages
3. In case of sequential reading cycles from the same register, no need for a delay to occur – an ACK will be generated almost instantly

In addition to these, the Write CD arrangement proved to be a solid method for detecting the instant of the Write operation completion. It can be used under any circumstances always yielding reliable ACK generation and it is considered far superior than any delay-block based implementation on the basis that

1. It is a more reliable design and it produces an ACK signal which is not only a true indicator of operation completion, but also assures that the data stored in the Register is error-free and that the Register itself is fully functional
2. It does not require a timing tuning as the adjustable delay line
3. It should yield a slightly higher performance in all supply voltages because the delay line tuning can never be perfect.

The only slight setback, regarding the Write CD method, is that if it is scaled for application to larger memories (32/64 bits), the area and power will increase too. This is due to the added X-OR gates used for comparison. On the other hand a delay line can be used for any memory size without imposing extra hardware requirements. This however works both ways. If the comparator method is to be used on smaller memories, it would require less area and power while the delay line would not. Therefore, the proposed CD solution might be more suitable to small memory devices (Register, SRAM with small data-bus etc).

8. Conclusions

This paper proposes a simple and effective design for an asynchronous Register Bank with CD on both modes of operation. The final design performed extremely well in the simulations, generating ACK signals upon operation completion. Both CD methods proposed offer a number of advantages

when compared with delay-based implementations, justified by a wealth of reasons. The entire Register Bank including the ACK generating module, forms a fully self-timed memory solution for low-power applications that do not require large amounts of storage and operate under non-deterministic voltage sources.

  Furthermore, the simple Write CD method can be applied on other types of memory with larger storage ability, as it proved to be a solid write CD method. This however requires further investigation and comparison with current CD techniques.

9. References

[1] "Reconfigurable Asynchronous Intel 8051 Microprocessor." [Online]. Available: http://async.org.uk/chip-gallery.html.

[2] A. Mokhov, M. Rykunov, D. Sokolov, and A. Yakovlev, "Design of Processors with Reconfigurable Microarchitecture," *J. Low Power Electron. Appl.*, vol. 4, no. 1, pp. 26–43, Jan. 2014.

[3] Elia M. "Asynchronous Memory Controller with Data Validation".