Paper IV B 2

Data Acquisition Circuit for Tire Inspection Machine

Kevin Dazey

Electrical and Computer Engineering Student
Baylor University

Michael W. Thompson
Associate Professor
Engineering Department
Baylor University

Abstract

This paper describes a student project that involves the design of a digital data acquisition triggering circuit to be implemented on a Field-Programmable Gate Array (FPGA) chip. The project uses Xilinx Foundation Series Software for schematic design and simulation. The first major step of the design process was defining the project specifications. In addition to exploring the developmental history of this project, methods for improving the most current implementation of the triggering circuit were investigated. A prerequisite of this was to study the original schematic diagrams to determine the current design's functionality (which was no simple task). This paper describes a "redesign" of the circuit using a more formal state-diagram approach. The implementation of the design is accomplished through truth tables and a logic gate realization of system equations for a "ones-hot" state machine implementation. Using the Foundation Series software, the design was implemented on a XESS XS40 board. A motor encoder provides the timing signals for triggering the data acquisition process.

Introduction

This project originated with Dr. Mick Peterson, at the University of Maine. Dr. Peterson is a Mechanical Engineer, and is a consultant of the company that actually uses the design for testing the integrity of tires with an ultrasonic inspection machine. Dr. Peterson works in the field of nondestructive testing, which is used in this tire inspection process. This project was used as a substantial portion of the work accomplished in an independent study course at Baylor University.

Design Specification

Project Definition

One of the most time-consuming portions of this project was the interpretation and understanding of the design specification for the triggering circuit.

Project Definition

An initial description of the project was found in a personal correspondence, [1] which is summarized below:

> A design is needed for triggering a data acquisition system for ultrasonic scanning of tires, testing a tire's integrity. The scanning will be done in a circular manner on the sidewall of the rotating tire, and will scan at several different radii of the tire. The circuit should use a 2048-pulse encoder to acquire at least 400 data lines. Each data line corresponds to one full revolution of a tire, each at a different radius. A complete scan will take less than 7 minutes, scanning as many as 250,000 total data points. An input signal (referred to as an A-Pulse) is sent to the circuit every time a data point is taken. To avoid errors in counting these pulses, the design should have a periodic reset interrupt (Zero-pulse or Z-Pulse) at each full revolution. The circuit should ignore false Z-pulses by implementing a Z-Enable signal. The Z-Enable will be set high when the count of A-Pulses (data points) indicates that approximately one revolution has occurred. The Z-pulse along with the data point interrupts (A-pulses) will be transmitted as complementary signals to avoid common mode noise.
>
> Data will be acquired on alternate revolutions, implemented by a Z-Toggle, which alternates between 0 and 1 at each full revolution. The machinery will have time to increase in radius (mechanical movement) while Z-Toggle is low. Therefore, data will be acquired only when Z-Toggle is high. Another system input is the Motor_Clear signal, which is low while there is mechanical motion of the scanner.
>
> On power up, after the initial reset, one full rotation should occur before Z-Enable is set to 1. Z-Enable will be set when the count of A-Pulses is greater or equal to 2044, leaving a small window to look for the true Z-Pulse. Z-Toggle will change states while Z-Enable is high and when the true Z-Pulse is read. If the Z-Pulse is not detected, the system should skip acquiring data for one revolution and wait for the next Z-Pulse.

Design Process

Functional Design

This high-level design, while it may look simple, was not without problems. The first attempt at this high-level simulation was to use Xilinx Foundation Series' State Diagram Editor, shown below in Figure 1.

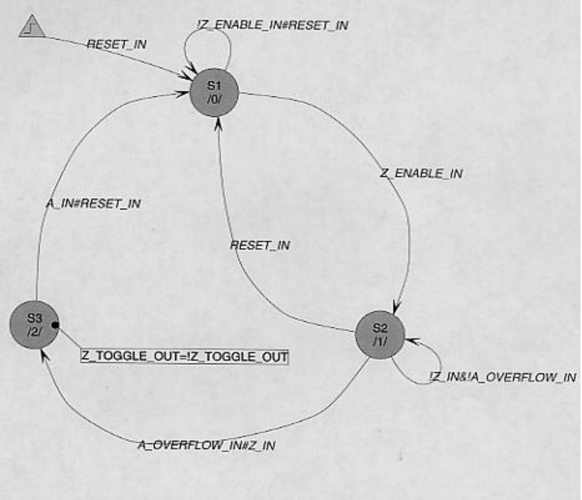

Figure 1. Xilinx State Diagram Editor

After trial-and-error, the states were labeled and the conditions for state transitions were generated. A detailed description of this type of approach is found in the literature[2]. This design was then converted into VHDL for simulation. The VHDL code generated by the Foundation tools was edited to arrive at a VHDL representation of the design. This design compiled, and started to simulate, as shown in Figure 2, but would freeze upon reaching the third state. Due to the author's lack of familiarity with VHDL another approach using a C++ simulation was implemented. This proved to be much simpler and easier to debug. While the code was very simple, it clearly defines the system's three states and their transitions. This allowed for a very natural transition into a state system representation.

State Design

One of the original goals of the project was do was to implement a one's hot state machine implementation (see Wakerly[3] for a detailed description of a one's-hot state machine design). An initial design of the state system is shown in Figure 2. This approach was found to generate an incorrect interpretation of the design specification. The original state design acquired data on every tire revolution never allowing the mechanical movement of changing radii to take place.

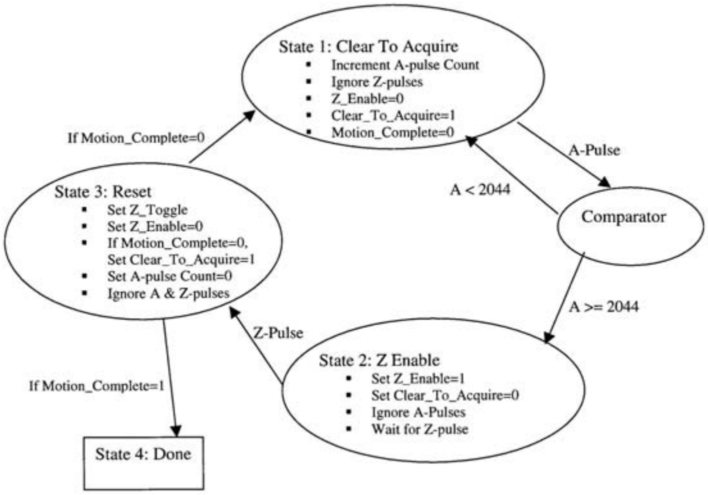

Figure 2. Initial State System Definition

The design was modified to accommodate this change and logic equations for state transitions were generated from truth tables. Difficulties were encountered when designing a single one's hot state design. It was finally realized that it was overly complicated to have a single one's hot state machine design that functions on every other tire revolution. The solution to this problem is to use two state machines in the design: one for the main state system and another for the Z-Toggle system. The final version of the State Definition is shown in Figure 3, including both state machines.

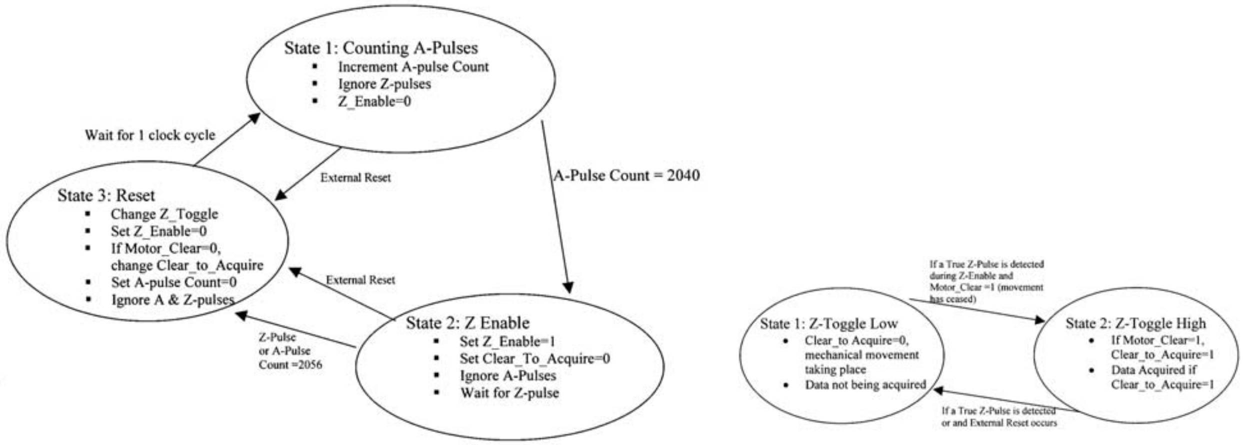
State 1: Counting A-Pulses
Increment A-pulse Count
Ignore Z-pulses
Z_Enable=0
Wait for 1 clock cycle
External Reset
A-Pulse Count = 2040
State 3: Reset
Change Z_Toggle
Set Z_Enable=0
If Motor_Clear=0, change Clear_to_Acquire
Set A-pulse Count=0
Ignore A & Z-pulses
External Reset
Z-Pulse or A-Pulse Count =2056
State 2: Z Enable
Set Z_Enable=1
Set Clear_To_Acquire=0
Ignore A-Pulses
Wait for Z-pulse
If a True Z-Pulse is detected during Z-Enable and Motor_Clear =1 (movement has ceased)
State 1: Z-Toggle Low
Clear_to Acquire=0, mechanical movement taking place
Data not being acquired
State 2: Z-Toggle High
If Motor_Clear=1, Clear_to_Acquire=1
Data Acquired if Clear_to_Acquire=1
If a True Z-Pulse is detected or and External Reset occurs

Figure 3. Main State System (left) and Z-Toggle State System (right)

System/Logic Design

The design features a CB4CE 4-bit binary counter (from the Foundation library) with a clear and enable inputs. It is desirable for the counter to have enable and asynchronous set/reset capabilities, so that an external asynchronous reset would be possible. These were cascaded to make a 12-bit counter for the A-Pulse Counter System, shown in Figure 4.

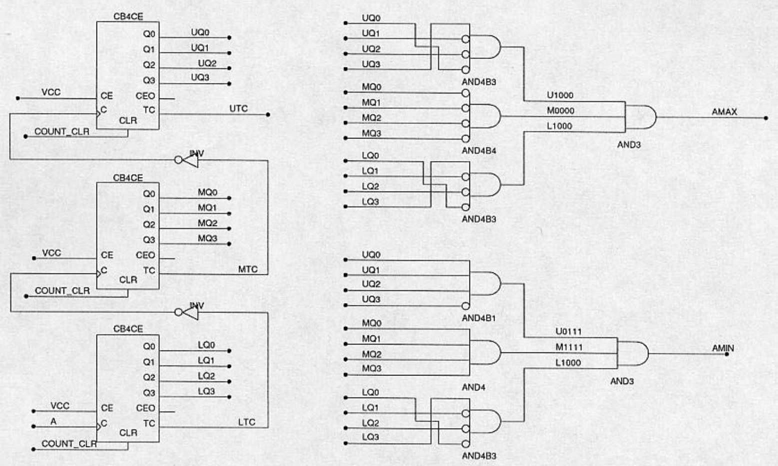

Figure 4. Cascaded 12-bit Counter Schematic

An input synchronizer system was added (the original design did not have this feature), as shown in Figure 5. The synchronizer circuit is believed to increase the reliability of the triggering circuit design[3].

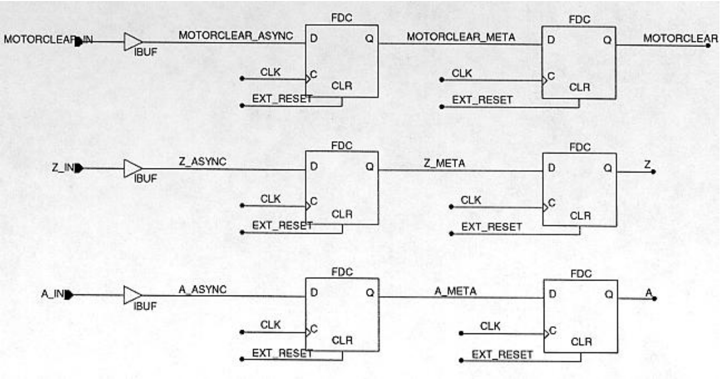

Figure 5. Input Synchronizer Schematic

Next, the state system was developed, shown in Figure 6, beginning with the logic equations from the truth table mentioned earlier.

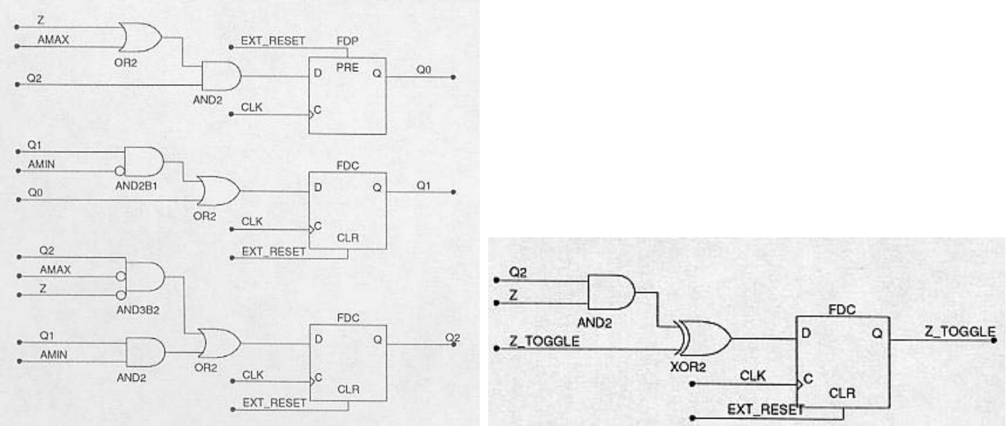

Figure 6. State System Schematic

Finally, the three main systems (synchronizer, counter, and state systems) were combined into a simple Overhead System, shown below in Figure 7.

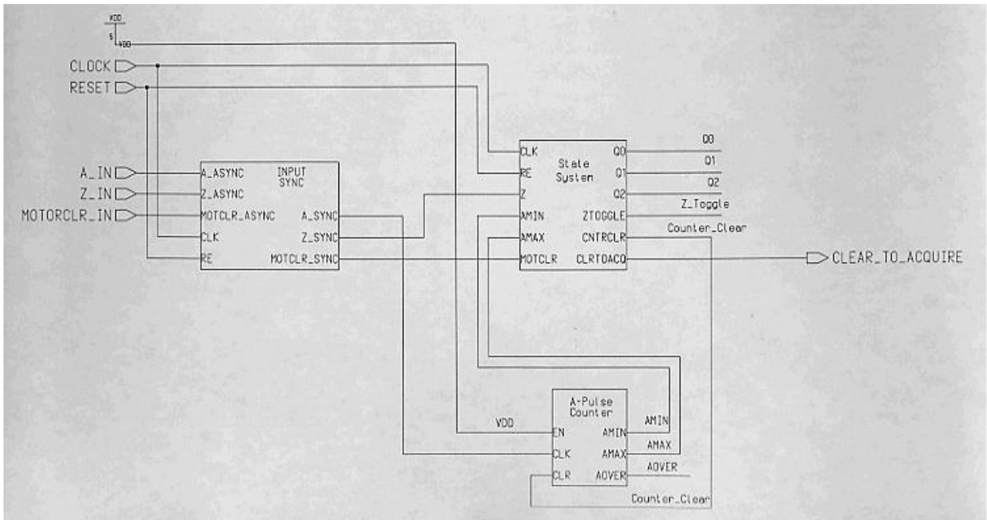

Figure 7. Overhead System Schematic

Simulation Process

Software Simulation

Most of the software simulation was done using Mentor Tools software, and later using Xilinx Foundation tools. Functional tests were run for each component, from the 4-bit Counter to the combined Overhead System shown above in Figure 7. The input signals to each component were manipulated to verify correct functionality, while the outputs were monitored and recorded. In addition to tests designed to check normal operation, several tests were run to check for abnormal situations, such as missing Z-Pulses or external resets. Figure 8 shows normal operation of the entire Overhead System.

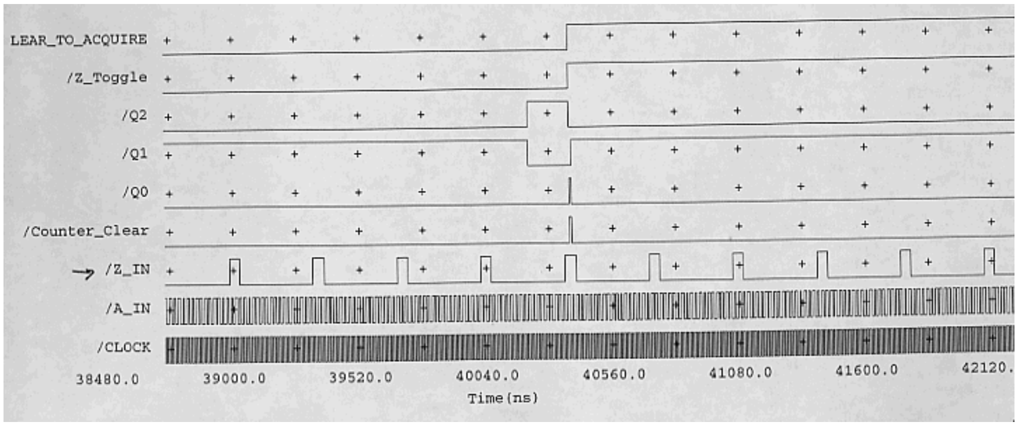

Figure 8. Simulation of Overhead System

FPGA Simulation

Now that the software schematic in Foundation Series has been generated and debugged, attempts are being made to program the FPGA and test the circuit using the hardware configuration used by the data acquisition system. We have access to a dc motor with an attached shaft encoder for testing the design (as shown in Figure 9).

Figure 9. DC Motor and Shaft Encoder

The purpose of the motor and encoder is to generate a series of pulses to be interpreted as A-Pulses by our circuit. It effectively sends the FPGA a clock signal, whose frequency is proportional to the voltage supplied to the motor. After overcoming some initial difficulty in downloading the design onto the Xilinx chip, we are now in the process of performing hardware testing of our design. Final design verification will be accomplished by using the design in the tire-scanning data acquisition system and comparing the operation to the existing design. The Xilinx FPGA Laboratory board used for this project is shown below in Figure 10.

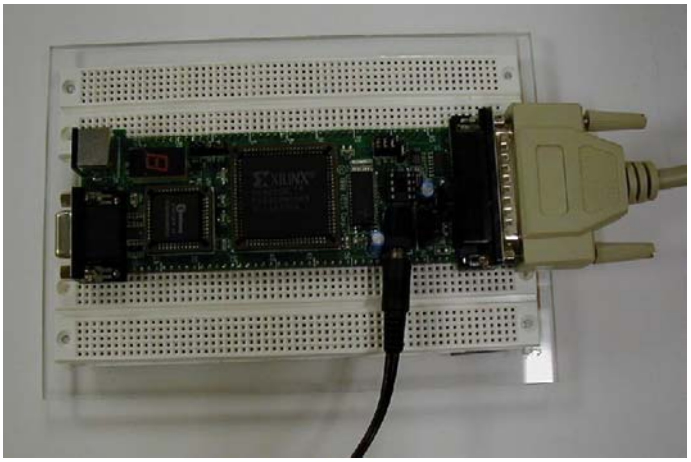

Figure 10. Xilinx FPGA Laboratory Board

Conclusion

The design that we have implemented is the fourth version of a data acquisition triggering circuit. The original circuit was hand wired, using discrete electronic design on a breadboard. It was then adjusted for implementation on a Printed Circuit Board (PCB), operating in a more robust manner than the previous design. The next version was done on a Complex Programmable Logic Device (CPLD), which allowed the design to be edited without ordering a new board. This was the design that existed when we were presented this project. Our new design is significantly improved over the CPLD design and uses an FPGA chip, which is a type of CPLD. The project's life cycle is far from over, as should be expected. As long as there are advances in circuit design technology, there will be ways to improve this particular project. One of the most crucial elements of any design project is determining how it can be further improved upon. That is one of the main advantages of an FPGA implementation: it allows the designer to change the design instantaneously and without cost whenever the application requirements change.

References

1. Dr. Mick Peterson, Department of Mechanical Engineering, Personal Correspondence.
2. David E. Van Den Bout, *The Practical Xilinx Designer Lab Book : Version 1.5,* Prentice Hall, January 1999.
3. John F. Wakerly, *Digital Design, Principles & Practices: 3rd Edition,* Prentice Hall, 2000.

KEVIN DAZEY
Kevin Dazey is currently an undergraduate student in Baylor University's Electrical and Computer Engineering program. He is in his second semester of Independent Study in this project with Dr. Thompson. He will graduate in May of 2002 with a Bachelor of Science in Engineering from Baylor University. Kevin is a member of Baylor's chapter of the Institute of Electrical and Electronics Engineers (IEEE).

MICHAEL W. THOMPSON
Dr. Thompson is an Associate Professor in the Engineering Department at Baylor University. His research interests are signal processing and statistical communications theory. Dr. Thompson served as the supervising professor for Mr. Dazey's independent study project.