Homework 7: VGA Circle Report
December 11, 2017
Sabbir Ahmed

1 Background

In this project students will explicitly implement a computational finite-state machine, utilize rescheduling and resource sharing, and become familiar with the concept of using an on-chip clock multiplier. Students will leverage the faster clock to implement computations in a serial fashion. In this HW students will display a circle on the screen, examine analysis reports, and modify synthesis options.

2 Design Approach

The source code for interfacing with UART was provided. The controller FSM was implemented to provide all the glue logic required to interface the UART with the `sqrt` module and the block RAM. The multiplexer controlling the `tx_data` in the datapath is integrated into the controller module as the states `ramhi` and `ramlo`. Figure 1 provides the top-level block diagram of the project implementation.

2.1 Controller FSM

The controller FSM consists of 7 states visualized in Figure 2.

2.1.1 addrhi

This is the initial state of the controller to detect when the high bytes of the RAM block address from `rx_data` have been transferred from the UART. The state proceeds to `addrlo` when the shifter at `rx` is empty and `AH` is ready.

2.1.2 addrlo

Activated when the shifter at `rx` is empty and `AH` is ready. This is the second state of the controller to detect when the low bytes of the RAM block address from `rx_data` have been transferred

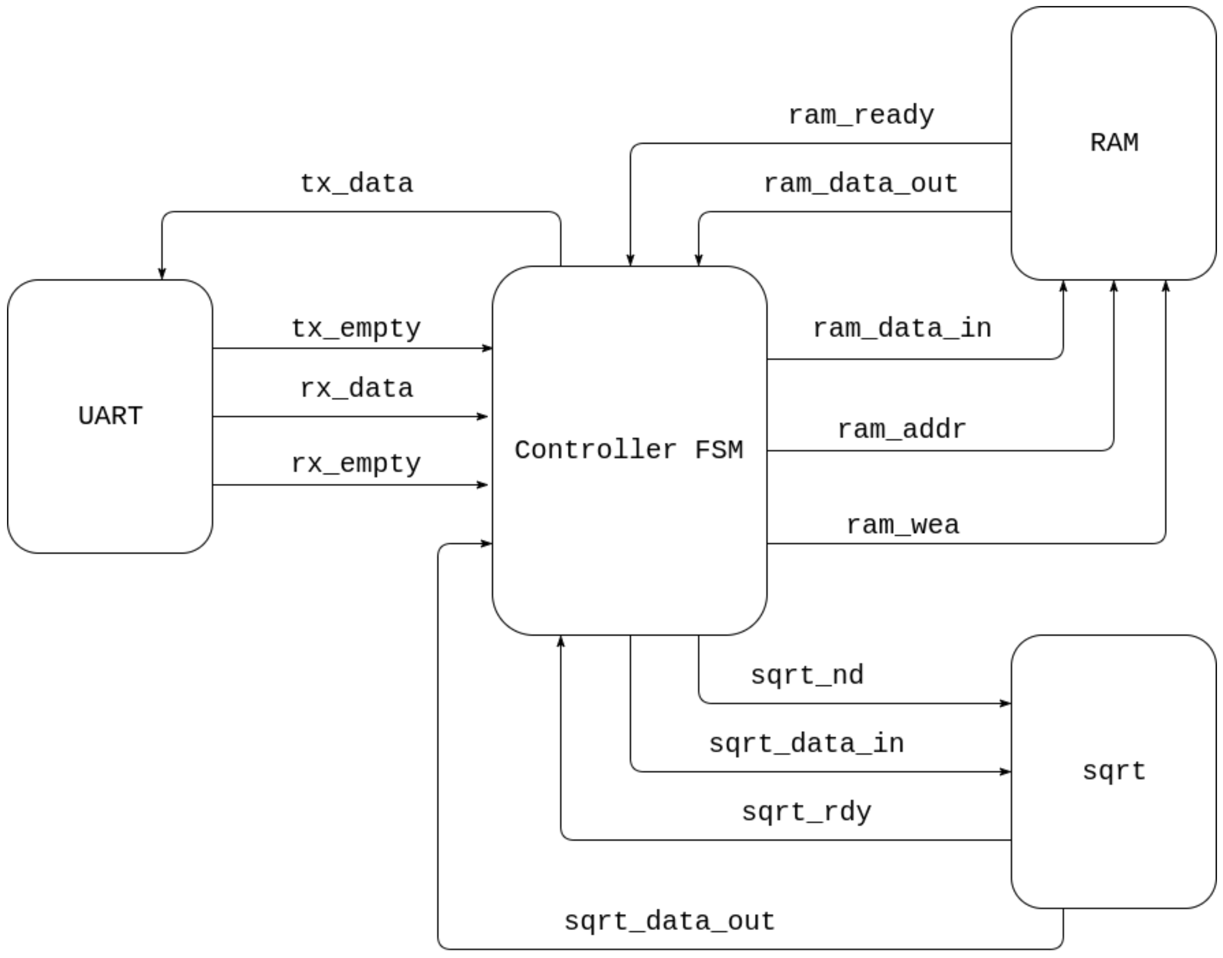

Figure 1: Block Diagram of the Implementation

from the UART. The state saves the entire 12-bit address of the RAM block and then proceeds to `datahi` when the shifter at `rx` is empty and `AL` is ready.

2.1.3 datahi

Activated when the shifter at `rx` is empty and `AL` is ready. This is the third state of the controller to detect when the high bytes of the RAM block input data from `rx_data` have been transferred from the UART. The state proceeds to `datahi` when the shifter at `rx` is empty and `DH` is ready.

2.1.4 datalo

Activated when the shifter at `rx` is empty and `DH` is ready. This is the fourth state of the controller to detect when the low bytes of the RAM block input data from `rx_data` have been transferred

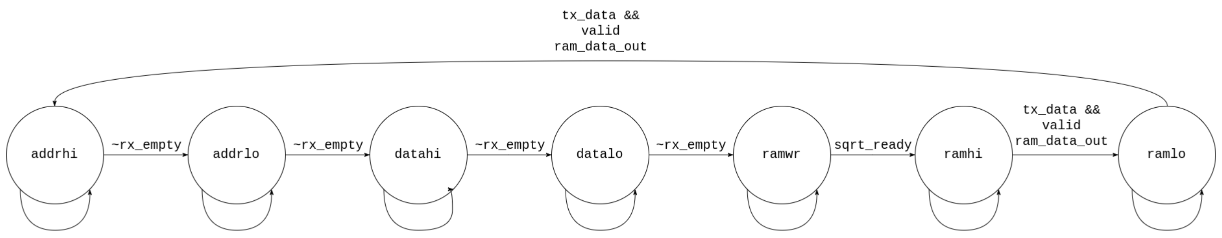

Figure 2: State Diagram of the Implementation of the Controller

from the UART. The state sends the entire 16-bit data to the `sqrt` module and then proceeds to `ramwr` when the shifter at `rx` is empty and `DL` is ready.

2.1.5 ramwr

Activated when the `sqrt` module is and `DL` is ready. This is the fifth state of the controller to write to the RAM block. The state sends the entire 12-bit address and the output of the `sqrt` module to the RAM block and then proceeds to `ramhi`.

2.1.6 ramhi

Activated when the shifter at `tx` is empty and data read from the RAM block is valid and ready. This is the sixth state of the controller to read from the RAM block with the same address and passed back to the UART. The state sends the high 8-bit data read from the `sqrt` module to the `tx_data` in the UART.

2.1.7 ramlo

Activated when the shifter at `tx` is empty and data read from the RAM block is valid and ready. This is the final state of the controller to read from the RAM block with the same address and passed back to the UART. The state sends the low 8-bit data read from the `sqrt` module to the `tx_data` in the UART.

3 Testing

The project submission includes test benches for the controller as well as the top-level module integrating all the modules. Figure 3 shows an example use of the implementation with the memory file contents: 01, 01, 00, 04.

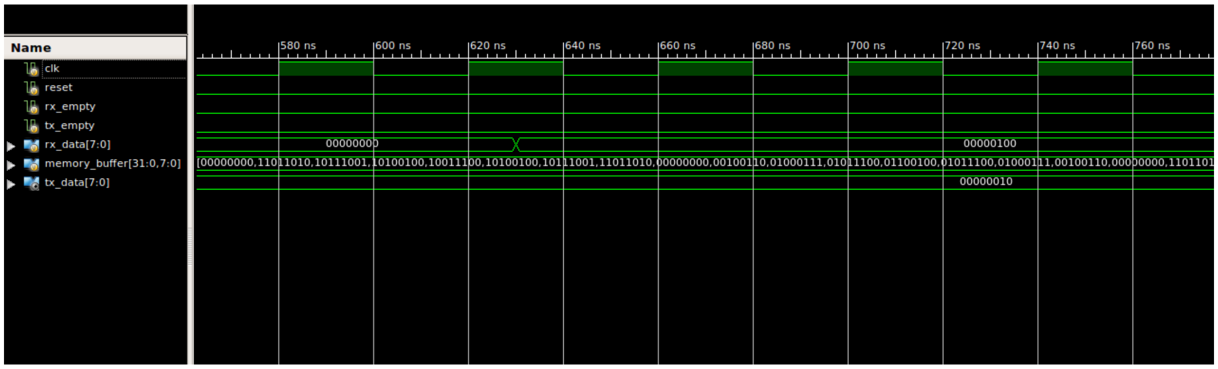

Figure 3: Sample Testbench Waveform of the Controller Module

Figure 3 demonstrates the unit after its final state. The address to the RAM block was passed as 0001 0000 0001 (low 4 bits of the high 01 hex value and the entire low 01 hex value). The data passed to the `sqrt` module was 0000 0000 0000 0100 (00 and 04). The `sqrt` module returns 0000 0010 ($\sqrt{4} = 2$).