The goal of the assignment is to develop an understanding for the different Verilog concepts regarding full system level design. The is the 2nd of two projects, focusing on DataPath and Controller implementations as pertinent to a store and forward router.

***Project 2 – Pt 1, 2, & 3***

The following diagram is encapsulates Parts 1, 2, and 3 of the project, showcasing all details of the router, the details of the DataPath, and the handshaking that will be performed by the Controller:

**Register Transfer Level (RTL) Router – DataPath and Controller Layout & Interconnections**



The following information describes the design and assumptions behind each component of the system:

**Data Ports**:

* The Data Ports are independent of the clock, so they received data *in real time*, as would happen in an actual routing system.
* All Data Ports are connected to one unified bus, called *dp\_bus*¸which will determine what information will eventually get read into the buffer of the router based on which port submitted the request. Signals from the Controller will dictate which data is actually transmitted.
* If the Data Port was selected and acknowledged to transmit information, it will assert *ready* and visually confirm to the test bench that transmission will be occurring.

**Router**:

* The main memory structure is its *buffer* (16 word register, with 8 bits per word).
* The *rst* command will clear all contents of *buffer* and set them = 0.
* When *st\_router* is first asserted, then the Router undergoes 16 clock cycles of filling in its *buffer* with data *@posedge clk* from the respective Data Port that sent the *request*. The *buffer* is filled sequentially using the counter *inAddr* from the Controller.
* When *fw\_router* is first asserted, which happens directly after the 16 clock cycles of *st\_router*, each word of the *buffer* it sent to the *output\_port*, one at a time, *@posedge clk*. The *buffer* is read sequentially using the counter *outAddr* from the Controller.

**Controller**:

* Prime responsibility is to handshake signals between all different storage elements and the outside world (the Testbench, in this case).
* Contains a total of four states:
  + IDLE – The system hangs here until it sees that a *request* has been made by one of the Data Ports. In this case, the model assumes that the Controller would know when the request is being sent, which is why the input comes from the outside world, and not the Data Ports themselves.
  + ACK – Need one clock cycle for the router to acknowledge that it is ready to receive data. The *acknowledge* signal is asserted when *st\_router* and *fw\_router* are both 0.
  + STORE – For a total of 16 clock cycles *@posedge clk*, the *buffer* of the Router is filled sequentially (using *inAddr*) to the *output\_port*. The system then returns to its IDLE state, and awaits another request from a Data Port to arrive.
  + FORWARD – For a total of 16 clock cycles *@posedge clk*, the *buffer* of the Router is output sequentially (using *inAddr*) with the data from the Data Port from which the *request* was initially sent. The system them waits for another *request* to be sent from another Data Port.

The *Router* and *DataPorts* make up the *DataPath*, with the *Controller* handling the handshaking between all components of the *rtlRouter*.

This concludes the analysis for the Project 2, Parts 1, 2, & 3.

***Project 2 – Pt 4 & 5***

The Verilog code for this section can be found in the Pt 4 & 5 folder of the ZIP file submitted:

*DataPort.v*

*Router.v*

*DataPath.v*

*Controller.v*

*rtlRouter.v*

*rtlRouterTester.v*

An inspection of all these Verilog files will show that the system was implemented exactly according to the System Design as described in Pt 1, 2, & 3 of this project.

A Testbench was developed in order to showcase the operability of the system implementation. Output waveform files are found in the Pt 4 & 5 folder of the ZIP file submitted:

*WF\_DataPort1.bmp*

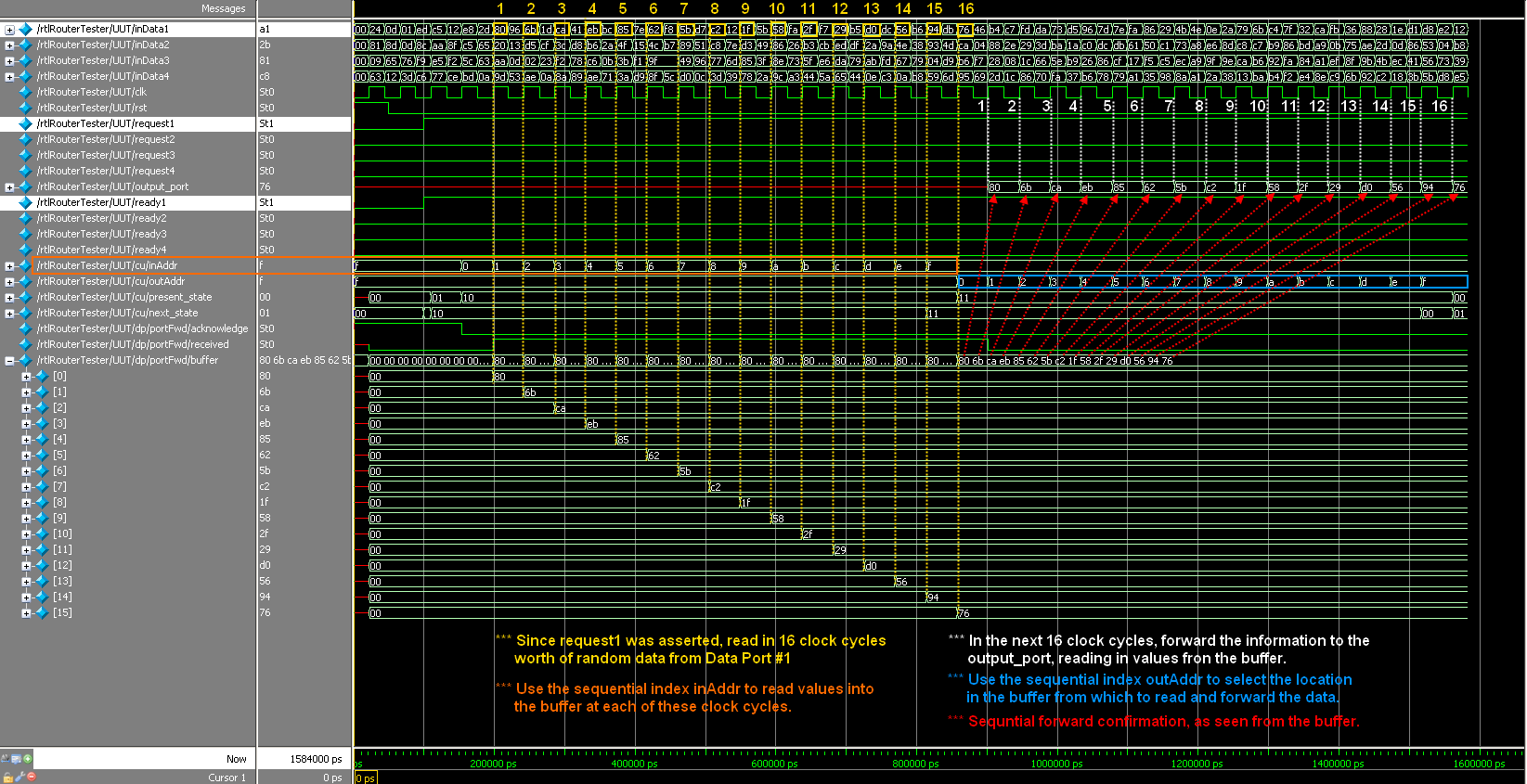
*WF\_DataPort2.bmp*

*WF\_DataPort3.bmp*

*WF\_DataPort4.bmp*

In an effort to showcase the functionality of the design, the *WF\_DataPort1.bmp* file has been edited to highlight the correct implementation, operation, and timing of this RTL design. This same analysis can be performed for the other output files. The output is seen on the next page:

***WF\_DataPort1.bmp Waveform Output***

****

This concludes the analysis for the Project 2, Parts 4 & 5.