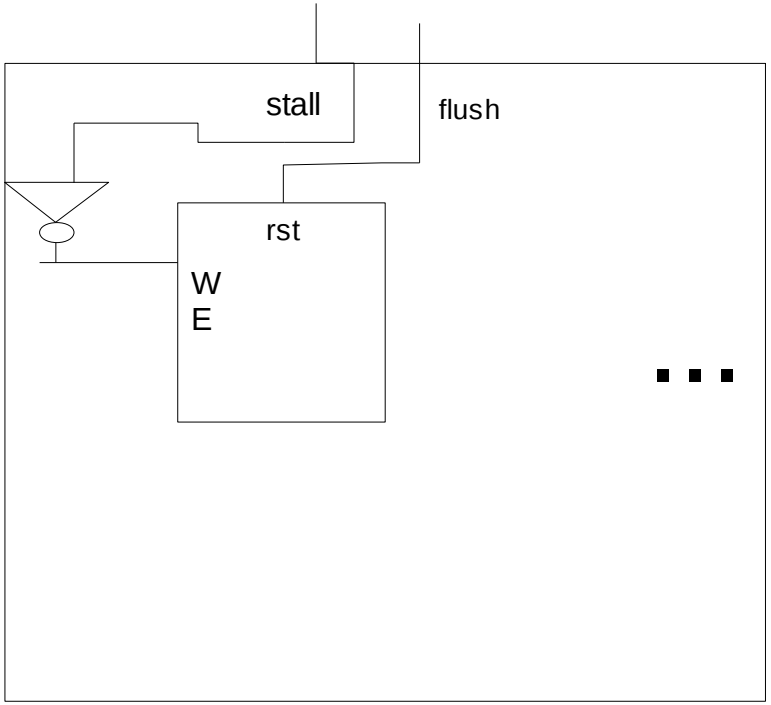
0. Prelab

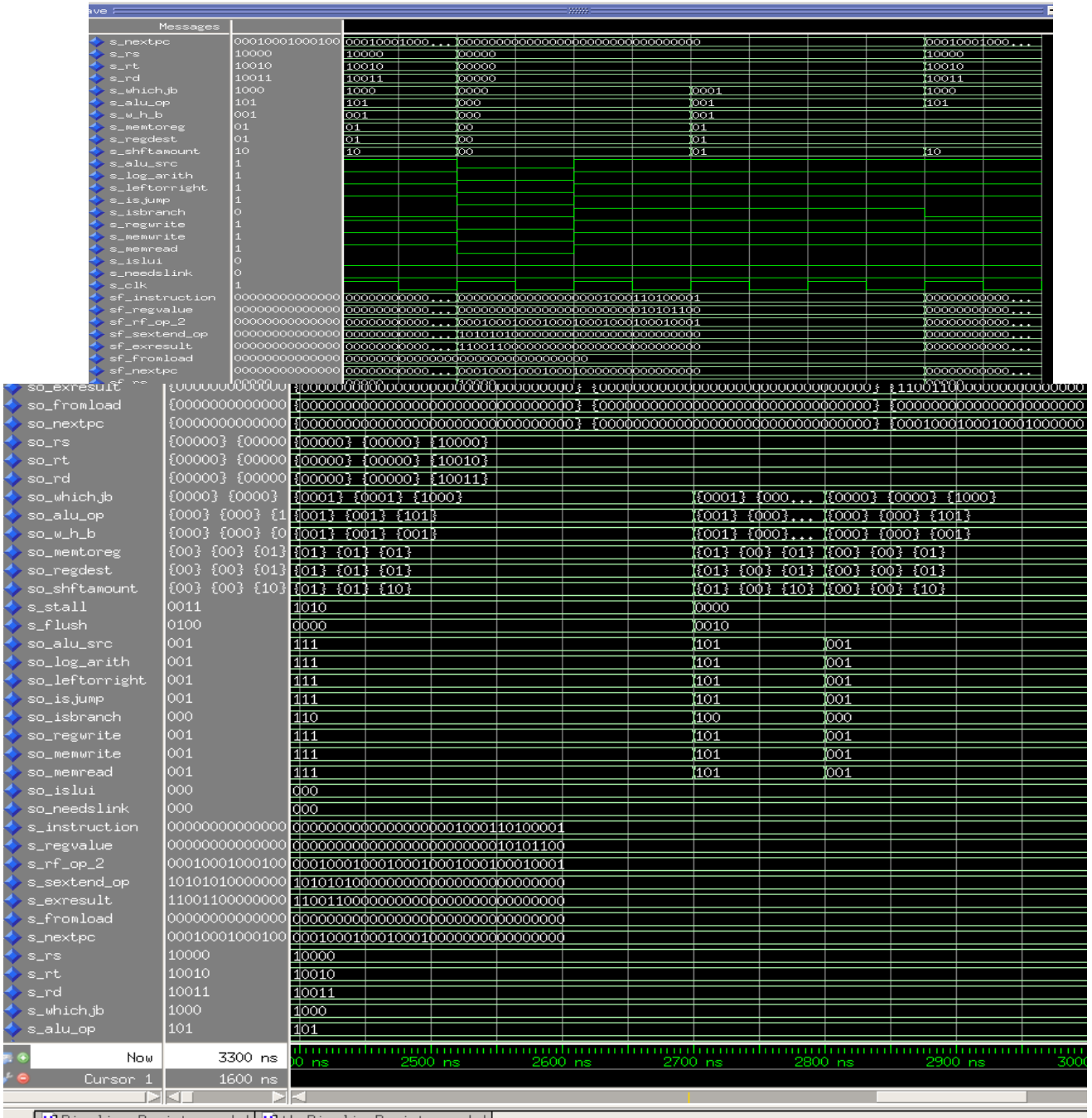
This is in the file omniscient spreadsheet.

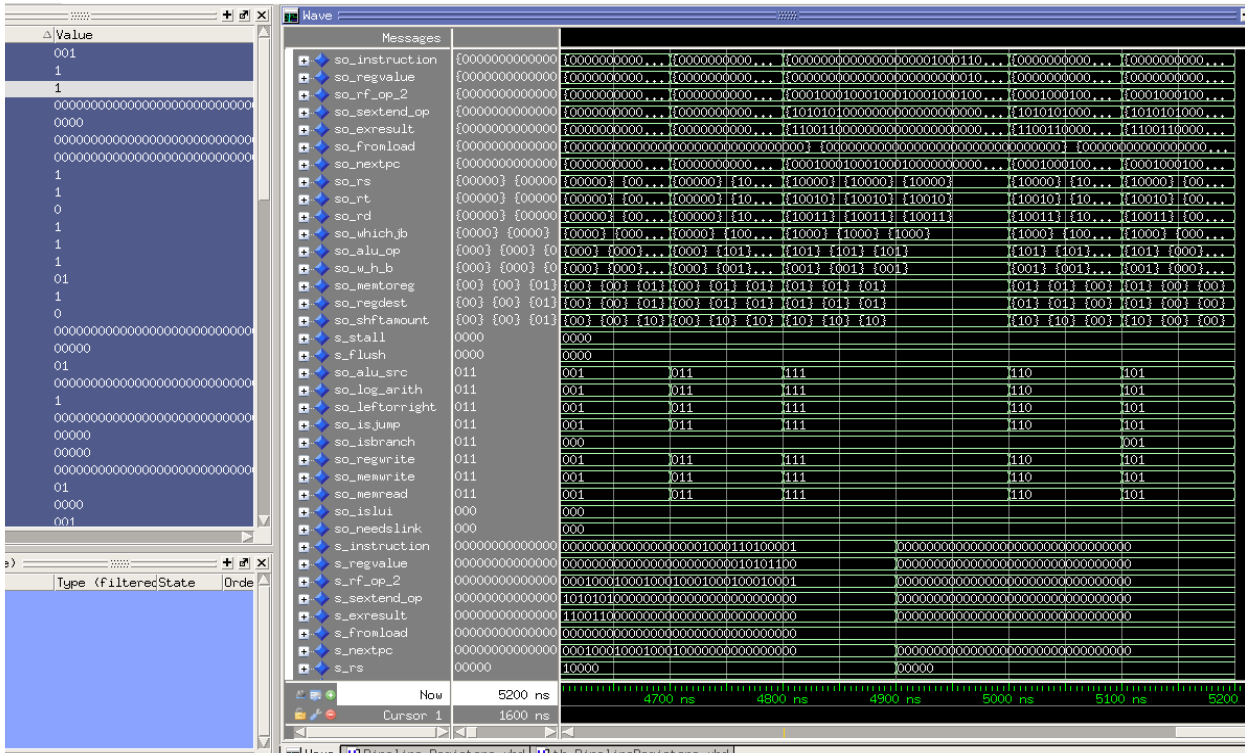
1. Pipelined Registers

a) To implement an ideal N bit register with stalling and flushing we can simply allow the stall and flush to access the reset and write enable bits. Then we just not the stall and set reset directly to the flush input and it would be implemented if it was ideal.



b) We created the file pipeline registers and a test bench which was verified by our TA. Results of the test bench:


Messages
s_nextpc
s_rs
s_rt
s_rd
s_whichjb
s_alu_op
s_w_h_b
s_memtoreg
s_regdest
s_shftamount
s_alu_src
s_log_arith
s_leftorright
s_isjump
s_isbranch
s_regwrite
s_memwrite
s_memread
s_islui
s_needslink
s_clk
sf_instruction
sf_regvalue
sf_rf_op_2
sf_sextend_op
sf_exresult
sf_fromload
sf_nextpc
so_exresult
so_fromload
so_nextpc
so_rs
so_rt
so_rd
so_whichjb
so_alu_op
so_w_h_b
so_memtoreg
so_regdest
so_shftamount
s_stall
s_flush
so_alu_src
so_log_arith
so_leftorright
so_isjump
so_isbranch
so_regwrite
so_memwrite
so_memread
so_islui
so_needslink
s_instruction
s_regvalue
s_rf_op_2
s_sextend_op
s_exresult
s_fromload
s_nextpc
s_rs
s_rt
s_rd
s_whichjb
s_alu_op
Now
3300 ns
Cursor 1
1600 ns
2500 ns
2600 ns
2700 ns
2800 ns
2900 ns



2. Data Dependencies

All parts number two can be seen in the Omniscient Spreadsheet on the last three tabs.

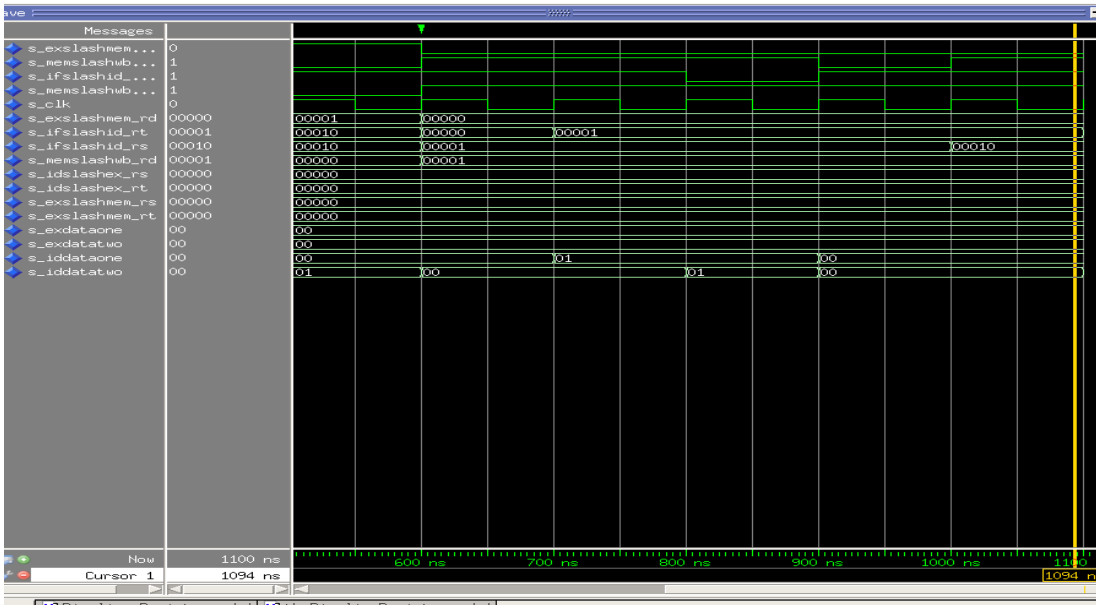
3. **Forwarding and Hazard Detection**

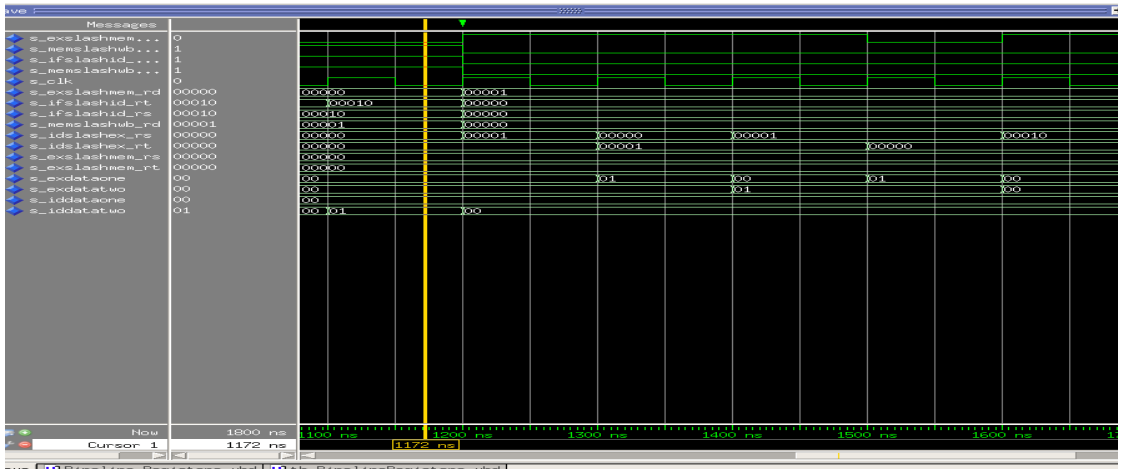
   Shown in the Dependencies tab of the Omniscient Spreadsheet.

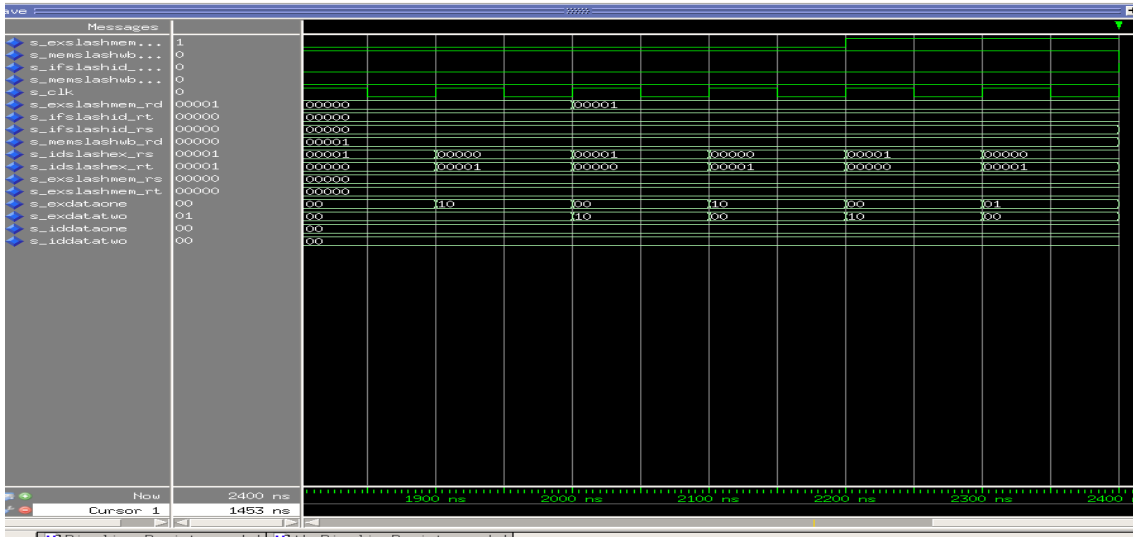
4. **MIPS Pipelined Processor**

After putting in the pipelined registers we then created a Forwarding unit and Hazarding Unit which the test benches are listed shown below.

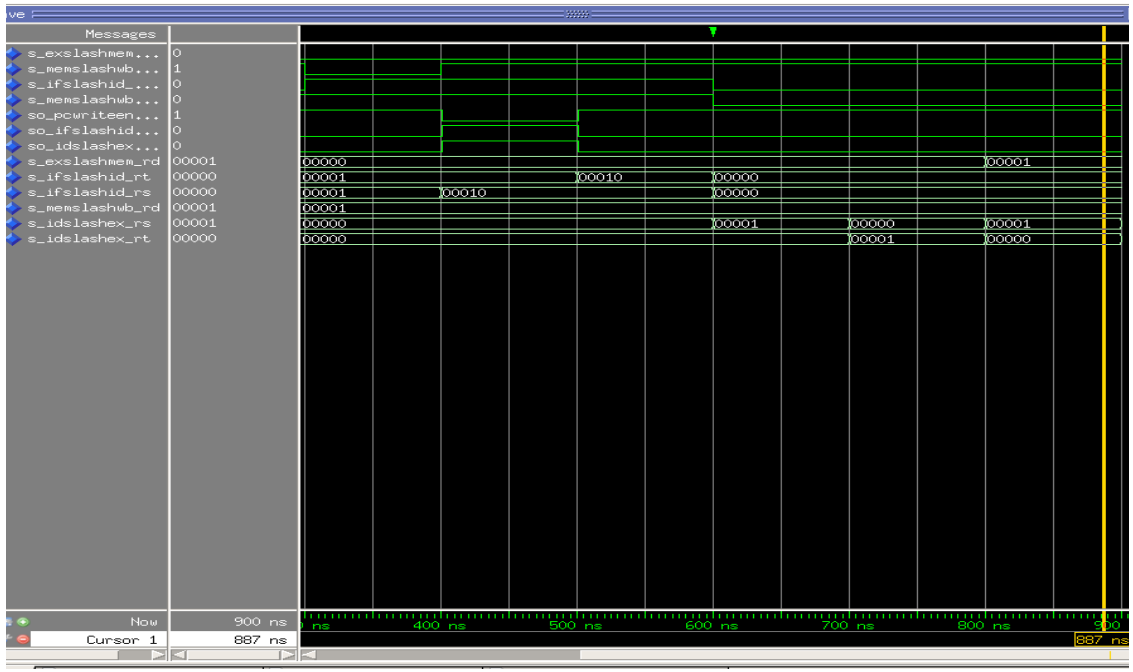
Forwarding:

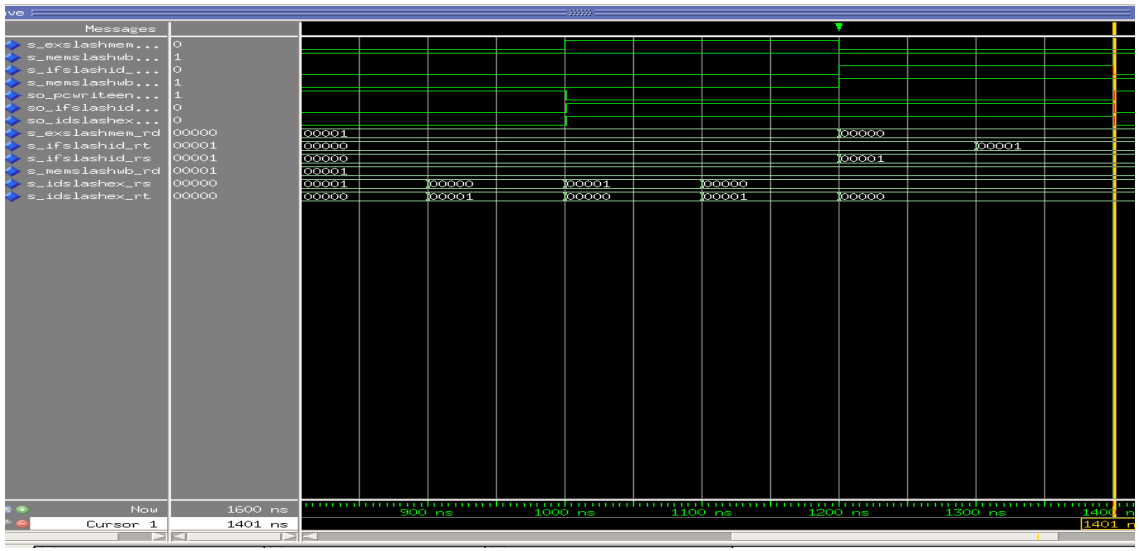


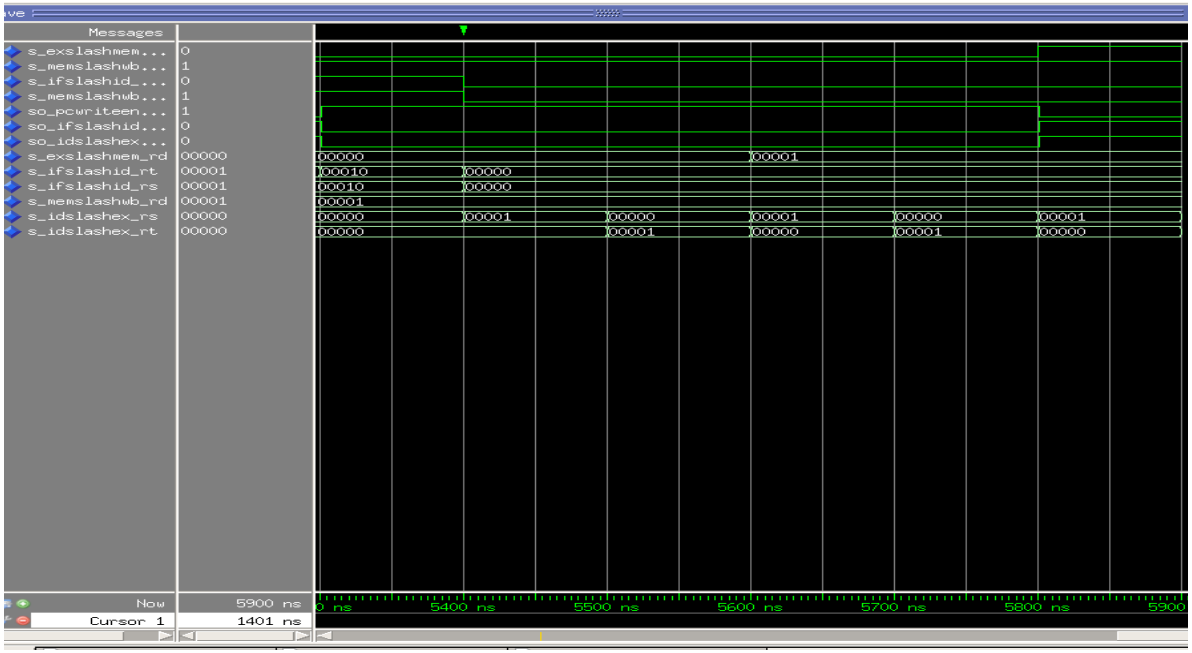




Hazard Detection:

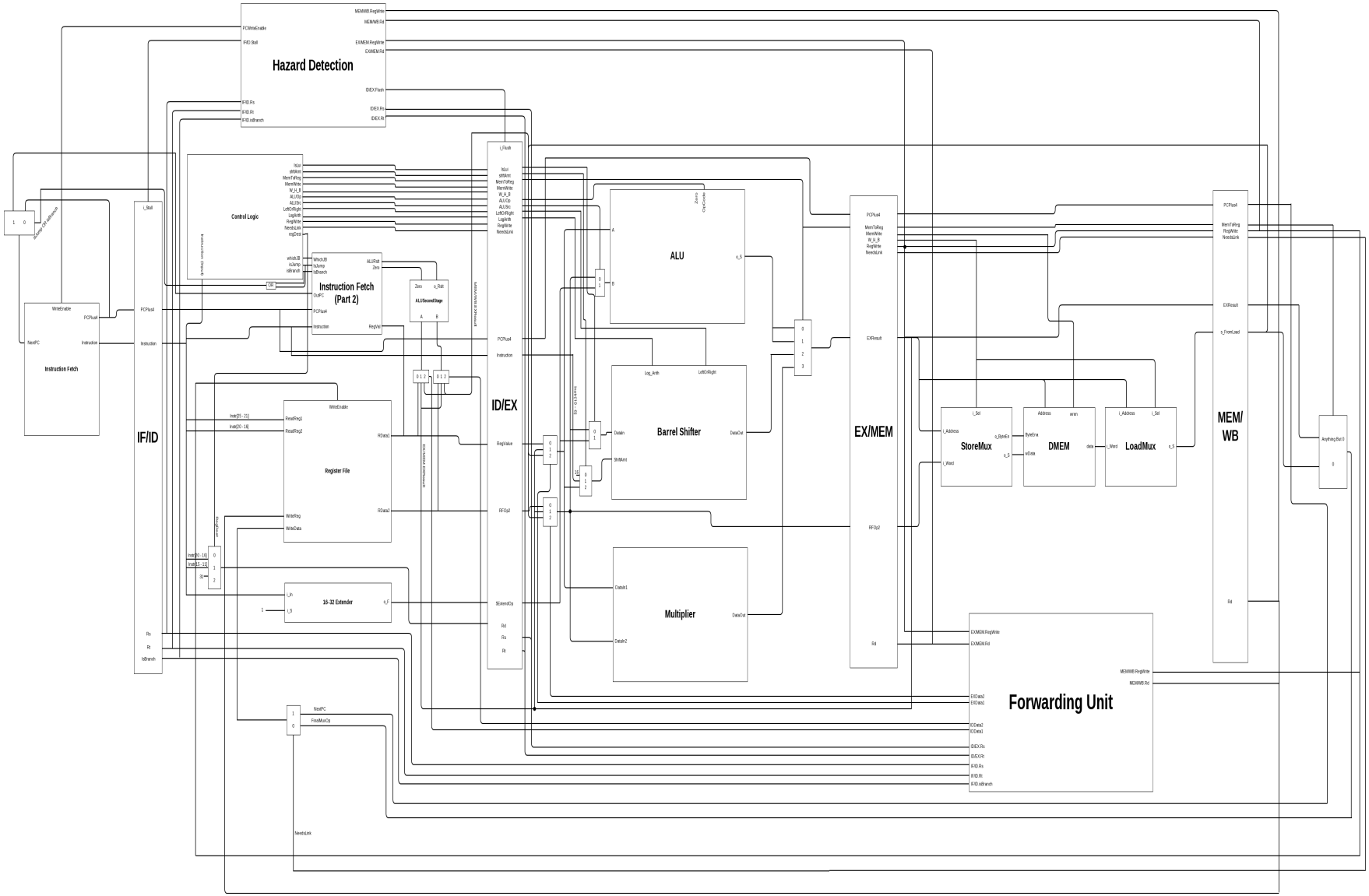

Messages
s_exslashmem...
s_memslashwb...
s_ifslashid_...
s_memslashwb...
so_pcwriteen...
so_ifslashid...
so_idslashex...
s_exslashmem_rd
s_ifslashid_rt
s_ifslashid_rs
s_memslashwb_rd
s_idslashex_rs
s_idslashex_rt
Now
900 ns
Cursor 1
887 ns
400 ns
500 ns
600 ns
700 ns
800 ns


Messages
s_exslashmem...
s_memslashwb...
s_ifslashid_...
s_memslashwb...
so_pcwriteen...
so_ifslashid...
so_idslashex...
s_exslashmem_rd
s_ifslashid_rt
s_ifslashid_rs
s_memslashwb_rd
s_idslashex_rs
s_idslashex_rt
Now
1600 ns
Cursor 1
1401 ns
900 ns
1000 ns
1100 ns
1200 ns
1300 ns



Our Design is as follows below:

CprE 381 – Project Part C
Group 9: Scott Connell, Brian Reber, Arjay Vander Velden



If you can not see our image check out the png in the zip file.

5. **Testing**

a) First we made and tested an application that uses every instruction **without** any data dependencies. It does not have any data dependencies because our code has a lot of no ops Our Code is in the test/asm folder and is called allinstructionsNoDependcies.s .

b) Next we made and tested an application that uses every instruction **with** exhaustively the data dependencies of our processor. Our Code is in the test/asm folder and is called Allinstructions.s.

c) Then we tested with our two sorting algorithms from Project Part B. Also located in the test/asm is the MergeSort.s and the Bubblesort.s.