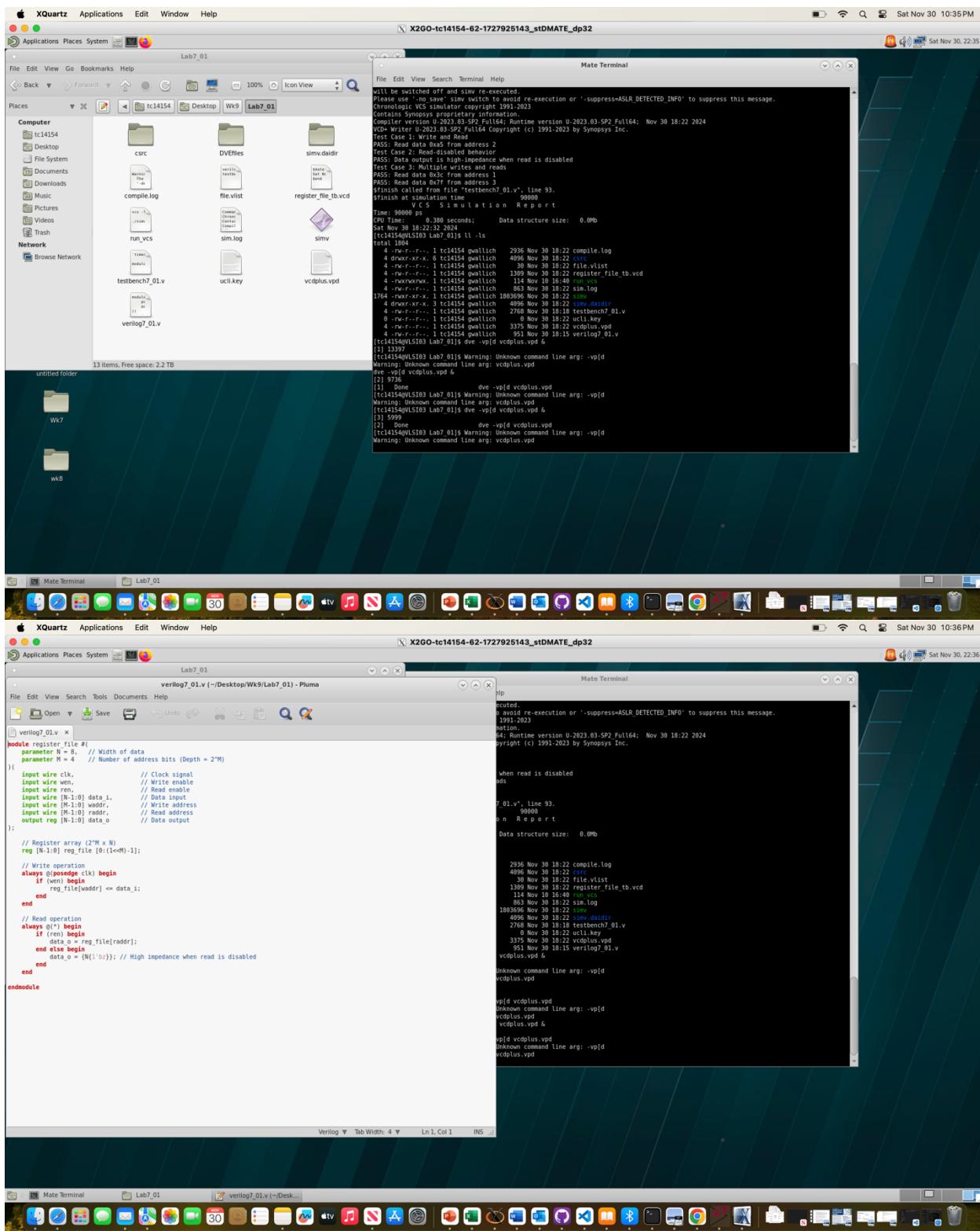


Lab07_01 a and b:

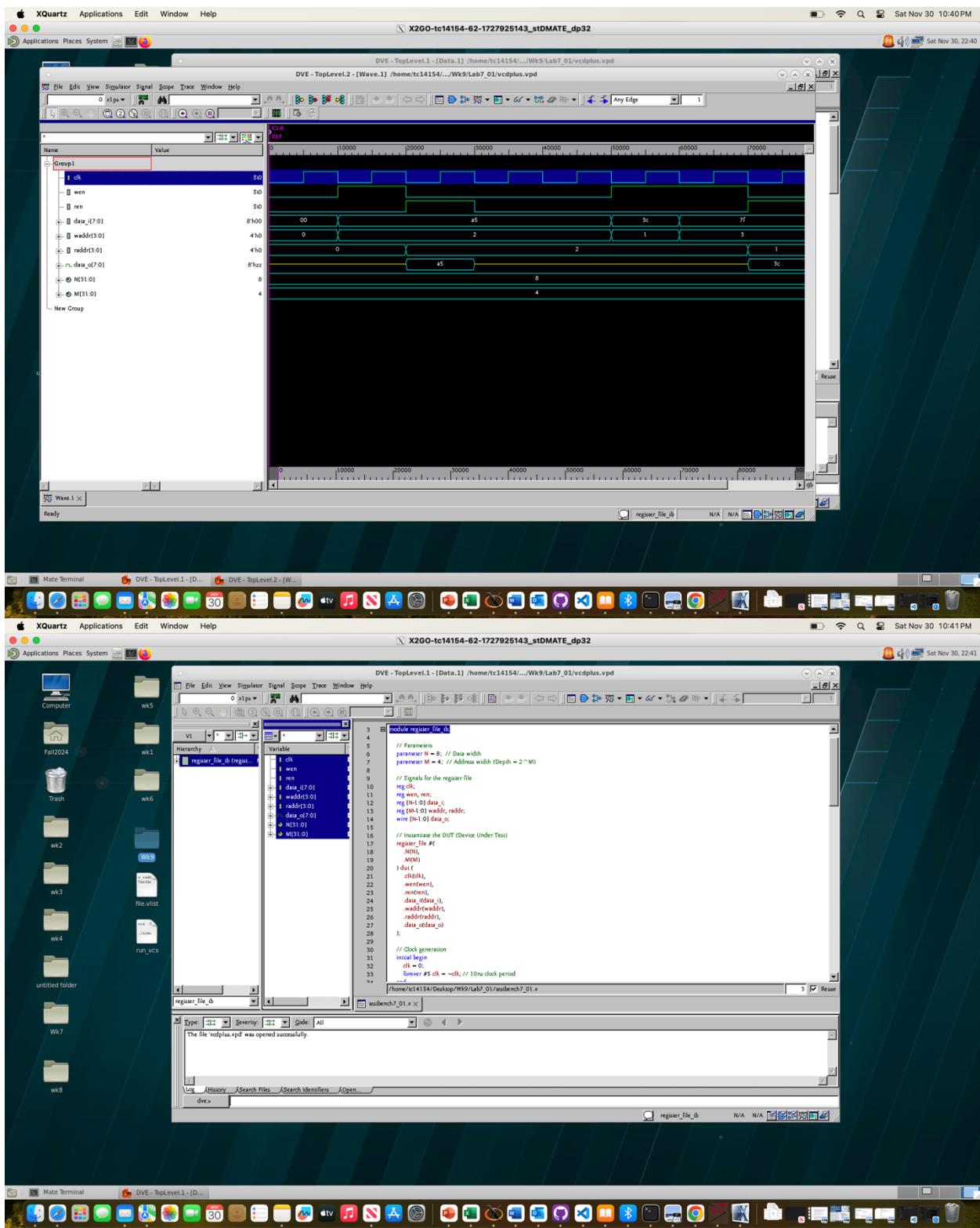


```

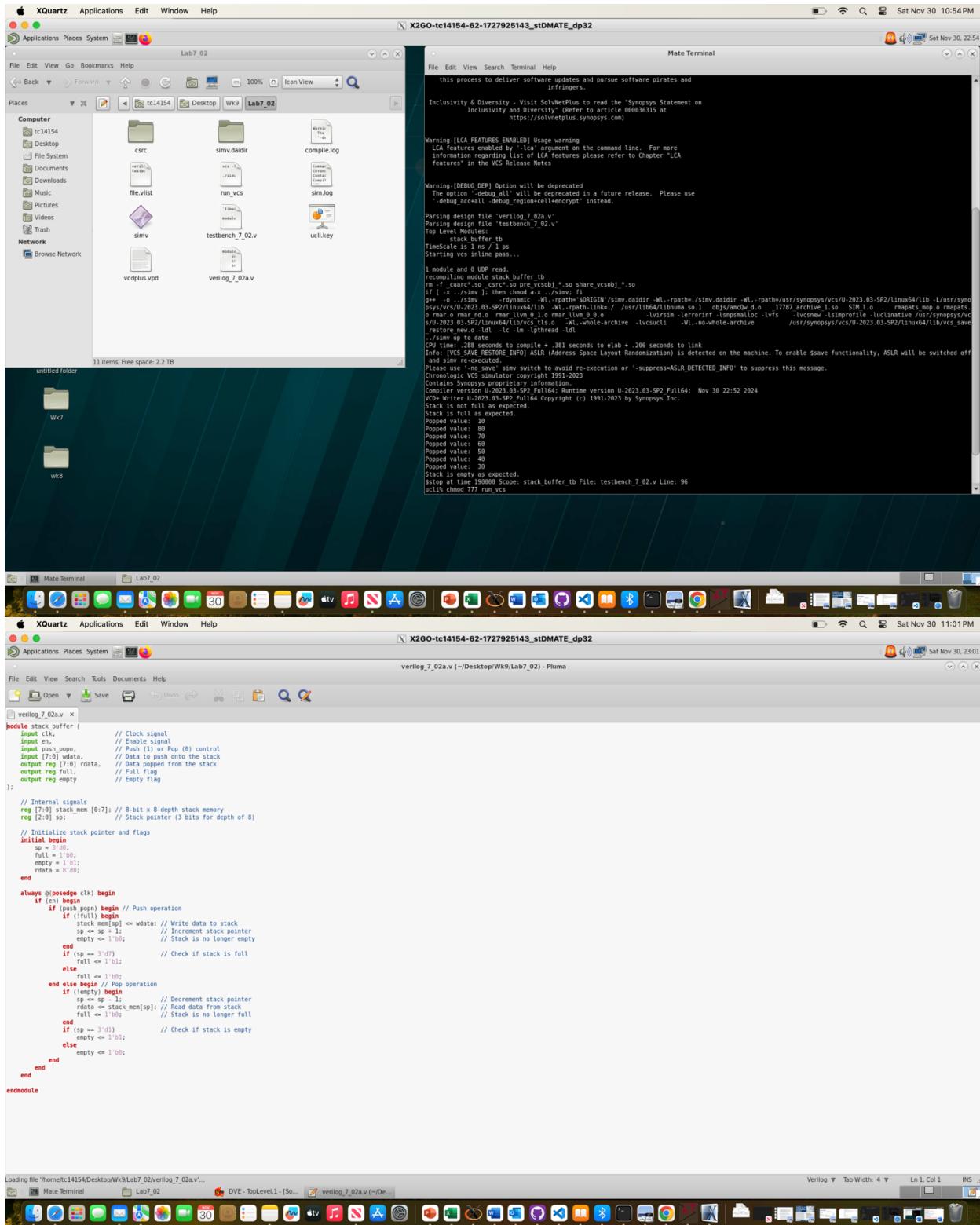
  XQuartz Applications Edit Window Help
  Application Places System testbench7_01.v (~/Desktop/Wk9/Lab7_01) - Pluma
  File Edit View Search Tools Documents Help
  Open Save Undo Redo Find Replace
  testbench7_01.v timescale 1ns/1ps
  module register_file_tb;
  // Parameters
  parameter N = 8; // Data width
  parameter M = 4; // Address width (Depth = 2^M)
  // Signals for the register file
  reg clk;
  reg wen;
  reg [1:0] data_i;
  reg [M-1:0] waddr, raddr;
  wire [N-1:0] data_o;
  // Instantiate the DUT (Device Under Test)
  register_file #(
    .N(N),
    .M(M)
  ) dut (
    .clk(clk),
    .wen(wen),
    .ren(rren),
    .data_i(data_i),
    .waddr(waddr),
    .raddr(raddr),
    .data_o(data_o)
  );
  // Clock generation
  initial begin
    clk = 0;
    forever #5 clk = ~clk; // 10ns clock period
  end
  // Stimulus
  initial begin
    // Initialize signals
    wen = 0; ren = 0;
    data_i = 0; waddr = 0; raddr = 0;
    // Test case 1: Write and Read from the register file
    #10;
    $display("Test Case 1: Write and Read");
    wen = 1; data_i = 8'hAS; waddr = 4'h2; // Write 0xAS to address 2
    #10; wen = 0;
    ren = 1; raddr = 4'h2; // Read from address 2
    #10;
    if (data_o === 8'hAS)
      $display("PASS: Read data 0xAS from address 2", data_o);
    else
      $display("FAIL: Expected 0xAS but got 0x", data_o);
    ren = 0;
    // Test case 2: Check read-disabled behavior
    #10;
    $display("Test Case 2: Read-disabled behavior");
    wen = 1; data_i = 8'hAS; waddr = 4'h2; ren = 1;
    raddr = 4'h0; // High-impedance
    #10;
    if (data_o === 8'hZ)
      $display("PASS: Data output is high-impedance when read is disabled");
    else
      $display("FAIL: Data output should be high-impedance but got 0x", data_o);
    // Test case 3: Write and Read from multiple addresses
    #10;
    $display("Test Case 3: Multiple writes and reads");
    wen = 1; data_i = 8'hABC; waddr = 4'h1; // Write 0xAB to address 1
    #10; data_i = 8'hDEF; waddr = 4'h3; // Write 0xEF to address 3
    #10; wen = 0;
    // Read from address 1
    ren = 1; raddr = 4'h1;
    #10;
    if (data_o === 8'hABC)
      $display("PASS: Read data 0xAB from address 1", data_o);
    else
      $display("FAIL: Expected 0xAB but got 0x", data_o);
    // Read from address 3
    raddr = 4'h3;
    #10;
    if (data_o === 8'hEF)
      $display("PASS: Read data 0xEF from address 3", data_o);
    else
      $display("FAIL: Expected 0xEF but got 0x", data_o);
    ren = 0;
    // End simulation
    $finish;
  end
  // Dump waveform
  initial begin
    $dumpfile("register_file_tb.vcd");
    $dumpvars(0, register_file_tb);
  end
endmodule

```

The screenshot shows two XQuartz windows running on a Mac OS X desktop. Both windows are titled 'testbench7_01.v (~/Desktop/Wk9/Lab7_01) - Pluma'. The top window displays the Verilog code for a register file testbench. The bottom window shows the simulation results, which include several lines of text indicating successful test cases (e.g., 'PASS') and failing ones ('FAIL'). The desktop background features the standard Mac OS X desktop icons.



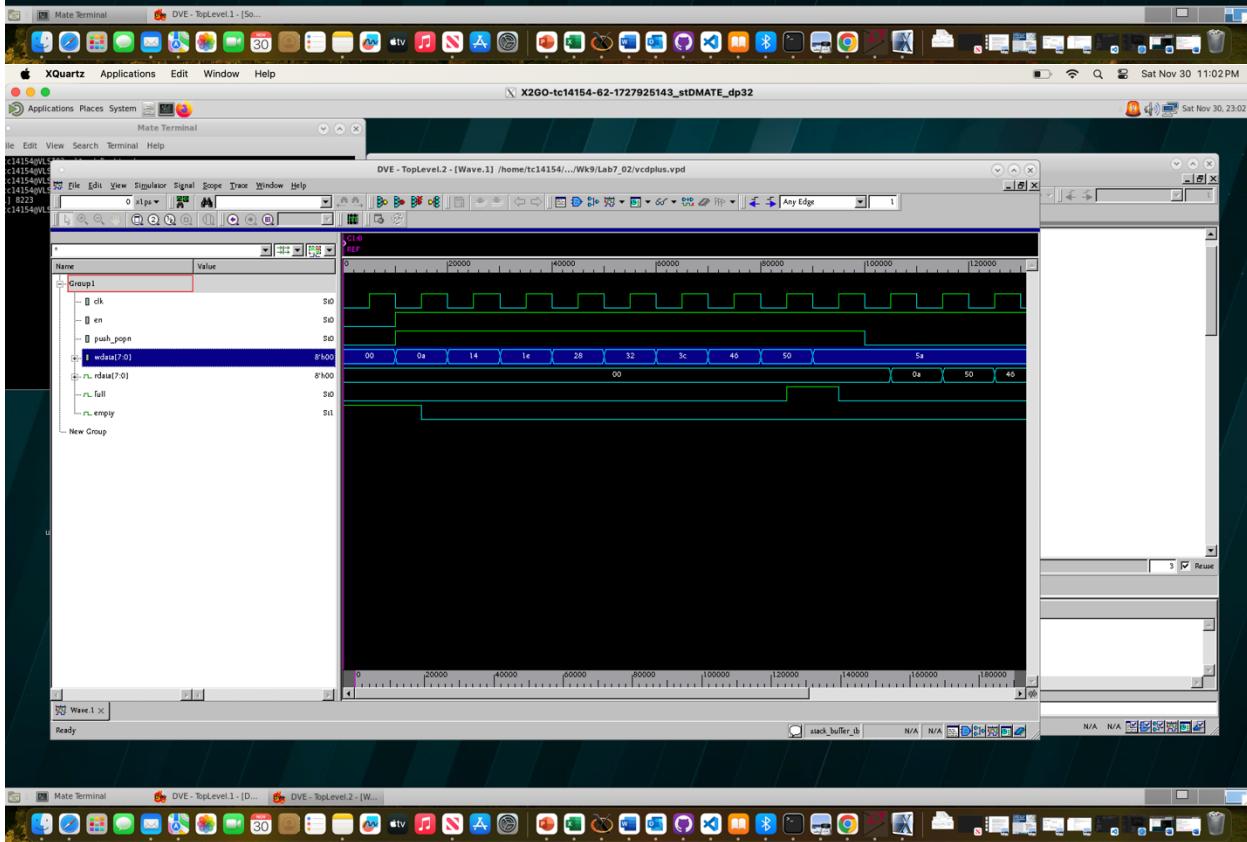
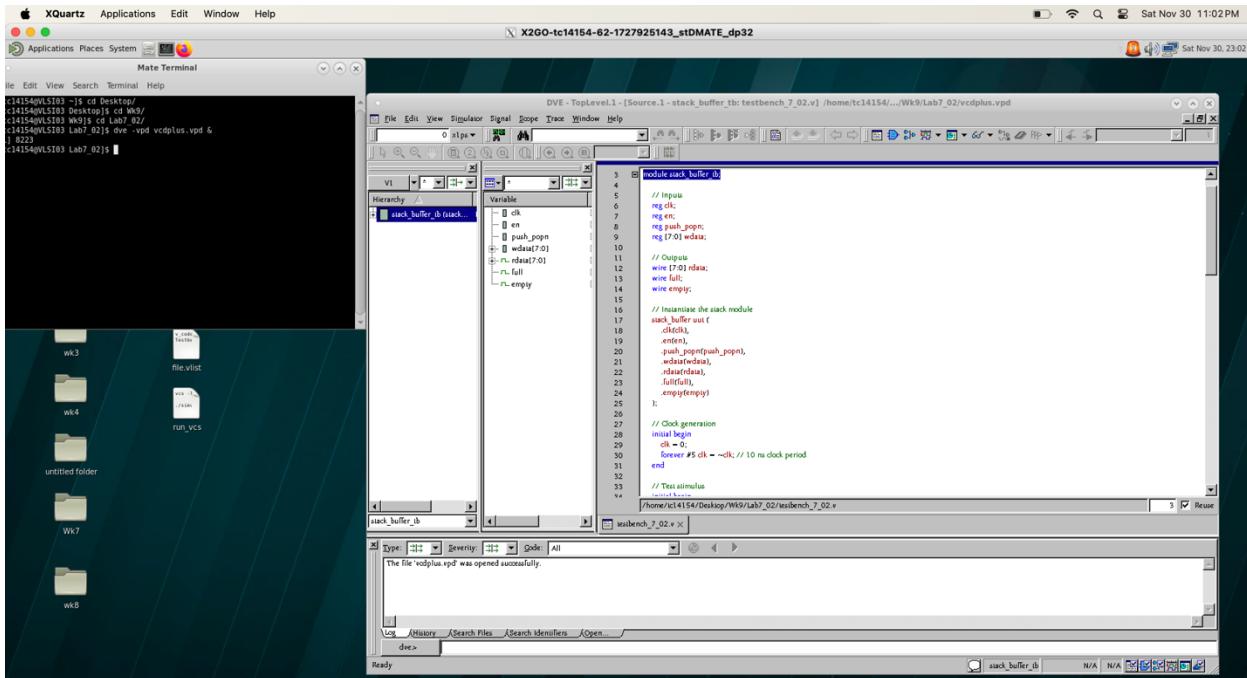
Lab07_02 a and b:



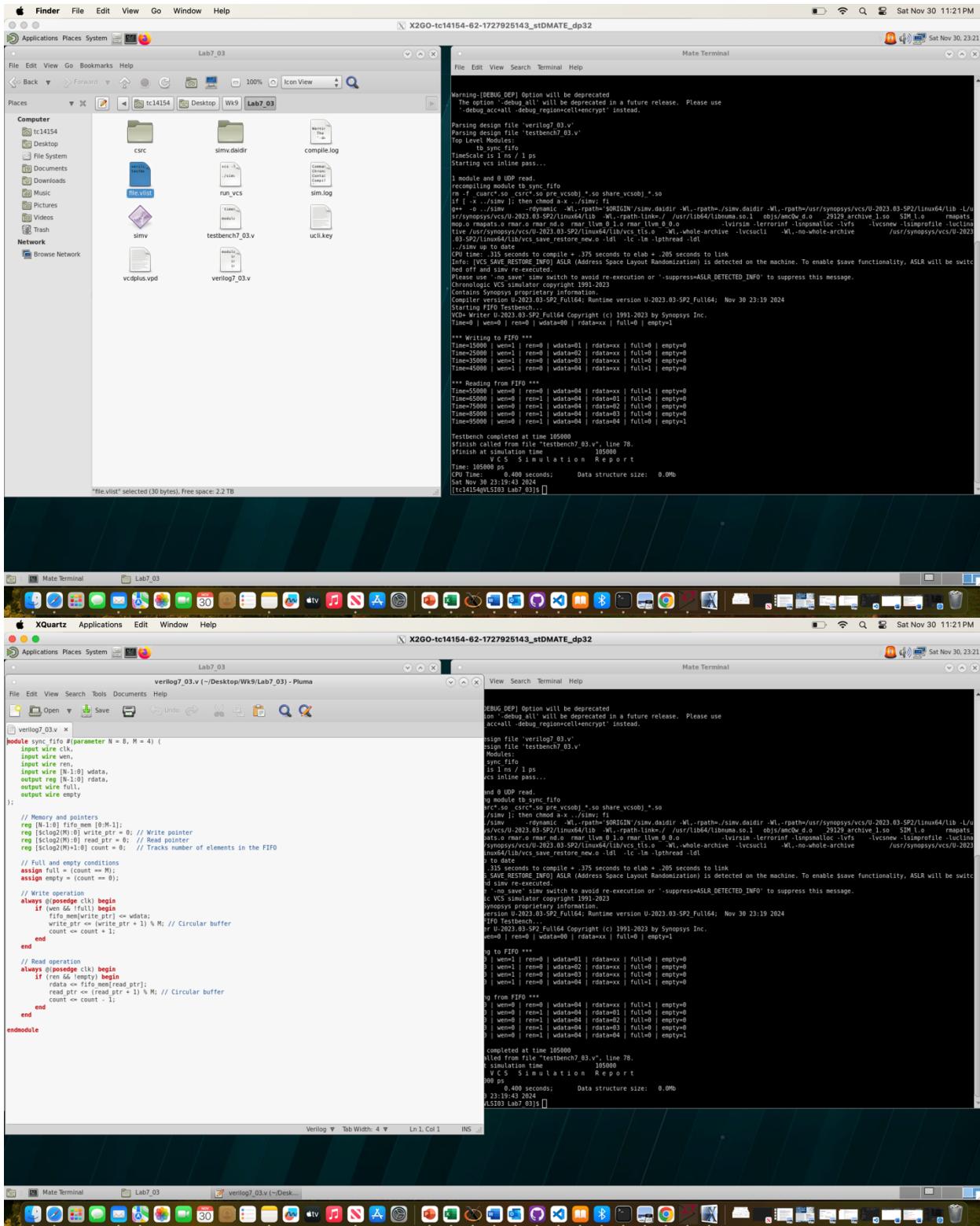
```

  XQuartz Applications Edit Window Help
  X2GO-tc14154-62-1727925143_stDMATE_dp32
  testbench_7_02.v (~/Desktop/Wk9/Lab7_02) - Pluma
  File Edit View Search Tools Documents Help
  Open Save Undo Redo Find Replace Copy Cut Copy All Find Next Find Previous
  testbench_7_02.v x
  timescale 1ns/1ps
  module stack_buffer_tb;
    // Inputs
    reg clk;
    reg en;
    reg push;
    reg [7:0] wdata;
    // Outputs
    wire [7:0] rdata;
    wire full;
    wire empty;
    // Instantiate the stack module
    stack buffer uut (
      .clk(clk),
      .en(en),
      .push(push),
      .wdata(wdata),
      .rdata(rdata),
      .full(full),
      .empty(empty)
    );
    // Clock generation
    initial begin
      clk = 0;
      forever #5 clk = ~clk; // 10 ns clock period
    end
    // Test stimulus
    initial begin
      // Initialize inputs
      en = 0;
      push = 0;
      wdata = 0;
      // Wait for reset
      #10;
      en = 1;
      // Push 4 values onto the stack
      push = 1; // Push mode
      wdata = 0@10; #10;
      wdata = 1@10; #10;
      wdata = 2@10; #10;
      wdata = 3@10; #10;
      // Check full flag (stack is not full yet)
      if (full) $display("Test Failed: Stack shouldn't be full yet");
      else $display("Stack is not full as expected.");
      // Push 4 more values to fill the stack
      wdata = 4@10; #10;
      wdata = 5@10; #10;
      wdata = 6@10; #10;
      wdata = 7@10; #10;
      // Check full flag (stack is not full yet)
      if (full) $display("Test Failed: Stack shouldn't be full yet");
      else $display("Stack is not full as expected.");
      // Push 4 more values to fill the stack
      wdata = 8@10; #10;
      wdata = 9@10; #10;
      wdata = 10@10; #10;
      wdata = 11@10; #10;
      // Check full flag (stack should be full now)
      if (full) $display("Stack is full as expected.");
      else $display("Test Failed: Stack should be full.");
      // Try pushing when the stack is full (shouldn't push)
      wdata = 12@10; #10;
      if (full) $display("No further push allowed when full, as expected.");
      // Pop all values from the stack
      push = 0; // Pop mode
      #10;
      $display("Popped value: %d", rdata);
      $display("Popped value: %d", rdata);
      #10;
      // Check empty flag (stack should now be empty)
      if (empty) $display("Stack is empty as expected.");
      else $display("Test Failed: Stack should be empty.");
      // Try popping when the stack is empty (shouldn't pop)
      #10;
      if (empty) $display("No further pop allowed when empty, as expected.");
      // End of test
      $stop;
    end
    initial
    begin
      sverilogplus;
    end
  endmodule

```



Lab07_03 a and b:



```

  XQuartz Applications Edit Window Help
  Application Places System testbench7_03.v ~/Desktop/Wk9/Lab7_03 - Pluma
  File Edit View Search Tools Documents Help
  Open Save Undo Redo Find Replace Copy Cut Copy All Find Next Find Previous
  timescale 1ns / 1ps
  module tb_sync_fifo;
    // Parameters for FIFO
    parameter N = 8; // Data width
    parameter M = 4; // Depth of the FIFO
    // Testbench signals
    reg clk;
    reg wen;
    reg ren;
    reg [N-1:0] wdata;
    wire [N-1:0] rdata;
    wire full;
    wire empty;
    // Instantiate the FIFO module
    sync_fifo #(N, M) ififo (
      .clk(clk),
      .wen(wen),
      .ren(ren),
      .wdata(wdata),
      .rdata(rdata),
      .full(full),
      .empty(empty)
    );
    // Clock generation
    initial clk = 0;
    always #5 clk = ~clk; // 10 ns clock period
    // Testbench sequence
    initial begin
      $display("Starting FIFO Testbench..."); $monitor("%time@%t | wen=%b | ren=%b | wdata=%h | rdata=%h | full=%b | empty=%b", $time, wen, ren, wdata, rdata, full, empty);
      // Initialize signals
      wen = 0;
      ren = 0;
      wdata = 0;
      // Reset the FIFO
      #10;
      // Write data into the FIFO
      $display("\n*** Writing to FIFO ***");
      repeat (M) begin
        @(posedge clk);
        if (!full) begin
          wen = 1;
          wdata = wdata + 8'h1; // Increment data
        end else begin
          wen = 0;
        end
        $display("FIFO is full at time %0t", $time);
      end
      @(posedge clk);
      wen = 0;
    end
    // Read data from the FIFO
    $display("\n*** Reading from FIFO ***");
    repeat (M) begin
      @(posedge clk);
      if (!empty) begin
        ren = 1;
      end else begin
        ren = 0;
      end
      $display("FIFO is empty at time %0t", $time);
    end
    @(posedge clk);
    ren = 0;
    // End simulation
    $display("\nTestbench completed at time %0t", $time);
    $finish;
  end
  initial begin
    sverilogplus;
  end
endmodule

```

XQuartz Applications Edit Window Help

X2GO-tc14154-62-1727925143_stDMATE_dp2

DVE - TopLevel.1 - [Source.1 - tb_sync_fifo: testbench7_03.v] /home/tc14154/.../Wk

Mate Terminal

```

File Edit View Search Terminal Help
Warning: DEBUG DEP Option will be deprecated
The option --debug_all will be deprecated in a future release. Please use
--debug_all=+regioncell+encrypt instead.

Parsing design file 'verilog.v'
Extracting design file 'testbench7_03.v'
Top Level Modules:
tb_sync_fifo
TimeScale: 1ns / 1 ps
Starting vcs inline pass...

1 module tb_sync_fifo
2   // Parameters for FIFO
3   parameter N = 8; // Data width
4   parameter M = 4; // Depth of the FIFO
5
6   // Testbench signals
7   reg [N-1:0] wdata;
8   reg [M-1:0] rdaddr;
9   reg [N-1:0] rdata;
10  reg [N-1:0] wen;
11  reg [M-1:0] rdreq;
12  reg [N-1:0] rdwdata;
13  reg [M-1:0] rdfull;
14  reg [N-1:0] rdempty;
15  reg [M-1:0] rdvalid;
16
17  // Initialize the FIFO module
18  initial begin
19    $readmem("WIFL_M", wdata);
20    $readmem("RIFL_M", rdaddr);
21    wen=0;
22    rdreq=0;
23    rdvalid=0;
24    rdwdata=0;
25    rdfull=0;
26    rdempty=0;
27  end
28
29  // Clock generation
30  initial begin
31    #5ns #10 ns
32    always #5 ns ~clk; // 10 ns clock period
33  end
34
35  // Testbench sequencer
36
37  initial begin
38    $readmem("WIFL_M", wdata);
39    $readmem("RIFL_M", rdaddr);
40    wen=0;
41    rdreq=0;
42    rdvalid=0;
43    rdwdata=0;
44    rdfull=0;
45    rdempty=0;
46  end
47
48  task write();
49    begin
50      wen=1;
51      rdreq=1;
52      rdvalid=0;
53      rdwdata=0;
54      rdfull=0;
55      rdempty=0;
56      #10 ns;
57      wen=0;
58      rdreq=0;
59      rdvalid=0;
60      rdwdata=0;
61      rdfull=0;
62      rdempty=0;
63    end
64  endtask
65
66  task read();
67    begin
68      rdreq=1;
69      rdvalid=0;
70      rdwdata=0;
71      rdfull=0;
72      rdempty=0;
73      #10 ns;
74      rdreq=0;
75      rdvalid=1;
76      rdwdata=8'h00;
77      rdfull=0;
78      rdempty=0;
79    end
80  endtask
81
82  initial begin
83    $readmem("WIFL_M", wdata);
84    $readmem("RIFL_M", rdaddr);
85    wen=0;
86    rdreq=0;
87    rdvalid=0;
88    rdwdata=0;
89    rdfull=0;
90    rdempty=0;
91  end
92
93  task write();
94    begin
95      wen=1;
96      rdreq=1;
97      rdvalid=0;
98      rdwdata=0;
99      rdfull=0;
100     rdempty=0;
101     #10 ns;
102     wen=0;
103     rdreq=0;
104     rdvalid=0;
105     rdwdata=0;
106     rdfull=0;
107     rdempty=0;
108   end
109 endtask
110
111 task read();
112   begin
113     rdreq=1;
114     rdvalid=0;
115     rdwdata=0;
116     rdfull=0;
117     rdempty=0;
118     #10 ns;
119     rdreq=0;
120     rdvalid=1;
121     rdwdata=8'h00;
122     rdfull=0;
123     rdempty=0;
124   end
125 endtask
126
127 initial begin
128   $readmem("WIFL_M", wdata);
129   $readmem("RIFL_M", rdaddr);
130   wen=0;
131   rdreq=0;
132   rdvalid=0;
133   rdwdata=0;
134   rdfull=0;
135   rdempty=0;
136 end
137
138 task write();
139   begin
140     wen=1;
141     rdreq=1;
142     rdvalid=0;
143     rdwdata=0;
144     rdfull=0;
145     rdempty=0;
146     #10 ns;
147     wen=0;
148     rdreq=0;
149     rdvalid=0;
150     rdwdata=0;
151     rdfull=0;
152     rdempty=0;
153   end
154 endtask
155
156 task read();
157   begin
158     rdreq=1;
159     rdvalid=0;
160     rdwdata=0;
161     rdfull=0;
162     rdempty=0;
163     #10 ns;
164     rdreq=0;
165     rdvalid=1;
166     rdwdata=8'h00;
167     rdfull=0;
168     rdempty=0;
169   end
170 endtask
171
172 initial begin
173   $readmem("WIFL_M", wdata);
174   $readmem("RIFL_M", rdaddr);
175   wen=0;
176   rdreq=0;
177   rdvalid=0;
178   rdwdata=0;
179   rdfull=0;
180   rdempty=0;
181 end
182
183 task write();
184   begin
185     wen=1;
186     rdreq=1;
187     rdvalid=0;
188     rdwdata=0;
189     rdfull=0;
190     rdempty=0;
191     #10 ns;
192     wen=0;
193     rdreq=0;
194     rdvalid=0;
195     rdwdata=0;
196     rdfull=0;
197     rdempty=0;
198   end
199 endtask
200
201 task read();
202   begin
203     rdreq=1;
204     rdvalid=0;
205     rdwdata=0;
206     rdfull=0;
207     rdempty=0;
208     #10 ns;
209     rdreq=0;
210     rdvalid=1;
211     rdwdata=8'h00;
212     rdfull=0;
213     rdempty=0;
214   end
215 endtask
216
217 initial begin
218   $readmem("WIFL_M", wdata);
219   $readmem("RIFL_M", rdaddr);
220   wen=0;
221   rdreq=0;
222   rdvalid=0;
223   rdwdata=0;
224   rdfull=0;
225   rdempty=0;
226 end
227
228 task write();
229   begin
230     wen=1;
231     rdreq=1;
232     rdvalid=0;
233     rdwdata=0;
234     rdfull=0;
235     rdempty=0;
236     #10 ns;
237     wen=0;
238     rdreq=0;
239     rdvalid=0;
240     rdwdata=0;
241     rdfull=0;
242     rdempty=0;
243   end
244 endtask
245
246 task read();
247   begin
248     rdreq=1;
249     rdvalid=0;
250     rdwdata=0;
251     rdfull=0;
252     rdempty=0;
253     #10 ns;
254     rdreq=0;
255     rdvalid=1;
256     rdwdata=8'h00;
257     rdfull=0;
258     rdempty=0;
259   end
260 endtask
261
262 initial begin
263   $readmem("WIFL_M", wdata);
264   $readmem("RIFL_M", rdaddr);
265   wen=0;
266   rdreq=0;
267   rdvalid=0;
268   rdwdata=0;
269   rdfull=0;
270   rdempty=0;
271 end
272
273 task write();
274   begin
275     wen=1;
276     rdreq=1;
277     rdvalid=0;
278     rdwdata=0;
279     rdfull=0;
280     rdempty=0;
281     #10 ns;
282     wen=0;
283     rdreq=0;
284     rdvalid=0;
285     rdwdata=0;
286     rdfull=0;
287     rdempty=0;
288   end
289 endtask
290
291 task read();
292   begin
293     rdreq=1;
294     rdvalid=0;
295     rdwdata=0;
296     rdfull=0;
297     rdempty=0;
298     #10 ns;
299     rdreq=0;
300     rdvalid=1;
301     rdwdata=8'h00;
302     rdfull=0;
303     rdempty=0;
304   end
305 endtask
306
307 initial begin
308   $readmem("WIFL_M", wdata);
309   $readmem("RIFL_M", rdaddr);
310   wen=0;
311   rdreq=0;
312   rdvalid=0;
313   rdwdata=0;
314   rdfull=0;
315   rdempty=0;
316 end
317
318 task write();
319   begin
320     wen=1;
321     rdreq=1;
322     rdvalid=0;
323     rdwdata=0;
324     rdfull=0;
325     rdempty=0;
326     #10 ns;
327     wen=0;
328     rdreq=0;
329     rdvalid=0;
330     rdwdata=0;
331     rdfull=0;
332     rdempty=0;
333   end
334 endtask
335
336 task read();
337   begin
338     rdreq=1;
339     rdvalid=0;
340     rdwdata=0;
341     rdfull=0;
342     rdempty=0;
343     #10 ns;
344     rdreq=0;
345     rdvalid=1;
346     rdwdata=8'h00;
347     rdfull=0;
348     rdempty=0;
349   end
350 endtask
351
352 initial begin
353   $readmem("WIFL_M", wdata);
354   $readmem("RIFL_M", rdaddr);
355   wen=0;
356   rdreq=0;
357   rdvalid=0;
358   rdwdata=0;
359   rdfull=0;
360   rdempty=0;
361 end
362
363 task write();
364   begin
365     wen=1;
366     rdreq=1;
367     rdvalid=0;
368     rdwdata=0;
369     rdfull=0;
370     rdempty=0;
371     #10 ns;
372     wen=0;
373     rdreq=0;
374     rdvalid=0;
375     rdwdata=0;
376     rdfull=0;
377     rdempty=0;
378   end
379 endtask
380
381 task read();
382   begin
383     rdreq=1;
384     rdvalid=0;
385     rdwdata=0;
386     rdfull=0;
387     rdempty=0;
388     #10 ns;
389     rdreq=0;
390     rdvalid=1;
391     rdwdata=8'h00;
392     rdfull=0;
393     rdempty=0;
394   end
395 endtask
396
397 initial begin
398   $readmem("WIFL_M", wdata);
399   $readmem("RIFL_M", rdaddr);
400   wen=0;
401   rdreq=0;
402   rdvalid=0;
403   rdwdata=0;
404   rdfull=0;
405   rdempty=0;
406 end
407
408 task write();
409   begin
410     wen=1;
411     rdreq=1;
412     rdvalid=0;
413     rdwdata=0;
414     rdfull=0;
415     rdempty=0;
416     #10 ns;
417     wen=0;
418     rdreq=0;
419     rdvalid=0;
420     rdwdata=0;
421     rdfull=0;
422     rdempty=0;
423   end
424 endtask
425
426 task read();
427   begin
428     rdreq=1;
429     rdvalid=0;
430     rdwdata=0;
431     rdfull=0;
432     rdempty=0;
433     #10 ns;
434     rdreq=0;
435     rdvalid=1;
436     rdwdata=8'h00;
437     rdfull=0;
438     rdempty=0;
439   end
440 endtask
441
442 initial begin
443   $readmem("WIFL_M", wdata);
444   $readmem("RIFL_M", rdaddr);
445   wen=0;
446   rdreq=0;
447   rdvalid=0;
448   rdwdata=0;
449   rdfull=0;
450   rdempty=0;
451 end
452
453 task write();
454   begin
455     wen=1;
456     rdreq=1;
457     rdvalid=0;
458     rdwdata=0;
459     rdfull=0;
460     rdempty=0;
461     #10 ns;
462     wen=0;
463     rdreq=0;
464     rdvalid=0;
465     rdwdata=0;
466     rdfull=0;
467     rdempty=0;
468   end
469 endtask
470
471 task read();
472   begin
473     rdreq=1;
474     rdvalid=0;
475     rdwdata=0;
476     rdfull=0;
477     rdempty=0;
478     #10 ns;
479     rdreq=0;
480     rdvalid=1;
481     rdwdata=8'h00;
482     rdfull=0;
483     rdempty=0;
484   end
485 endtask
486
487 initial begin
488   $readmem("WIFL_M", wdata);
489   $readmem("RIFL_M", rdaddr);
490   wen=0;
491   rdreq=0;
492   rdvalid=0;
493   rdwdata=0;
494   rdfull=0;
495   rdempty=0;
496 end
497
498 task write();
499   begin
500     wen=1;
501     rdreq=1;
502     rdvalid=0;
503     rdwdata=0;
504     rdfull=0;
505     rdempty=0;
506     #10 ns;
507     wen=0;
508     rdreq=0;
509     rdvalid=0;
510     rdwdata=0;
511     rdfull=0;
512     rdempty=0;
513   end
514 endtask
515
516 task read();
517   begin
518     rdreq=1;
519     rdvalid=0;
520     rdwdata=0;
521     rdfull=0;
522     rdempty=0;
523     #10 ns;
524     rdreq=0;
525     rdvalid=1;
526     rdwdata=8'h00;
527     rdfull=0;
528     rdempty=0;
529   end
530 endtask
531
532 initial begin
533   $readmem("WIFL_M", wdata);
534   $readmem("RIFL_M", rdaddr);
535   wen=0;
536   rdreq=0;
537   rdvalid=0;
538   rdwdata=0;
539   rdfull=0;
540   rdempty=0;
541 end
542
543 task write();
544   begin
545     wen=1;
546     rdreq=1;
547     rdvalid=0;
548     rdwdata=0;
549     rdfull=0;
550     rdempty=0;
551     #10 ns;
552     wen=0;
553     rdreq=0;
554     rdvalid=0;
555     rdwdata=0;
556     rdfull=0;
557     rdempty=0;
558   end
559 endtask
560
561 task read();
562   begin
563     rdreq=1;
564     rdvalid=0;
565     rdwdata=0;
566     rdfull=0;
567     rdempty=0;
568     #10 ns;
569     rdreq=0;
570     rdvalid=1;
571     rdwdata=8'h00;
572     rdfull=0;
573     rdempty=0;
574   end
575 endtask
576
577 initial begin
578   $readmem("WIFL_M", wdata);
579   $readmem("RIFL_M", rdaddr);
580   wen=0;
581   rdreq=0;
582   rdvalid=0;
583   rdwdata=0;
584   rdfull=0;
585   rdempty=0;
586 end
587
588 task write();
589   begin
590     wen=1;
591     rdreq=1;
592     rdvalid=0;
593     rdwdata=0;
594     rdfull=0;
595     rdempty=0;
596     #10 ns;
597     wen=0;
598     rdreq=0;
599     rdvalid=0;
600     rdwdata=0;
601     rdfull=0;
602     rdempty=0;
603   end
604 endtask
605
606 task read();
607   begin
608     rdreq=1;
609     rdvalid=0;
610     rdwdata=0;
611     rdfull=0;
612     rdempty=0;
613     #10 ns;
614     rdreq=0;
615     rdvalid=1;
616     rdwdata=8'h00;
617     rdfull=0;
618     rdempty=0;
619   end
620 endtask
621
622 initial begin
623   $readmem("WIFL_M", wdata);
624   $readmem("RIFL_M", rdaddr);
625   wen=0;
626   rdreq=0;
627   rdvalid=0;
628   rdwdata=0;
629   rdfull=0;
630   rdempty=0;
631 end
632
633 task write();
634   begin
635     wen=1;
636     rdreq=1;
637     rdvalid=0;
638     rdwdata=0;
639     rdfull=0;
640     rdempty=0;
641     #10 ns;
642     wen=0;
643     rdreq=0;
644     rdvalid=0;
645     rdwdata=0;
646     rdfull=0;
647     rdempty=0;
648   end
649 endtask
650
651 task read();
652   begin
653     rdreq=1;
654     rdvalid=0;
655     rdwdata=0;
656     rdfull=0;
657     rdempty=0;
658     #10 ns;
659     rdreq=0;
660     rdvalid=1;
661     rdwdata=8'h00;
662     rdfull=0;
663     rdempty=0;
664   end
665 endtask
666
667 initial begin
668   $readmem("WIFL_M", wdata);
669   $readmem("RIFL_M", rdaddr);
670   wen=0;
671   rdreq=0;
672   rdvalid=0;
673   rdwdata=0;
674   rdfull=0;
675   rdempty=0;
676 end
677
678 task write();
679   begin
680     wen=1;
681     rdreq=1;
682     rdvalid=0;
683     rdwdata=0;
684     rdfull=0;
685     rdempty=0;
686     #10 ns;
687     wen=0;
688     rdreq=0;
689     rdvalid=0;
690     rdwdata=0;
691     rdfull=0;
692     rdempty=0;
693   end
694 endtask
695
696 task read();
697   begin
698     rdreq=1;
699     rdvalid=0;
700     rdwdata=0;
701     rdfull=0;
702     rdempty=0;
703     #10 ns;
704     rdreq=0;
705     rdvalid=1;
706     rdwdata=8'h00;
707     rdfull=0;
708     rdempty=0;
709   end
710 endtask
711
712 initial begin
713   $readmem("WIFL_M", wdata);
714   $readmem("RIFL_M", rdaddr);
715   wen=0;
716   rdreq=0;
717   rdvalid=0;
718   rdwdata=0;
719   rdfull=0;
720   rdempty=0;
721 end
722
723 task write();
724   begin
725     wen=1;
726     rdreq=1;
727     rdvalid=0;
728     rdwdata=0;
729     rdfull=0;
730     rdempty=0;
731     #10 ns;
732     wen=0;
733     rdreq=0;
734     rdvalid=0;
735     rdwdata=0;
736     rdfull=0;
737     rdempty=0;
738   end
739 endtask
740
741 task read();
742   begin
743     rdreq=1;
744     rdvalid=0;
745     rdwdata=0;
746     rdfull=0;
747     rdempty=0;
748     #10 ns;
749     rdreq=0;
750     rdvalid=1;
751     rdwdata=8'h00;
752     rdfull=0;
753     rdempty=0;
754   end
755 endtask
756
757 initial begin
758   $readmem("WIFL_M", wdata);
759   $readmem("RIFL_M", rdaddr);
760   wen=0;
761   rdreq=0;
762   rdvalid=0;
763   rdwdata=0;
764   rdfull=0;
765   rdempty=0;
766 end
767
768 task write();
769   begin
770     wen=1;
771     rdreq=1;
772     rdvalid=0;
773     rdwdata=0;
774     rdfull=0;
775     rdempty=0;
776     #10 ns;
777     wen=0;
778     rdreq=0;
779     rdvalid=0;
780     rdwdata=0;
781     rdfull=0;
782     rdempty=0;
783   end
784 endtask
785
786 task read();
787   begin
788     rdreq=1;
789     rdvalid=0;
790     rdwdata=0;
791     rdfull=0;
792     rdempty=0;
793     #10 ns;
794     rdreq=0;
795     rdvalid=1;
796     rdwdata=8'h00;
797     rdfull=0;
798     rdempty=0;
799   end
800 endtask
801
802 initial begin
803   $readmem("WIFL_M", wdata);
804   $readmem("RIFL_M", rdaddr);
805   wen=0;
806   rdreq=0;
807   rdvalid=0;
808   rdwdata=0;
809   rdfull=0;
810   rdempty=0;
811 end
812
813 task write();
814   begin
815     wen=1;
816     rdreq=1;
817     rdvalid=0;
818     rdwdata=0;
819     rdfull=0;
820     rdempty=0;
821     #10 ns;
822     wen=0;
823     rdreq=0;
824     rdvalid=0;
825     rdwdata=0;
826     rdfull=0;
827     rdempty=0;
828   end
829 endtask
830
831 task read();
832   begin
833     rdreq=1;
834     rdvalid=0;
835     rdwdata=0;
836     rdfull=0;
837     rdempty=0;
838     #10 ns;
839     rdreq=0;
840     rdvalid=1;
841     rdwdata=8'h00;
842     rdfull=0;
843     rdempty=0;
844   end
845 endtask
846
847 initial begin
848   $readmem("WIFL_M", wdata);
849   $readmem("RIFL_M", rdaddr);
850   wen=0;
851   rdreq=0;
852   rdvalid=0;
853   rdwdata=0;
854   rdfull=0;
855   rdempty=0;
856 end
857
858 task write();
859   begin
860     wen=1;
861     rdreq=1;
862     rdvalid=0;
863     rdwdata=0;
864     rdfull=0;
865     rdempty=0;
866     #10 ns;
867     wen=0;
868     rdreq=0;
869     rdvalid=0;
870     rdwdata=0;
871     rdfull=0;
872     rdempty=0;
873   end
874 endtask
875
876 task read();
877   begin
878     rdreq=1;
879     rdvalid=0;
880     rdwdata=0;
881     rdfull=0;
882     rdempty=0;
883     #10 ns;
884     rdreq=0;
885     rdvalid=1;
886     rdwdata=8'h00;
887     rdfull=0;
888     rdempty=0;
889   end
890 endtask
891
892 initial begin
893   $readmem("WIFL_M", wdata);
894   $readmem("RIFL_M", rdaddr);
895   wen=0;
896   rdreq=0;
897   rdvalid=0;
898   rdwdata=0;
899   rdfull=0;
900   rdempty=0;
901 end
902
903 task write();
904   begin
905     wen=1;
906     rdreq=1;
907     rdvalid=0;
908     rdwdata=0;
909     rdfull=0;
910     rdempty=0;
911     #10 ns;
912     wen=0;
913     rdreq=0;
914     rdvalid=0;
915     rdwdata=0;
916     rdfull=0;
917     rdempty=0;
918   end
919 endtask
920
921 task read();
922   begin
923     rdreq=1;
924     rdvalid=0;
925     rdwdata=0;
926     rdfull=0;
927     rdempty=0;
928     #10 ns;
929     rdreq=0;
930     rdvalid=1;
931     rdwdata=8'h00;
932     rdfull=0;
933     rdempty=0;
934   end
935 endtask
936
937 initial begin
938   $readmem("WIFL_M", wdata);
939   $readmem("RIFL_M", rdaddr);
940   wen=0;
941   rdreq=0;
942   rdvalid=0;
943   rdwdata=0;
944   rdfull=0;
945   rdempty=0;
946 end
947
948 task write();
949   begin
950     wen=1;
951     rdreq=1;
952     rdvalid=0;
953     rdwdata=0;
954     rdfull=0;
955     rdempty=0;
956     #10 ns;
957     wen=0;
958     rdreq=0;
959     rdvalid=0;
960     rdwdata=0;
961     rdfull=0;
962     rdempty=0;
963   end
964 endtask
965
966 task read();
967   begin
968     rdreq=1;
969     rdvalid=0;
970     rdwdata=0;
971     rdfull=0;
972     rdempty=0;
973     #10 ns;
974     rdreq=0;
975     rdvalid=1;
976     rdwdata=8'h00;
977     rdfull=0;
978     rdempty=0;
979   end
980 endtask
981
982 initial begin
983   $readmem("WIFL_M", wdata);
984   $readmem("RIFL_M", rdaddr);
985   wen=0;
986   rdreq=0;
987   rdvalid=0;
988   rdwdata=0;
989   rdfull=0;
990   rdempty=0;
991 end
992
993 task write();
994   begin
995     wen=1;
996     rdreq=1;
997     rdvalid=0;
998     rdwdata=0;
999     rdfull=0;
1000    rdempty=0;
1001    #10 ns;
1002    wen=0;
1003    rdreq=0;
1004    rdvalid=0;
1005    rdwdata=0;
1006    rdfull=0;
1007    rdempty=0;
1008  end
1009 endtask
1010
1011 task read();
1012  begin
1013    rdreq=1;
1014    rdvalid=0;
1015    rdwdata=0;
1016    rdfull=0;
1017    rdempty=0;
1018    #10 ns;
1019    rdreq=0;
1020    rdvalid=1;
1021    rdwdata=8'h00;
1022    rdfull=0;
1023    rdempty=0;
1024  end
1025 endtask
1026
1027 initial begin
1028   $readmem("WIFL_M", wdata);
1029   $readmem("RIFL_M", rdaddr);
1030   wen=0;
1031   rdreq=0;
1032   rdvalid=0;
1033   rdwdata=0;
1034   rdfull=0;
1035   rdempty=0;
1036 end
1037
1038 task write();
1039  begin
1040    wen=1;
1041    rdreq=1;
1042    rdvalid=0;
1043    rdwdata=0;
1044    rdfull=0;
1045    rdempty=0;
1046    #10 ns;
1047    wen=0;
1048    rdreq=0;
1049    rdvalid=0;
1050    rdwdata=0;
1051    rdfull=0;
1052    rdempty=0;
1053  end
1054 endtask
1055
1056 task read();
1057  begin
1058    rdreq=1;
1059    rdvalid=0;
1060    rdwdata=0;
1061    rdfull=0;
1062    rdempty=0;
1063    #10 ns;
1064    rdreq=0;
1065    rdvalid=1;
1066    rdwdata=8'h00;
1067    rdfull=0;
1068    rdempty=0;
1069  end
1070 endtask
1071
1072 initial begin
1073   $readmem("WIFL_M", wdata);
1074   $readmem("RIFL_M", rdaddr);
1075   wen=0;
1076   rdreq=0;
1077   rdvalid=0;
1078   rdwdata=0;
1079   rdfull=0;
1080   rdempty=0;
1081 end
1082
1083 task write();
1084  begin
1085    wen=1;
1086    rdreq=1;
1087    rdvalid=0;
1088    rdwdata=0;
1089    rdfull=0;
1090    rdempty=0;
1091    #10 ns;
1092    wen=0;
1093    rdreq=0;
1094    rdvalid=0;
1095    rdwdata=0;
1096    rdfull=0;
1097    rdempty=0;
1098  end
1099 endtask
1100
1101 task read();
1102  begin
1103    rdreq=1;
1104    rdvalid=0;
1105    rdwdata=0;
1106    rdfull=0;
1107    rdempty=0;
1108    #10 ns;
1109    rdreq=0;
1110    rdvalid=1;
1111    rdwdata=8'h00;
1112    rdfull=0;
1113    rdempty=0;
1114  end
1115 endtask
1116
1117 initial begin
1118   $readmem("WIFL_M", wdata);
1119   $readmem("RIFL_M", rdaddr);
1120   wen=0;
1121   rdreq=0;
1122   rdvalid=0;
1123   rdwdata=0;
1124   rdfull=0;
1125   rdempty=0;
1126 end
1127
1128 task write();
1129  begin
1130    wen=1;
1131    rdreq=1;
1132    rdvalid=0;
1133    rdwdata=0;
1134    rdfull=0;
1135    rdempty=0;
1136    #10 ns;
1137    wen=0;
1138    rdreq=0;
1139    rdvalid=0;
1140    rdwdata=0;
1141    rdfull=0;
1142    rdempty=0;
1143  end
1144 endtask
1145
1146 task read();
1147  begin
1148    rdreq=1;
1149    rdvalid=0;
1150    rdwdata=0;
1151    rdfull=0;
1152    rdempty=0;
1153    #10 ns;
1154    rdreq=0;
1155    rdvalid=1;
1156    rdwdata=8'h00;
1157    rdfull=0;
1158    rdempty=0;
1159  end
1160 endtask
1161
1162 initial begin
1163   $readmem("WIFL_M", wdata);
1164   $readmem("RIFL_M", rdaddr);
1165   wen=0;
1166   rdreq=0;
1167   rdvalid=0;
1168   rdwdata=0;
1169   rdfull=0;
1170   rdempty=0;
1171 end
1172
1173 task write();
1174  begin
1175    wen=1;
1176    rdreq=1;
1177    rdvalid=0;
1178    rdwdata=0;
1179    rdfull=0;
1180    rdempty=0;
1181    #10 ns;
1182    wen=0;
1183    rdreq=0;
1184    rdvalid=0;
1185    rdwdata=0;
1186    rdfull=0;
1187    rdempty=0;
1188  end
1189 endtask
1190
1191 task read();
1192  begin
1193    rdreq=1;
1194    rdvalid=0;
1195    rdwdata=0;
1196    rdfull=0;
1197    rdempty=0;
1198    #10 ns;
1199    rdreq=0;
1200    rdvalid=1;
1201    rdwdata=8'h00;
1202    rdfull=0;
1203    rdempty=0;
1204  end
1205 endtask
1206
1207 initial begin
1208   $readmem("WIFL_M", wdata);
1209   $readmem("RIFL_M", rdaddr);
1210   wen=0;
1211   rdreq=0;
1212   rdvalid=0;
1213   rdwdata=0;
1214   rdfull=0;
1215   rdempty=0;
1216 end
1217
1218 task write();
1219  begin
1220    wen=1;
1221    rdreq=1;
1222    rdvalid=0;
1223    rdwdata=0;
1224    rdfull=0;
1225    rdempty=0;
1226    #10 ns;
1227    wen=0;
1228    rdreq=0;
1229    rdvalid=0;
1230    rdwdata=0;
1231    rdfull=0;
1232    rdempty=0;
1233  end
1234 endtask
1235
1236 task read();
1237  begin
1238    rdreq=1;
1239    rdvalid=0;
1240    rdwdata=0;
1241    rdfull=0;
1242    rdempty=0;
1243    #10 ns;
1244    rdreq=0;
1245    rdvalid=1;
1246    rdwdata=8'h00;
1247    rdfull=0;
1248    rdempty=0;
1249  end
1250 endtask
1251
1252 initial begin
1253   $readmem("WIFL_M", wdata);
1254   $readmem("RIFL_M", rdaddr);
1255   wen=0;
1256   rdreq=0;
1257   rdvalid=0;
1258   rdwdata=0;
1259   rdfull=0;
1260   rdempty=0;
1261 end
1262
1263 task write();
1264  begin
1265    wen=1;
1266    rdreq=1;
1267    rdvalid=0;
1268    rdwdata=0;
1269    rdfull=0;
1270    rdempty=0;
1271    #10 ns;
1272    wen=0;
1273    rdreq=0;
1274    rdvalid=0;
1275    rdwdata=0;
1276    rdfull=0;
1277    rdempty=0;
1278  end
1279 endtask
1280
1281 task read();
1282  begin
1283    rdreq=1;
1284    rdvalid=0;
1285    rdwdata=0;
1286    rdfull=0;
1287    rdempty=0;
1288    #10 ns;
1289    rdreq=0;
1290    rdvalid=1;
1291    rdwdata=8'h00;
1292    rdfull=0;
1293    rdempty=0;
1294  end
1295 endtask
1296
1297 initial begin
1298   $readmem
```