

```
1  -- May Trinh, Payton Fisher, Danny Woodard
2  -- Logic Design
3  -- Mini Processor Project
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7  use ieee.numeric_std.all;
8
9  entity main is
10     port(
11
12         upd, exe: in std_logic; -- 2 clocks
13         instruction : in std_logic_vector(15 downto 0);
14         out1: out std_logic_vector(6 downto 0);
15         out2: out std_logic_vector (6 downto 0)
16     );
17
18 end entity;
19
20 -- architecture for main entity
21
22 architecture rtl of main is
23
24     -- component to decode the instruction
25     component Decoder is
26     port(
27         instruction : in std_logic_vector (15 downto 0);
28         -- upd      : in std_logic; -- update destination register
29         -- exe       : in std_logic; -- execute the operation
30         -- exe out    : out std_logic;
31         -- upd_out    : out std_logic;
32         desReg       : out std_logic;
33         chosenInstr  : out std_logic_vector(3 downto 0);
34         isSub        : out std_logic;
35         isConst      : out std_logic
36     );
37
38 end component;
39
40 component registerEnabler is
41
42     port
43     (
44         desReg       : in std_logic;
45         upd          : in std_logic;
46         enableR0     : out std_logic;
47         enableR1     : out std_logic
48     );
49 end component;
50
51
52 -- register entity
53 component reg is
54     port(
55         rin: in std_logic_vector(7 downto 0);--input register
56         enable: in std_logic;
57         rout : out std_logic_vector (7 downto 0)
58     );
59 end component;
60
61
62 -- multiplexor for register
63 component regMux is
64
65     port
66     (
67         r0: in std_logic_vector (7 downto 0);
68         r1: in std_logic_vector (7 downto 0);
```

```
69         s: in std_logic;
70         y: out std_logic_vector (7 downto 0)
71     );
72
73 end component;
74
75 -- multiplexor for operator
76
77 component mux is
78     port (
79         instr    : in std_logic_vector (3 downto 0);
80         addorsub  : in std_logic_vector (7 downto 0);
81         sub       : in std_logic_vector (7 downto 0);
82         isConst: in std_logic;
83         addConst: in std_logic_vector (7 downto 0);
84         subConst: in std_logic_vector (7 downto 0);
85         xand      : in std_logic_vector (7 downto 0);
86         mov       : in std_logic_vector (7 downto 0);
87         neg       : in std_logic_vector (7 downto 0);
88         Eor       : in std_logic_vector (7 downto 0);
89         shl       : in std_logic_vector (7 downto 0);
90         shr       : in std_logic_vector (7 downto 0);
91         to reg    : out std_logic_vector (7 downto 0)
92     );
93 end component;
94
95 component mov is
96     port(
97         num: in std_logic_vector (7 downto 0);
98         rd : out std_logic_vector (7 downto 0)
99     );
100 end component;
101
102 component display is
103     port(
104         rs: in std_logic_vector ( 7 downto 0);
105         a_seg: out std_logic_vector (6 downto 0);
106         b_seg : out std_logic_vector (6 downto 0)
107     );
108 end component;
109
110 component add is
111     port (a,b: in std_logic_vector(7 downto 0);
112         isSub : in std_logic;
113         sum: out std_logic_vector (7 downto 0)
114     );
115 end component;
116
117 component subtract is
118     port (a,b: in std_logic_vector(7 downto 0);
119         isSub : in std_logic;
120         sum: out std_logic_vector (7 downto 0)
121     );
122 end component;
123
124 component getConst is
125     port (
126         a: in std_logic_vector(2 downto 0);
127         isConst : in std_logic;
128         new val: out std_logic_vector(7 downto 0)
129     );
130 end component;
131
132 component andReg is
133     port (
134         r0,r1: in std_logic_vector (7 downto 0);
135         rd: out std_logic_vector (7 downto 0)
136     );
```

```

137     end component;
138
139
140
141     -- neg entity - complement Rs, stores in Rd
142     component neg is
143     port (
144         rs: in std_logic_vector (7 downto 0);
145         rd: out std_logic_vector (7 downto 0)
146     );
147     end component;
148
149     -- eor entity-- xor Rs and Rd, stores in Rd
150     component eor is
151     port(
152         r0, r1 : in std_logic_vector(7 downto 0);
153         rd: out std_logic_vector(7 downto 0)
154     );
155     end component;
156
157
158     -----
159     -- lsl entity - left shift Rs by 1 bit, stores in Rd
160     component lsl is
161     port (
162         rs: in std_logic_vector (7 downto 0);
163         rd: out std_logic_vector (7 downto 0)
164     );
165     end component;
166
167
168     component lsr is
169     port (
170         rs: in std_logic_vector (7 downto 0);
171         rd: out std_logic_vector (7 downto 0)
172     );
173     end component;
174
175
176     signal v: std_logic_vector(7 downto 0);
177     signal rs: std_logic_vector (0 downto 0);
178     signal reg0, reg1, reg3: std_logic_vector(7 downto 0);
179     signal rs1, rs2, rs3: std_logic_vector(7 downto 0);
180
181     -- for multiplexor instruction
182     signal addSubAns, constAddAns, subAns,subConstAns, andAns, movAns, negAns, eorAns, shlAns,
    shrAns, finalAns : std_logic_vector (7 downto 0);
183     signal chosenRegv, displayRegv: std_logic_vector ( 7 downto 0);
184     signal r0_new, r1_new :std_logic_vector (7 downto 0);
185
186     --decoder's needs
187     signal exe_outM, upd_outM, wRegM, enableAL_M, enableBL_M, isSubM, isConstM: std_logic;
188     signal chosenInstrM: std_logic_vector (3 downto 0);
189
190     --for register
191     signal r0 outM, r1 outM, tempOut: std_logic_vector(7 downto 0);
192     signal a1,b1 : std_logic;
193
194     begin
195
196
197         --D: Decoder port map(instruction, (not upd), (not exe), exe outM, upd outM, wRegM,
    chosenInstrM, isSubM, isConstM);
198         D: Decoder port map (instruction, wRegM, chosenInstrM, isSubM, isConstM);
199         --regE: registerEnabler port map(wRegM, upd outM, enableAL M, enableBL M);
200
201         -- operator move
202         a: mov port map (instruction(7 downto 0), movAns);

```

```

203
204     wReg1: regMux port map(r0 outM, r1 outM, instruction(6), reg0);
205     wReg2: regMux port map(r0 outM, r1 outM, instruction(3), reg1);
206
207     addReg: add port map (reg0, reg1, isSubM, addSubAns);
208     sub: subtract port map (reg0, reg1, isSubM, subAns);
209
210     const: getConst port map(instruction(8 downto 6), isConstM, r0 new);
211
212     -- add/subtract constant
213     addConst: add port map (r0 new, reg1, isSubM, constAddAns);
214     subConst: subtract port map (reg1, r0 new, isSubM, subConstAns);
215
216
217     wReg3: regMux port map(r0 outM, r1 outM, instruction(0), reg3);
218     -- eor operator
219     xorReg: eor port map (reg1, reg3, eorAns);
220
221     -- and operator
222     andRegister: andReg port map (reg1, reg3, andAns);
223
224     -- multiplexor to decide the register for operation
225     wRegPass: regMux port map(r0 outM, r1 outM, instruction(3), chosenRegv);
226
227     -- neg operator
228     negate: neg port map (chosenRegv, negAns);
229
230     -- shift_left operator
231     shl: lsl port map (chosenRegv, shlAns);
232
233     -- shift_right operator
234     shr: lsr port map (chosenRegv, shrAns);
235
236     -- answers passed through a multiplexor to determine the instruction
237     multiplexor: mux port map (chosenInstrM, addSubAns, subAns, isConstM, constAddAns,
subConstAns, andAns, movAns, negAns, eorAns, shlAns, shrAns, finalAns);
238
239
240     regE: registerEnabler port map(wRegM, upd, enableAL M, enableBL M);
241
242     -- temp register to store the value and passes it to R0 or R1 when necessary
243     temp: reg port map (finalAns, (not exe), tempOut);
244     r0: reg port map (tempOut, enableAL_M, r0_outM);
245     r1: reg port map (tempOut, enableBL M, r1_outM);
246
247
248     displayReg: regMux port map (r0 outM, r1 outM, wRegM, displayRegv);
249     displ: display port map (displayRegv, out1, out2);
250
251 end;
252
253
254 library ieee;
255 use ieee.std_logic_1164.all;
256 use ieee.numeric_std.all;
257
258 -----
259 entity Decoder is
260     port(
261         instruction : in std_logic_vector (15 downto 0);
262         -- upd       : in std_logic; -- update destination register
263         -- exe       : in std_logic; -- execute the operation
264         -- exe_out    : out std_logic;
265         -- upd_out    : out std_logic;
266         desReg       : out std_logic;
267         chosenInstr  : out std_logic_vector(3 downto 0);
268         isSub        : out std_logic;
269         isConst      : out std_logic

```

```
270     );
271
272 end entity;
273
274 architecture rtl of Decoder is
275
276 signal firstFour: std_logic_vector (3 downto 0);
277 signal reg: std_logic_vector (2 downto 0);
278 signal rd: std_logic;
279 signal instrC: std_logic_vector (3 downto 0);
280 signal updd, exed: std_logic;
281 begin
282
283     firstFour <= instruction(15 downto 12);
284
285     process(instruction) is
286     begin
287
288         if (firstFour = "0010") then -- move instruction
289             reg <= instruction(10 downto 8);
290
291         else
292             reg <= instruction(2 downto 0);-- destination register
293         end if;
294
295         if(reg = "000") then
296             rd <= '0'; -- destination register is R0
297         else
298             rd <= '1'; -- destination register is R1
299         end if;
300     end process;
301
302     process(instruction) is
303     begin
304
305         --add instruction
306         if(instruction (15 downto 9 ) = "0001100") then
307             instrC <= "0000";
308             isSub <= '0';
309             isConst <= '0';
310
311         -- add constant
312         elsif (instruction (15 downto 9) = "0001110") then
313             instrC <= "1001";
314             isSub <= '0';
315             isConst <= '1';
316
317         -- and instruction
318         elsif ((instruction (15 downto 6) = "010000000")) then
319             instrC <= "0001";
320
321         -- mov instruction
322         elsif((instruction (15 downto 11) = "00100")) then
323             instrC <= "0010";
324
325         -- negate instruction
326         elsif((instruction (15 downto 6) = "0100001001")) then
327             instrC <= "0011";
328
329         -- xor instruction
330         elsif((instruction (15 downto 6)= "0100000001")) then
331             instrC <= "0100";
332
333         -- shift left instruction
334         elsif((instruction (15 downto 6) = "0000000001")) then
335             instrC <= "0101";
336
337         -- shift right instruction
```

```

338         elsif((instruction (15 downto 6) = "0000100001")) then
339             instrC <= "0110";
340
341             -- subtract instruction
342         elsif(instruction (15 downto 9) = "0001101") then
343             isSub <= '1';
344             instrC <= "0111";
345             isConst <='0';
346
347             -- subtract constant
348         elsif(instruction (15 downto 9) = "0001111") then
349             isSub <= '1';
350             instrC <= "1000";
351             isConst <= '1';
352         end if;
353
354     end process;
355
356     -- process(upd, exe) is
357     -- begin
358     --
359     --     if(upd = '0' and exe = '0') then -- both clock cannot be ON, -> set both to OFF
360     --         updd <= '1';
361     --         exed <= '1';
362     --     else
363     --         updd <= upd;
364     --         exed <= exe;
365     --     end if;
366     -- end process;
367     --
368     -- exe_out <= exed;
369     -- upd_out <= updd;
370     desReg <= rd; -- destination register
371     chosenInstr <= instrC; -- chosen instruction
372 end rtl;
373
374
375 -----
376 library ieee;
377 use ieee.std_logic_1164.all;
378 use ieee.numeric_std.all;
379 -----
380 -- register entity
381 entity reg is
382     port(
383         rin: in std_logic_vector(7 downto 0);--input register
384         enable: in std_logic;
385         rout : out std_logic_vector (7 downto 0)
386     );
387 end entity;
388
389
390 architecture r of reg is
391     signal rt: std_logic_vector (7 downto 0);
392 begin
393     process (enable) is
394     begin
395
396         if (enable = '1') then
397             rout <= rin;
398         end if;
399
400     end process;
401 end;
402
403 -----
404 library ieee;
405 use ieee.std_logic_1164.all;

```

```
406
407 entity registerEnabler is
408
409     port
410     (
411         desReg      : in std logic;
412         upd         : in std logic;
413         enableR0    : out std logic;
414         enableR1    : out std logic
415     );
416
417 end entity;
418
419 architecture rtl of registerEnabler is
420     signal setA, setB: std logic;
421 begin
422
423     process(upd, desReg) is
424     begin
425
426         if(upd = '1') then -- clock is off when it's ONE
427             --None of the registers are updated
428             setA <= '0';
429             setB <= '0';
430         elsif (upd = '0') then -- clock is ON when it's ZERO
431             -- if 0 -> R0 is updated, if 1-> R1 is updated
432             if(desReg = '0') then -- update R0
433                 setA <= '1';
434                 setB <= '0';
435             else -- Update R1
436                 setA <= '0';
437                 setB <= '1';
438             end if;
439
440         end if;
441     end process;
442
443     enableR0 <= setA;
444     enableR1 <= setB;
445
446     -- if (upd ='0') then
447     --     if(desReg ='0') then
448     --         enableR0 <= '1';
449     --     else
450     --         enableR1 <= '1';
451     --     end if;
452     -- end if;
453     -- end process;
454
455 end rtl;
456
457
458 -----
459 --Reg Multiplexor: Decides which register value is to be used
460
461 library ieee;
462 use ieee.std logic 1164.all;
463
464 entity regMux is
465
466     port
467     (
468         r0: in std logic vector (7 downto 0);
469         r1: in std logic vector (7 downto 0);
470         s: in std logic;
471         y: out std logic vector (7 downto 0)
472     );
473
```

```
474     end entity;
475
476     architecture rtl of regMux is
477     signal currAns: std_logic_vector (7 downto 0);
478     begin
479
480         process(r0,r1, s) is
481         begin
482
483             if (s = '0') then
484                 currAns <= r0;
485             else
486                 currAns <= r1;
487
488             end if;
489
490         end process;
491         y <= currAns;
492
493     end rtl;
494
495
496     library ieee;
497     use ieee.std_logic_1164.all;
498     use ieee.numeric_std.all;
499     -----
500     -- multiplexer to select the instruction
501
502     entity mux is
503     port (
504         instr    : in std_logic_vector (3 downto 0);
505         addsub    : in std_logic_vector (7 downto 0);
506         sub       : in std_logic_vector (7 downto 0);
507         isConst   : in std_logic;
508         addConst  : in std_logic_vector (7 downto 0);
509         subConst  : in std_logic_vector (7 downto 0);
510         xand      : in std_logic_vector (7 downto 0);
511         mov       : in std_logic_vector (7 downto 0);
512         neg       : in std_logic_vector (7 downto 0);
513         Eor       : in std_logic_vector (7 downto 0);
514         shl       : in std_logic_vector (7 downto 0);
515         shr       : in std_logic_vector (7 downto 0);
516         to_reg    : out std_logic_vector (7 downto 0)
517     );
518     end entity;
519
520     architecture synth of mux is
521     signal currAns: std_logic_vector (7 downto 0);
522
523     begin
524         process (instr, isConst)
525         begin
526             if (instr = "0000") then -- add instruction
527                 currAns <= addsub;
528
529             elsif (instr = "1001") then --add const
530                 currAns <= addConst;
531
532             elsif(instr = "0111") then -- subtract instruction
533                 currAns <= sub;
534
535             elsif (instr = "1000") then
536                 currAns <= subConst;-- subtract constant
537
538             elsif (instr = "0001") then
539                 --and
540                 currAns <= xand;
541             elsif(instr = "0010") then
```



```

542         --mov
543         currAns <= mov;
544     elsif(instr = "0011") then
545         --neg
546         currAns <= neg;
547     elsif(instr = "0100") then
548         --or
549         currAns <= Eor;
550     elsif(instr = "0101") then
551         --shl
552         currAns <= shl;
553     elsif(instr = "0110") then
554         --shr
555         currAns <= shr;
556
557     end if;
558
559 end process;
560
561 to reg <= currAns;
562
563 end;
564
565
566 -----
567 library ieee;
568 use ieee.std_logic_1164.all;
569 use ieee.numeric_std.all;
570
571 -- mov entity
572 --Move 8 bit value to Rd
573 entity mov is
574     port(
575         num: in std_logic_vector (7 downto 0);
576         rd : out std_logic_vector (7 downto 0)
577     );
578 end entity;
579
580 architecture m of mov is
581 begin
582     rd <= num;
583 end;
584
585 -----
586
587 library ieee;
588 use ieee.std_logic_1164.all;
589 use ieee.numeric_std.all;
590 -----
591 -- out entity- display
592 entity display is
593     port (
594         rs: in std_logic_vector ( 7 downto 0);
595         a_seg: out std_logic_vector (6 downto 0);
596         b_seg : out std_logic_vector (6 downto 0)
597     );
598 end entity;
599
600 architecture d of display is
601     signal a,b: std_logic_vector (3 downto 0);
602
603 begin
604     a <= rs(7 downto 4);
605     b <= rs(3 downto 0);
606     process(a)
607     begin
608         case a is
609             when "0000" =>

```

```
610         a_seg <= "0000001"; --0
611
612     when "0001" =>
613         a_seg <= "1001111"; --1
614
615     when "0010" =>
616         a_seg <= "0010010"; --2
617
618     when "0011" =>
619         a_seg <= "0000110"; --3
620
621     when "0100" =>
622         a_seg <= "1001100"; --4
623
624     when "0101" =>
625         a_seg <= "0100100"; --5
626
627     when "0110" =>
628         a_seg <= "0100000"; --6
629
630     when "0111" =>
631         a_seg <= "0001111"; --7
632
633     when "1000" =>
634         a_seg <= "0000000"; --8
635
636     when "1001" =>
637         a_seg <= "0000100"; --9
638
639     when "1010" =>
640         a_seg <= "0001000"; --A
641
642     when "1011" =>
643         a_seg <= "1100000"; --b
644
645     when "1100" =>
646         a_seg <= "0110001"; --C
647
648     when "1101" =>
649         a_seg <= "1000010"; --d
650
651     when "1110" =>
652         a_seg <= "0110000"; --E
653
654     when "1111" =>
655         a_seg <= "0111000"; --F
656
657     end case;
658 end process;
659 process (b)
660 begin
661     case b is
662
663     when "0000" =>
664         b_seg <= "0000001"; --0
665
666     when "0001" =>
667         b_seg <= "1001111"; --1
668
669     when "0010" =>
670         b_seg <= "0010010"; --2
671
672     when "0011" =>
673         b_seg <= "0000110"; --3
674
675     when "0100" =>
676         b_seg <= "1001100"; --4
677
```

```
678         when "0101" =>
679             b seg <= "0100100"; --5
680
681         when "0110" =>
682             b seg <= "0100000"; --6
683
684         when "0111" =>
685             b seg <= "0001111"; --7
686
687         when "1000" =>
688             b seg <= "0000000"; --8
689
690         when "1001" =>
691             b seg <= "0000100"; --9
692
693         when "1010" =>
694             b seg <= "0001000"; --A
695
696         when "1011" =>
697             b seg <= "1100000"; --b
698
699         when "1100" =>
700             b seg <= "0110001"; --C
701
702         when "1101" =>
703             b seg <= "1000010"; --d
704
705         when "1110" =>
706             b seg <= "0110000"; --E
707
708         when "1111" =>
709             b seg <= "0111000"; --F
710
711     end case;
712 end process;
713
714 end;
715
716 -----
717
718 library ieee;
719 use ieee.std_logic_1164.all;
720 use ieee.numeric_std.all;
721
722 -- full adder for 1 bit register
723 entity oneBitAdder is
724     port (a, b, carry_in : in std_logic;
725           carry_out, sum : out std_logic);
726 end entity;
727
728 architecture rtl of oneBitAdder is
729     signal a1, a2, a3: std_logic;
730 begin
731
732     sum <= a xor b xor carry_in;
733     carry_out <= (a and b) or (carry_in and a) or (carry_in and b);
734
735 end rtl;
736
737
738
739
740 -----
741
742 library ieee;
743 use ieee.std_logic_1164.all;
744 use ieee.numeric_std.all;
745
```

```
746 entity add is
747     port (a,b: in std logic vector(7 downto 0);
748           isSub : in std logic;
749           sum: out std_logic_vector (7 downto 0)
750     );
751 end entity;
752 architecture a of add is
753
754     signal c1, c2, c3, c4, c5, c6, c7, c8: std logic;
755
756     component oneBitAdder
757         port (a, b, carry in : in std logic;
758               carry_out, sum : out std_logic);
759     end component;
760
761
762 begin
763
764     add0: oneBitAdder port map(a(0), b(0), '0', c1,sum(0));
765     add1: oneBitAdder port map(a(1), b(1), c1 ,c2, sum(1));
766     add2: oneBitAdder port map(a(2), b(2), c2, c3, sum(2));
767     add3: oneBitAdder port map(a(3), b(3), c3, c4, sum(3));
768     add4: oneBitAdder port map(a(4), b(4), c4, c5, sum(4));
769     add5: oneBitAdder port map(a(5), b(5), c5, c6 ,sum(5));
770     add6: oneBitAdder port map(a(6), b(6), c6, c7 ,sum(6));
771     add7: oneBitAdder port map(a(7), b(7), c7, c8 ,sum(7));
772
773 end;
774 -----
775 library ieee;
776 use ieee.std_logic_1164.all;
777 use ieee.numeric_std.all;
778
779 entity subtract is
780 port (a,b: in std logic vector(7 downto 0);
781       isSub : in std logic;
782       sum: out std_logic_vector (7 downto 0)
783 );
784 end entity;
785 architecture s of subtract is
786
787     signal c1, c2, c3, c4, c5, c6, c7, c8: std logic;
788
789     component oneBitAdder
790         port (a, b, carry in : in std logic;
791               carry_out, sum : out std_logic);
792     end component;
793
794
795 begin
796
797     add0: oneBitAdder port map(a(0), not b(0), '1', c1,sum(0));
798     add1: oneBitAdder port map(a(1), not b(1), c1 ,c2, sum(1));
799     add2: oneBitAdder port map(a(2), not b(2), c2, c3, sum(2));
800     add3: oneBitAdder port map(a(3), not b(3), c3, c4, sum(3));
801     add4: oneBitAdder port map(a(4), not b(4), c4, c5, sum(4));
802     add5: oneBitAdder port map(a(5), not b(5), c5, c6 ,sum(5));
803     add6: oneBitAdder port map(a(6), not b(6), c6, c7 ,sum(6));
804     add7: oneBitAdder port map(a(7), not b(7), c7, c8 ,sum(7));
805
806 end;
807
808 -----
809 library ieee;
810 use ieee.std logic 1164.all;
811 use ieee.numeric_std.all;
812 entity getConst is
813     port (
```

```
814         a: in std_logic_vector(2 downto 0);
815         isConst : in std_logic;
816         new_val: out std_logic_vector(7 downto 0)
817     );
818 end entity;
819
820 architecture gC of getConst is
821
822 begin
823     process(a, isConst)
824     begin
825         if (isConst = '1') then
826             new_val <= (7 downto 3 => '0') & a;
827         end if;
828     end process;
829 end;
830 -----
831
832
833 library ieee;
834 use ieee.std_logic_1164.all;
835 use ieee.numeric_std.all;
836 -----
837 -- and entity, and Rs and Rd, stores in Rd
838
839 entity andReg is
840     port (
841         r0,r1: in std_logic_vector (7 downto 0);
842         rd: out std_logic_vector (7 downto 0)
843     );
844 end entity;
845
846 architecture aR of andReg is
847 begin
848     rd <= r0 and r1;
849
850 end;
851
852 -----
853
854
855 library ieee;
856 use ieee.std_logic_1164.all;
857 use ieee.numeric_std.all;
858 -----
859 -- neg entity - complement Rs, stores in Rd
860 entity neg is
861     port (
862         rs: in std_logic_vector (7 downto 0);
863         rd: out std_logic_vector (7 downto 0)
864     );
865 end entity;
866
867 architecture n of neg is
868 begin
869     rd <= not rs;
870 end;
871
872 library ieee;
873 use ieee.std_logic_1164.all;
874 use ieee.numeric_std.all;
875 -----
876 -- eor entity-- xor Rs and Rd, stores in Rd
877 entity eor is
878     port(
879         r0, r1 : in std_logic_vector(7 downto 0);
880         rd: out std_logic_vector(7 downto 0)
881     );
```

```
882     end entity;
883
884     architecture e of eor is
885     begin
886         rd <= r0 xor r1;
887     end;
888
889
890     -----
891
892
893     library ieee;
894     use ieee.std_logic_1164.all;
895     use ieee.numeric_std.all;
896
897     -----
898     -- lsl entity - left shift Rs by 1 bit, stores in Rd
899     entity lsl is
900     port (
901         rs: in std_logic_vector (7 downto 0);
902         rd: out std_logic_vector (7 downto 0)
903     );
904     end entity;
905
906     architecture l of lsl is
907     signal shl temp : unsigned(7 downto 0);
908     begin
909
910         shl temp <= unsigned(rs);
911         rd <= std_logic_vector (shift_left(shl temp,1));
912     end;
913
914     -----
915
916
917     library ieee;
918     use ieee.std_logic_1164.all;
919     use ieee.numeric_std.all;
920
921     -----
922     -- lsr entity - right shift Rs by 1 bit, stores in Rd
923     entity lsr is
924     port (
925         rs: in std_logic_vector (7 downto 0);
926         rd: out std_logic_vector (7 downto 0)
927     );
928     end entity;
929
930     architecture r of lsr is
931
932     signal shr temp : unsigned(7 downto 0);
933     begin
934
935         shr temp <= unsigned(rs);
936         rd <= std_logic_vector (shift_right(shr temp,1));
937     end;
938
939
940
```