Lab 5 Tutorial

CPU Control Unit Design

OVERVIEW

In this lab we will implement the control unit that will provide the control signals to the 32-bit CPU data path designed in Lab 4b.

The CPU Data-Path is shown in the figure 1 below.

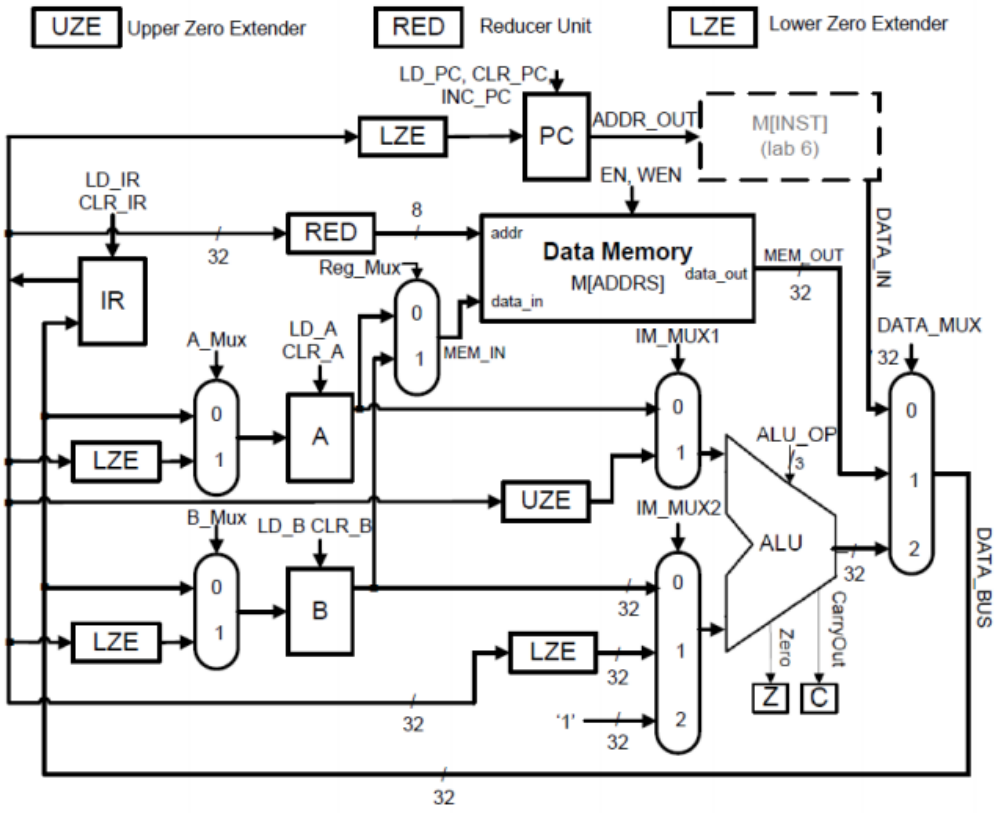


Figure 1: CPU Data-Path (https://www.ee.ryerson.ca/~courses/coe608/labs/lab4_b.pdf)

Table 1 below lists the CPU's supported instructions and their format.

Table 2 lists the control signal settings for each of instructions supported by our CPU.

Please refer to these tables to verify that your control unit functions correctly.

Table 1: Supported Instructions and their Format (https://www.ee.ryerson.ca/~courses/coe608/labs/CPU_Specification.pdf)

Mnemonic	Function	Instruction Word		
		IR[31..28]	IR[27..16]	IR[15..0]
LDAI	A <= IR[15:0]	0000	X	IMM
LDBI	B <= IR[15:0]	0001	X	IMM
STA	M[ADDRS] <= A, ADDRS <= IR[15:0]	0010	X	ADDRS
STB	M[ADDRS] <= B, ADDRS <= IR[15:0]	0011	X	ADDRS
LDA	A <= M[ADDRS] , ADDRS <= IR[15:0]	1001	X	ADDRS
LDB	B <= M[ADDRS] , ADDRS <= IR[15:0]	1010	X	ADDRS
LUI	A[31:16] <= IR[15:0], A[15:0] <= 0	0100	X	IMM
JMP	PC <= IR[15..0]	0101	X	ADDRS
BEQ	IF(A==B) then PC <= IR[15..0]	0110	X	ADDRS
BNE	IF(A!=B) then PC <= IR[15..0]	1000	X	ADDRS
		IR[31..28]	**IR[27..24]**	**IR[15..0]**
ADD	A <= A + B	0111	0000	X
ADDI	A <= A + IR[15..0]	0111	0001	IMM
SUB	A <= A - B	0111	0010	X
INCA	A <= A + 1	0111	0011	X
ROL	A <= A << 1	0111	0100	X
CLRA	A <= 0	0111	0101	X
CLRB	B <= 0	0111	0110	X
CLRC	C <= 0	0111	0111	X
CLRZ	Z <= 0	0111	1000	X
ANDI	A <= A AND IR[15..0]	0111	1001	IMM
TSTZ	If Z = 1 then PC <= PC + 1	0111	1010	X
AND	A <= A AND B	0111	1011	X
TSTC	If C = 1 then PC <= PC + 1	0111	1100	X
ORI	A <= A OR IR[15..0]	0111	1101	IMM
DECA	A <= A - 1	0111	1110	X
ROR	A <= A >> 1	0111	1111	X

Table 2: Data-Path Control Signals (https://www.ee.ryerson.ca/~courses/coe608/DATAPATH.pdf)

INST	CLR_IR LD_IR	LD_PC INC_PC	CLR_A LD_A	CLR_B LD_B	CLR_C LD_C	CLR_Z LD_Z	ALU OP	EN WEN	A/B MUX	REG MUX	Data MUX	IM_MUX1 IM_MUX2
LDA	0/0	0/0	0/1	0/0	0/0	0/0	XXX	1/0	0/X	X	01	X
LDB	0/0	0/0	0/0	0/1	0/0	0/0	XXX	1/0	X/0	X	01	X
STA	0/0	0/0	0/0	0/0	0/0	0/0	XXX	1/1	X	0	X	X
STB	0/0	0/0	0/0	0/0	0/0	0/0	XXX	1/1	X	1	X	X
JMP	0/0	1/0	0/0	0/0	0/0	0/0	XXX	X	X	X	X	X
LDAI	0/0	0/0	0/1	0/0	0/0	0/0	XXX	X	1/X	X	X	X
LDBI	0/0	0/0	0/0	0/1	0/0	0/0	XXX	X	X/1	X	X	X
LUI	0/0	0/0	0/1	1/0	0/0	0/0	001	X	0/X	X	10	1/X
ANDI	0/0	0/0	0/1	0/0	0/1	0/1	000	X	0/X	X	10	0/01
DECA	0/0	0/0	0/1	0/0	0/1	0/1	110	X	0/X	X	10	0/10
ADD	0/0	0/0	0/1	0/0	0/1	0/1	010	X	0/X	X	10	0/00
SUB	0/0	0/0	0/1	0/0	0/1	0/1	110	X	0/X	X	10	0/00
INCA	0/0	0/0	0/1	0/0	0/1	0/1	010	X	0/X	X	10	0/10
AND	0/0	0/0	0/1	0/0	0/1	0/1	000	X	0/X	X	10	0/00
ADDI	0/0	0/0	0/1	0/0	0/1	0/1	010	X	0/X	X	10	0/01
ORI	0/0	0/0	0/1	0/0	0/1	0/1	001	X	0/X	X	10	0/01
ROL	0/0	0/0	0/1	0/0	0/1	0/1	100	X	0/X	X	10	0/X
ROR	0/0	0/0	0/1	0/0	0/1	0/1	101	X	0/X	X	10	0/X
CLRA	0/0	0/0	1/0	0/0	0/0	0/0	XXX	X	X	X	X	X
CLRB	0/0	0/0	0/0	1/0	0/0	0/0	XXX	X	X	X	X	X
CLRC	0/0	0/0	0/0	0/0	1/0	0/0	XXX	X	X	X	X	X
CLRZ	0/0	0/0	0/0	0/0	0/0	1/0	XXX	X	X	X	X	X
PC <= PC+4	0/0	1/1	0/0	0/0	0/0	0/0	XXX	X	X	X	X	X
IR <= M[INST]	0/1	0/0	0/0	0/0	0/0	0/0	XXX	X	X	X	00	X
PC <= IR[15..0]	0/0	1/0	0/0	0/0	0/0	0/0	XXX	X	X	X	X	X

PROCEDURE

Control Unit (Control.vhd)

```
1   library ieee;
2   use ieee.std_logic_1164.ALL;
3
4   ENTITY Control IS
5      PORT(
6            clk, mclk : IN STD_LOGIC;
7            enable : IN STD_LOGIC;
8            statusC, statusZ : IN STD_LOGIC;
9            INST : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
10           A_Mux, B_Mux : OUT STD_LOGIC;
11           IM_MUX1, REG_Mux : OUT STD_LOGIC;
12           IM_MUX2, DATA_Mux : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
13           ALU_op : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
14           inc_PC, ld_PC : OUT STD_LOGIC;
15           clr_IR : OUT STD_LOGIC;
16           ld_IR : OUT STD_LOGIC;
17           clr_A, clr_B, clr_C, clr_Z : OUT STD_LOGIC;
18           ld_A, ld_B, ld_C, ld_Z : OUT STD_LOGIC;
19           T : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
20           wen, en : OUT STD_LOGIC
21        );
22  END Control;
23
24  ARCHITECTURE description OF Control IS
25     TYPE STATETYPE IS (state_0, state_1, state_2);
26     SIGNAL present_state: STATETYPE;
27     SIGNAL Instruction_sig: STD_LOGIC_VECTOR (3 downto 0);
28     SIGNAL Instruction_sig2:STD_LOGIC_VECTOR(7 downto 0);
29  BEGIN
30     Instruction_sig <= INST(31 DOWNTO 28);
31     Instruction_sig2<= INST(31 DOWNTO 24);
32
33     -------- OPERATION DECODER ---------
34     PROCESS (present_state, INST, statusC, statusZ, enable,Instruction_sig,Instruction_sig2)
35     BEGIN
36        if enable = '1' then
37           if present_state = state_0 then
38              DATA_Mux<="00";--Fetch Address of next instruction
39              clr_IR<='0';
40              ld_IR<='1';
41              ld_PC<='0';
42              inc_PC<='0';
43              clr_A<='0';
44              ld_A<='0';
45              ld_B<='0';
46              clr_B<='0';
47              clr_C<='0';
48              ld_C<='0';
49              clr_Z<='0';
50              ld_Z<='0';
51              en <= '0';
52              wen <= '0';
53
54           elsif present_state = state_1 then
55              clr_IR<='0';--INCREMENT PC COUNTER
56              ld_IR<='0';
57              ld_PC<='1';
58              inc_PC<='1';
59              clr_A<='0';
60              ld_A<='0';
61              ld_B<='0';
```

```
62              clr_B<='0';
63              clr_C<='0';
64              ld_C<='0';
65              clr_Z<='0';
66              ld_Z<='0';
67              en <= '0';
68              wen <= '0';
69
70              if Instruction_sig = "0010" then--STA
71                 clr_IR<='0';
72                 ld_IR<='0';
73                 ld_PC<='1';
74                 inc_PC<='1';
75                 clr_A<='0';
76                 ld_A<='0';
77                 ld_B<='0';
78                 clr_B<='0';
79                 clr_C<='0';
80                 ld_C<='0';
81                 clr_Z<='0';
82                 ld_Z<='0';
83                 REG_Mux<='0';
84                 DATA_Mux<="00";
85                 en <= '1';
86                 wen <= '1';
87
88              elsif Instruction_sig = "0011" then--STB
89                 clr_IR<='0';
90                 ld_Z<='0';
91                 ld_IR<='0';
92                 ld_PC<='1';
93                 inc_PC<='1';
94                 clr_A<='0';
95                 ld_A<='0';
96                 ld_B<='0';
97                 clr_B<='0';
98                 clr_C<='0';
99                 ld_C<='0';
100                clr_Z<='0';
101                ld_Z<='0';
102                REG_Mux<='1';
103                DATA_Mux<="00";
104                en <= '1';
105                wen <= '1';
106             elsif Instruction_sig = "1001" then --LDA
107                clr_IR<='0';
108                ld_IR<='0';
109                ld_PC<='1';
110                inc_PC<='1';
111                clr_A<='0';
112                ld_A<='1';
113                ld_B<='0';
114                clr_B<='0';
115                clr_C<='0';
116                ld_C<='0';
117                clr_Z<='0';
118                ld_Z<='0';
119                A_Mux<='0';
120                DATA_Mux<="01";
121                EN <= '1';
122                WEN <= '0';
```

```
123             elsif Instruction_sig = "1010" then --LDB
124                clr_IR<='0';
125                ld_IR<='0';
126                ld_PC<='1';
127                inc_PC<='1';
128                clr_A<='0';
129                ld_A<='0';
130                ld_B<='1';
131                clr_B<='0';
132                clr_C<='0';
133                ld_C<='0';
134                clr_Z<='0';
135                ld_Z<='0';
136                B_Mux<='0';
137                DATA_Mux<="01";
138                EN <= '1';
139                WEN <= '0';
140             end if;--END IF FOR LOAD STORE IN STAGE 1
141
142          elsif present_state = state_2 then
143
144             if Instruction_sig = "0101" then --JUMP
145                clr_IR<='0';
146                ld_IR<='0';
147                ld_PC<='1';
148                inc_PC<='0';
149                clr_A<='0';
150                ld_A<='0';
151                ld_B<='0';
152                clr_B<='0';
153                clr_C<='0';
154                ld_C<='0';
155                clr_Z<='0';
156                ld_Z<='0';
157
158             elsif Instruction_sig = "0110" then --BEQ
159                clr_IR<='0';
160                ld_IR<='0';
161                ld_PC<='1';
162                inc_PC<='0';
163                clr_A<='0';
164                ld_A<='0';
165                ld_B<='0';
166                clr_B<='0';
167                clr_C<='0';
168                ld_C<='0';
169                clr_Z<='0';
170                ld_Z<='0';
171
172             elsif Instruction_sig = "1000" then --BNE
173                clr_IR<='0';
174                ld_IR<='0';
175                ld_PC<='1';
176                inc_PC<='0';
177                clr_A<='0';
178                ld_A<='0';
179                ld_B<='0';
180                clr_B<='0';
181                clr_C<='0';
182                ld_C<='0';
183                clr_Z<='0';
```

```
184                ld_Z<='0';
185
186             elsif Instruction_sig = "1001" then --LDA
187                clr_IR<='0';
188                ld_IR<='0';
189                ld_PC<='0';
190                inc_PC<='0';
191                clr_A<='0';
192                ld_A<='1';
193                ld_B<='0';
194                clr_B<='0';
195                clr_C<='0';
196                ld_C<='0';
197                clr_Z<='0';
198                ld_Z<='0';
199                A_Mux<='0';
200                DATA_Mux<="01";
201                EN <= '1';
202                WEN <= '0';
203             elsif Instruction_sig = "1010" then --LDB
204                clr_IR<='0';
205                ld_IR<='0';
206                ld_PC<='0';
207                inc_PC<='0';
208                clr_A<='0';
209                ld_A<='0';
210                ld_B<='1';
211                clr_B<='0';
212                clr_C<='0';
213                ld_C<='0';
214                clr_Z<='0';
215                ld_Z<='0';
216                B_Mux<='0';
217                DATA_Mux<="01";
218                EN <= '1';
219                WEN <= '0';
220             elsif Instruction_sig = "0010" then--STA
221                clr_IR<='0';
222                ld_IR<='0';
223                ld_PC<='0';
224                inc_PC<='0';
225                clr_A<='0';
226                ld_A<='0';
227                ld_B<='0';
228                clr_B<='0';
229                clr_C<='0';
230                ld_C<='0';
231                clr_Z<='0';
232                ld_Z<='0';
233                REG_Mux<='0';
234                DATA_Mux<="00";
235                EN <= '1';
236                WEN <='1';
237
238             elsif Instruction_sig = "0011" then--STB
239                clr_IR<='0';
240                ld_IR<='0';
241                ld_PC<='0';
242                inc_PC<='0';
243                clr_A<='0';
244                ld_A<='0';
```

```
245                ld_B<='0';
246                clr_B<='0';
247                clr_C<='0';
248                ld_C<='0';
249                clr_Z<='0';
250                ld_Z<='0';
251                REG_Mux<='1';
252                DATA_Mux<="00";
253                en <= '1';
254                wen <= '1';
255             elsif Instruction_sig = "0000" then --LDAI
256                clr_IR<='0';
257                ld_IR<='0';
258                ld_PC<='0';
259                inc_PC<='0';
260                clr_A<='0';
261                ld_A<='1';
262                ld_B<='0';
263                clr_B<='0';
264                clr_C<='0';
265                ld_C<='0';
266                clr_Z<='0';
267                ld_Z<='0';
268                A_Mux <='1';
269             elsif Instruction_sig = "0001" then --LDBI
270                clr_IR<='0';
271                ld_IR<='0';
272                ld_PC<='0';
273                inc_PC<='0';
274                clr_A<='0';
275                ld_A<='0';
276                ld_B<='1';
277                clr_B<='0';
278                clr_C<='0';
279                ld_C<='0';
280                clr_Z<='0';
281                ld_Z<='0';
282                B_Mux <='1';
283             elsif Instruction_sig = "0100" then --LUI
284                clr_IR<='0';
285                ld_IR<='0';
286                ld_PC<='0';
287                inc_PC<='0';
288                clr_A<='0';
289                ld_A<='1';
290                ld_B<='0';
291                clr_B<='1';
292                clr_C<='0';
293                ld_C<='0';
294                clr_Z<='0';
295                ld_Z<='0';
296                ALU_op<="001";
297                A_Mux<='0';
298                DATA_Mux<="10";
299                IM_MUX1<='1';
300             elsif Instruction_sig2 = "01111001" then --ANDI
301                clr_IR<='0';
302                ld_IR<='0';
303                ld_PC<='0';
304                inc_PC<='0';
305                clr_A<='0';
```

```
306                ld_A<='1';
307                ld_B<='0';
308                clr_B<='0';
309                clr_C<='0';
310                ld_C<='1';
311                clr_Z<='0';
312                ld_Z<='1';
313                ALU_op<="000";
314                A_Mux<='0';
315                DATA_Mux<="10";
316                IM_MUX1<='0';
317                IM_MUX2<="01";
318             elsif Instruction_sig2 = "01111110" then --DECA
319                clr_IR<='0';
320                ld_IR<='0';
321                ld_PC<='0';
322                inc_PC<='0';
323                clr_A<='0';
324                ld_A<='1';
325                ld_B<='0';
326                clr_B<='0';
327                clr_C<='0';
328                ld_C<='1';
329                clr_Z<='0';
330                ld_Z<='1';
331                ALU_op<="110";
332                A_Mux<='0';
333                DATA_Mux<="10";
334                IM_MUX1<='0';
335                IM_MUX2<="10";
336             elsif Instruction_sig2 = "01110000" then --ADD
337                clr_IR<='0';
338                ld_IR<='0';
339                ld_PC<='0';
340                inc_PC<='0';
341                clr_A<='0';
342                ld_A<='1';
343                ld_B<='0';
344                clr_B<='0';
345                clr_C<='0';
346                ld_C<='1';
347                clr_Z<='0';
348                ld_Z<='1';
349                ALU_op<="010";
350                A_Mux<='0';
351                DATA_Mux<="10";
352                IM_MUX1<='0';
353                IM_MUX2<="00";
354             elsif Instruction_sig2 = "01110010" then --SUB
355                clr_IR<='0';
356                ld_IR<='0';
357                ld_PC<='0';
358                inc_PC<='0';
359                clr_A<='0';
360                ld_A<='1';
361                ld_B<='0';
362                clr_B<='0';
363                clr_C<='0';
364                ld_C<='1';
365                clr_Z<='0';
366                ld_Z<='1';
```

```
367                ALU_op<="110";
368                A_Mux<='0';
369                DATA_Mux<="10";
370                IM_MUX1<='0';
371                IM_MUX2<="00";
372             elsif Instruction_sig2 = "01110011" then --INCA
373                clr_IR<='0';
374                ld_IR<='0';
375                ld_PC<='0';
376                inc_PC<='0';
377                clr_A<='0';
378                ld_A<='1';
379                ld_B<='0';
380                clr_B<='0';
381                clr_C<='0';
382                ld_C<='1';
383                clr_Z<='0';
384                ld_Z<='1';
385                ALU_op<="010";
386                A_Mux<='0';
387                DATA_Mux<="10";
388                IM_MUX1<='0';
389                IM_MUX2<="10";
390             elsif Instruction_sig2 = "01111011" then --AND
391                clr_IR<='0';
392                ld_IR<='0';
393                ld_PC<='0';
394                inc_PC<='0';
395                clr_A<='0';
396                ld_A<='1';
397                ld_B<='0';
398                clr_B<='0';
399                clr_C<='0';
400                ld_C<='1';
401                clr_Z<='0';
402                ld_Z<='1';
403                ALU_op<="000";
404                A_Mux<='0';
405                DATA_Mux<="10";
406                IM_MUX1<='0';
407                IM_MUX2<="00";
408             elsif Instruction_sig2 = "01110001" then --ADDI
409                clr_IR<='0';
410                ld_IR<='0';
411                ld_PC<='0';
412                inc_PC<='0';
413                clr_A<='0';
414                ld_A<='1';
415                ld_B<='0';
416                clr_B<='0';
417                clr_C<='0';
418                ld_C<='1';
419                clr_Z<='0';
420                ld_Z<='1';
421                ALU_op<="010";
422                A_Mux<='0';
423                DATA_Mux<="10";
424                IM_MUX1<='0';
425                IM_MUX2<="01";
426             elsif Instruction_sig2 = "01111101" then --ORI
427                clr_IR<='0';
```

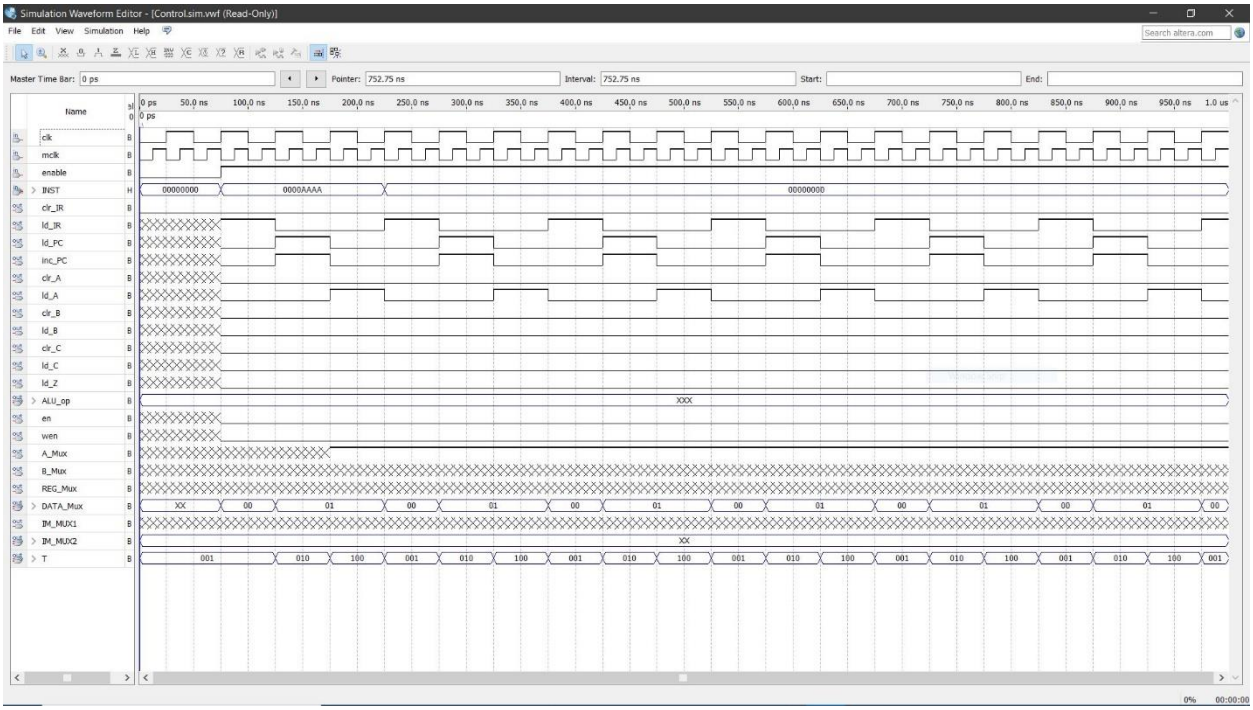
```
428                ld_IR<='0';
429                ld_PC<='0';
430                inc_PC<='0';
431                clr_A<='0';
432                ld_A<='1';
433                ld_B<='0';
434                clr_B<='0';
435                clr_C<='0';
436                ld_C<='1';
437                clr_Z<='0';
438                ld_Z<='1';
439                ALU_op<="001";
440                A_Mux<='0';
441                DATA_Mux<="10";
442                IM_MUX1<='0';
443                IM_MUX2<="01";
444             elsif Instruction_sig2 = "01110100" then --ROL
445                clr_IR<='0';
446                ld_IR<='0';
447                ld_PC<='0';
448                inc_PC<='0';
449                clr_A<='0';
450                ld_A<='1';
451                ld_B<='0';
452                clr_B<='0';
453                clr_C<='0';
454                ld_C<='1';
455                clr_Z<='0';
456                ld_Z<='1';
457                ALU_op<="100";
458                A_Mux<='0';
459                DATA_Mux<="10";
460                IM_MUX1<='0';
461             elsif Instruction_sig2 = "01111111" then --ROR
462                clr_IR<='0';
463                ld_IR<='0';
464                ld_PC<='0';
465                inc_PC<='0';
466                clr_A<='0';
467                ld_A<='1';
468                ld_B<='0';
469                clr_B<='0';
470                clr_C<='0';
471                ld_C<='1';
472                clr_Z<='0';
473                ld_Z<='1';
474                ALU_op<="101";
475                A_Mux<='0';
476                DATA_Mux<="10";
477                IM_MUX1<='0';
478             elsif Instruction_sig2 = "01110101" then --CLR_A
479                clr_IR<='0';
480                ld_IR<='0';
481                ld_PC<='0';
482                inc_PC<='0';
483                clr_A<='1';
484                ld_A<='0';
485                ld_B<='0';
486                clr_B<='0';
487                clr_C<='0';
488                ld_C<='0';
```

```
489                clr_Z<='0';
490                ld_Z<='0';
491             elsif Instruction_sig2 = "01110110" then --CLR_B
492                clr_IR<='0';
493                ld_IR<='0';
494                ld_PC<='0';
495                inc_PC<='0';
496                clr_A<='0';
497                ld_A<='0';
498                ld_B<='0';
499                clr_B<='1';
500                clr_C<='0';
501                ld_C<='0';
502                clr_Z<='0';
503                ld_Z<='0';
504             elsif Instruction_sig2 = "01110111" then --CLR_C
505                clr_IR<='0';
506                ld_IR<='0';
507                ld_PC<='0';
508                inc_PC<='0';
509                clr_A<='0';
510                ld_A<='0';
511                ld_B<='0';
512                clr_B<='0';
513                clr_C<='1';
514                ld_C<='0';
515                clr_Z<='0';
516                ld_Z<='0';
517             elsif Instruction_sig2 = "01111000" then --CLR_Z
518                clr_IR<='0';
519                ld_IR<='0';
520                ld_PC<='0';
521                inc_PC<='0';
522                clr_A<='0';
523                ld_A<='0';
524                ld_B<='0';
525                clr_B<='0';
526                clr_C<='0';
527                ld_C<='0';
528                clr_Z<='1';
529                ld_Z<='0';
530             elsif Instruction_sig2 = "01111010" then --TSTZ
531                if(statusZ ='1')then
532                   clr_IR<='0';--INCREMENT PC COUNTER
533                   ld_IR<='0';
534                   ld_PC<='1';
535                   inc_PC<='1';
536                   clr_A<='0';
537                   ld_A<='0';
538                   ld_B<='0';
539                   clr_B<='0';
540                   clr_C<='0';
541                   ld_C<='0';
542                   clr_Z<='0';
543                   ld_Z<='0';
544                end if;
545             elsif Instruction_sig2 = "01111100" then --TSTC
546                if(statusC ='1')then
547                   clr_IR<='0';--INCREMENT PC COUNTER
548                   ld_IR<='0';
549                   ld_PC<='1';
```

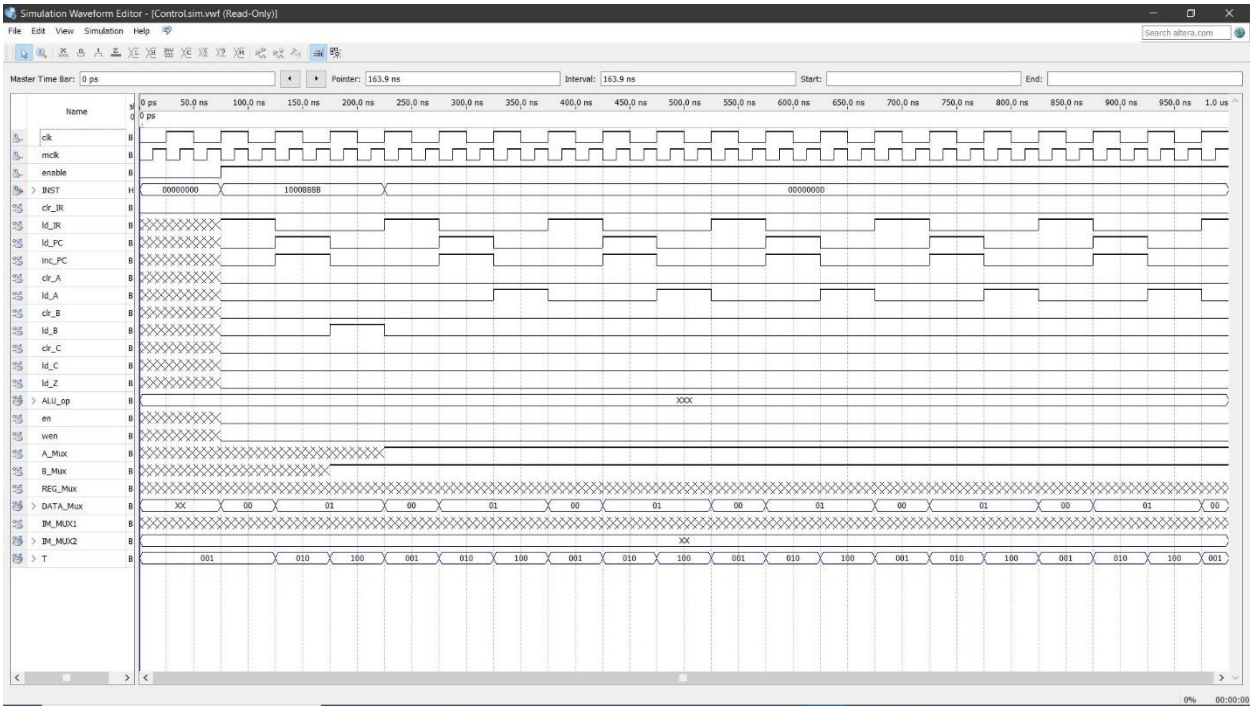
```
                    inc_PC<='1';
                    clr_A<='0';
                    ld_A<='0';
                    ld_B<='0';
                    clr_B<='0';
                    clr_C<='0';
                    ld_C<='0';
                    clr_Z<='0';
                    ld_Z<='0';
                end if;
             end if;--For state 2 Ops
          end if;
       end if; --For Enable
    END process;
-------- STATE MACHINE ---------
    PROCESS (clk, enable)
    begin
       if enable = '1' then
          if rising_edge (clk) then
             if present_state = state_0 then present_state <= state_1;
             elsif present_state = state_1 then present_state <= state_2;
             else present_state <= state_0;
             end if;
          end if;
       else present_state <= state_0;
       end if;
    END process;

    WITH present_state select
       T <=  "001" when state_0,
             "010" when state_1,
             "100" when state_2,
             "001" when others;
END description;
```

FUNCTIONAL SIMULATION WAVEFORMS

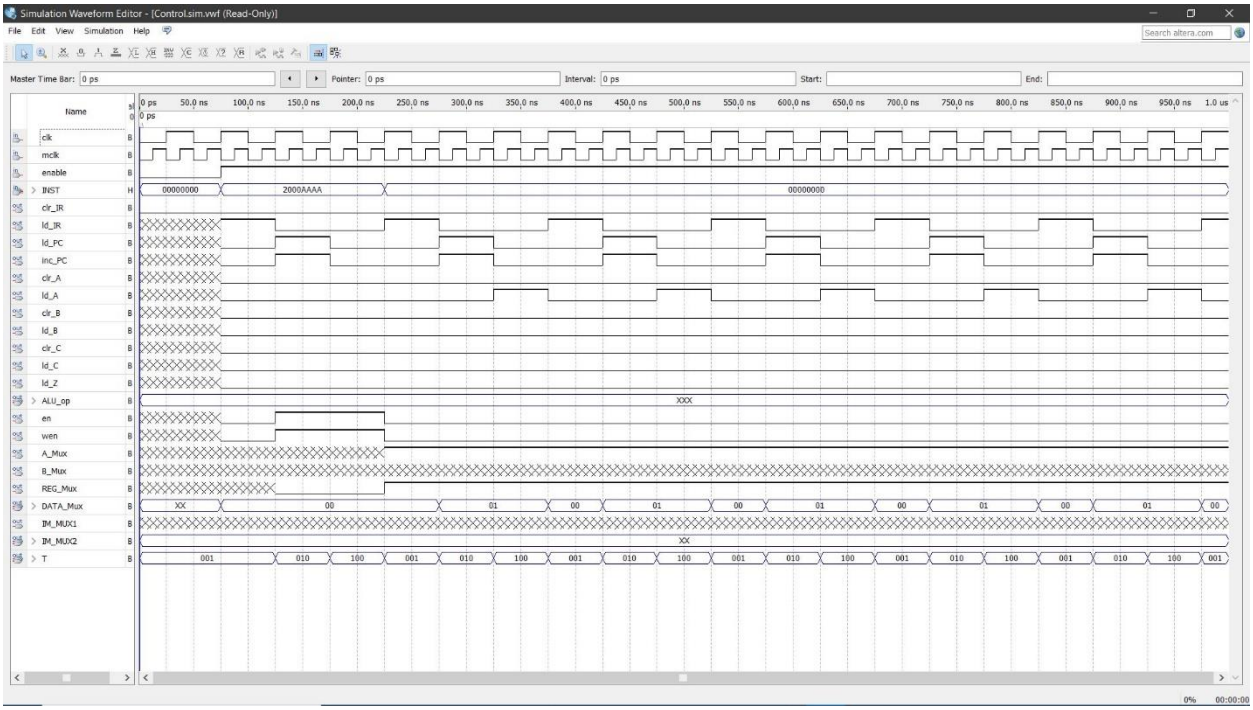
1. LDAI



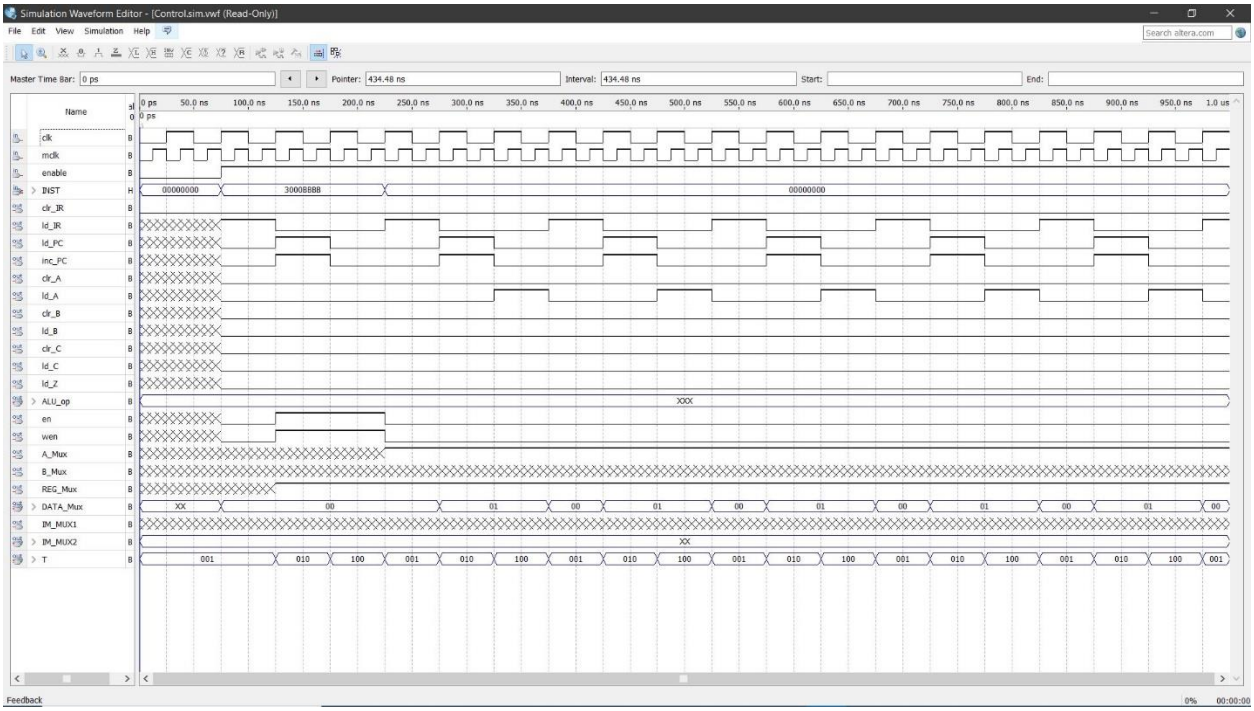
2. LDBI


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 163.9 ns
Interval: 163.9 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
10008888
00000000
XXX
XX

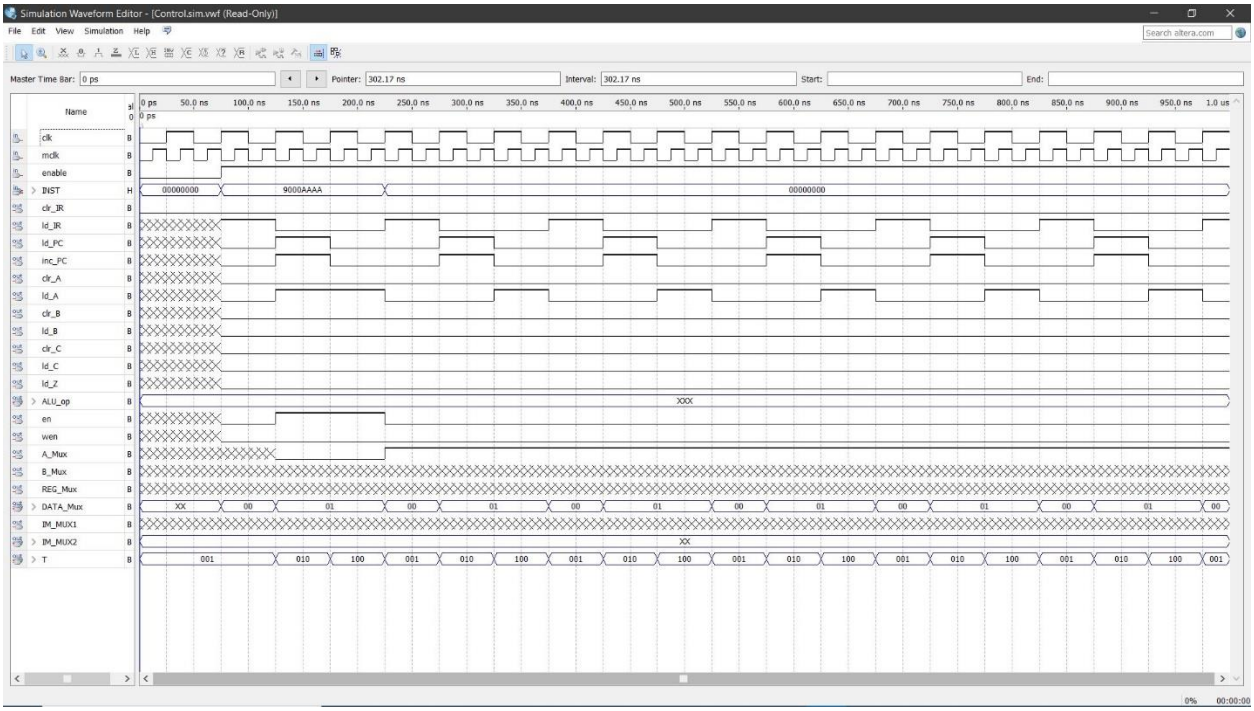
3. STA


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 0 ps
Interval: 0 ps
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
2000AAAA
00000000
XXX
XX

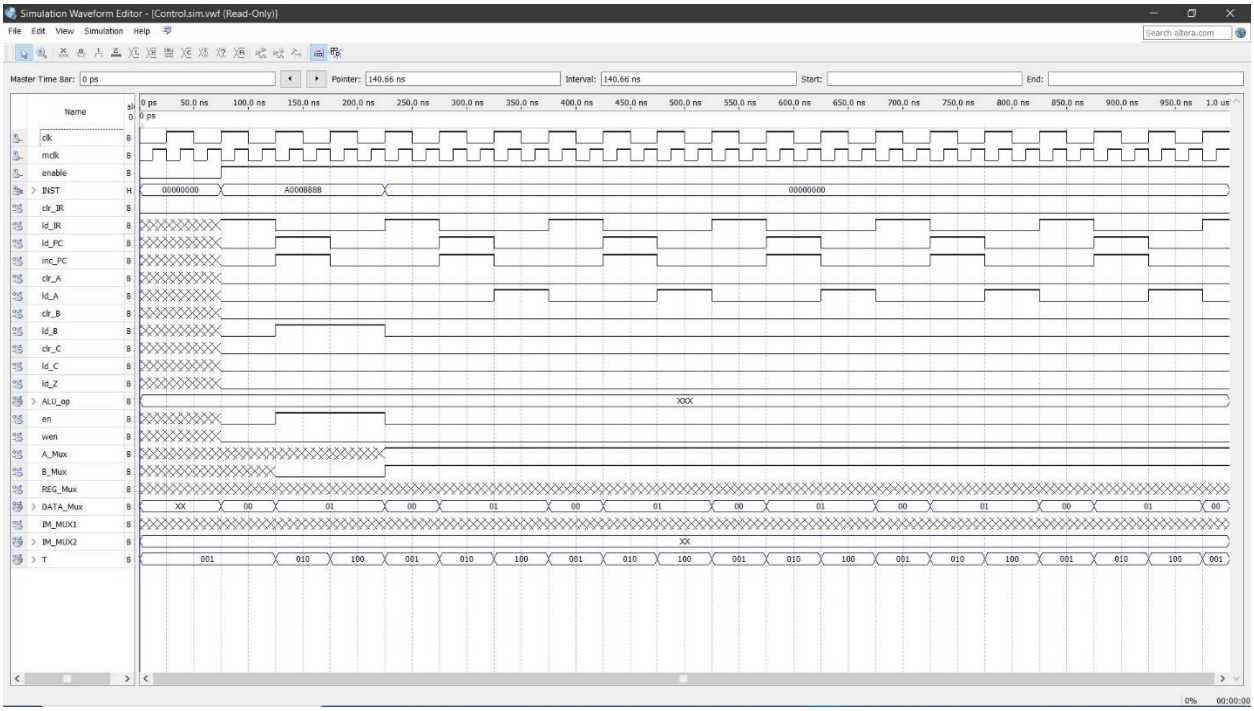
4. STB


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 434.48 ns
Interval: 434.48 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
30008BB8
00000000
XXX
XX
Feedback

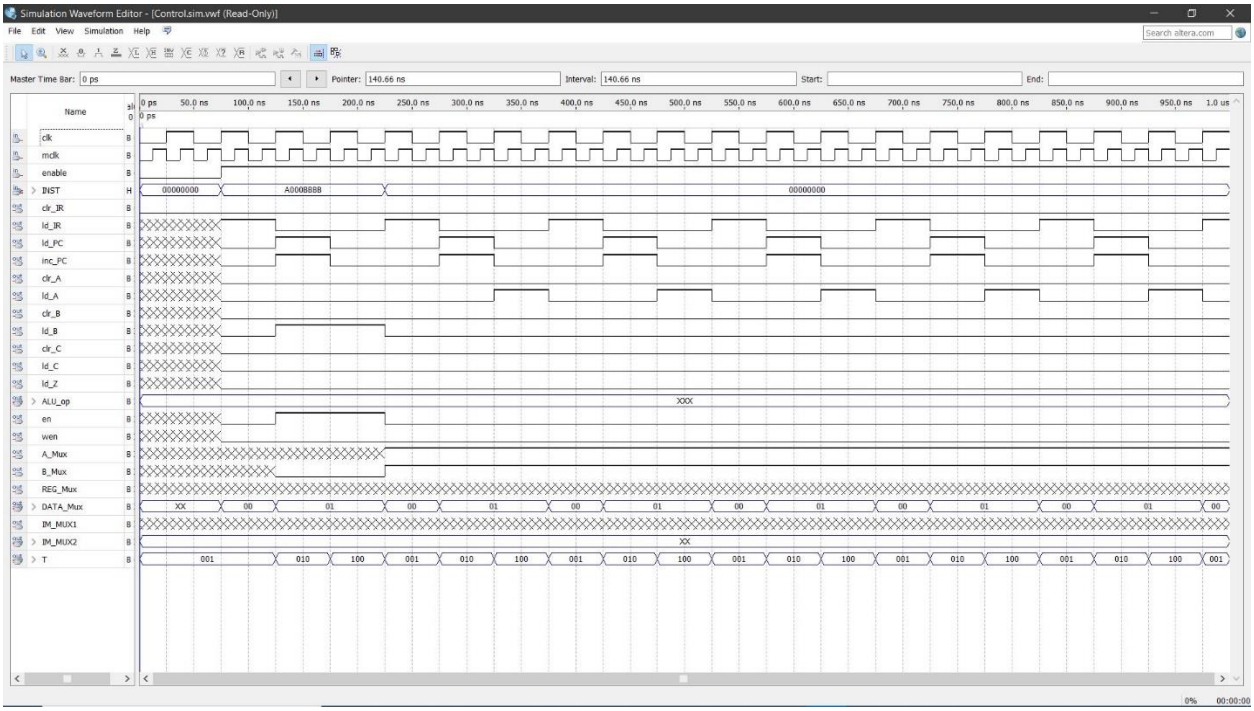
5. LDA


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 302.17 ns
Interval: 302.17 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
9000AAAA
00000000
XXX
XX

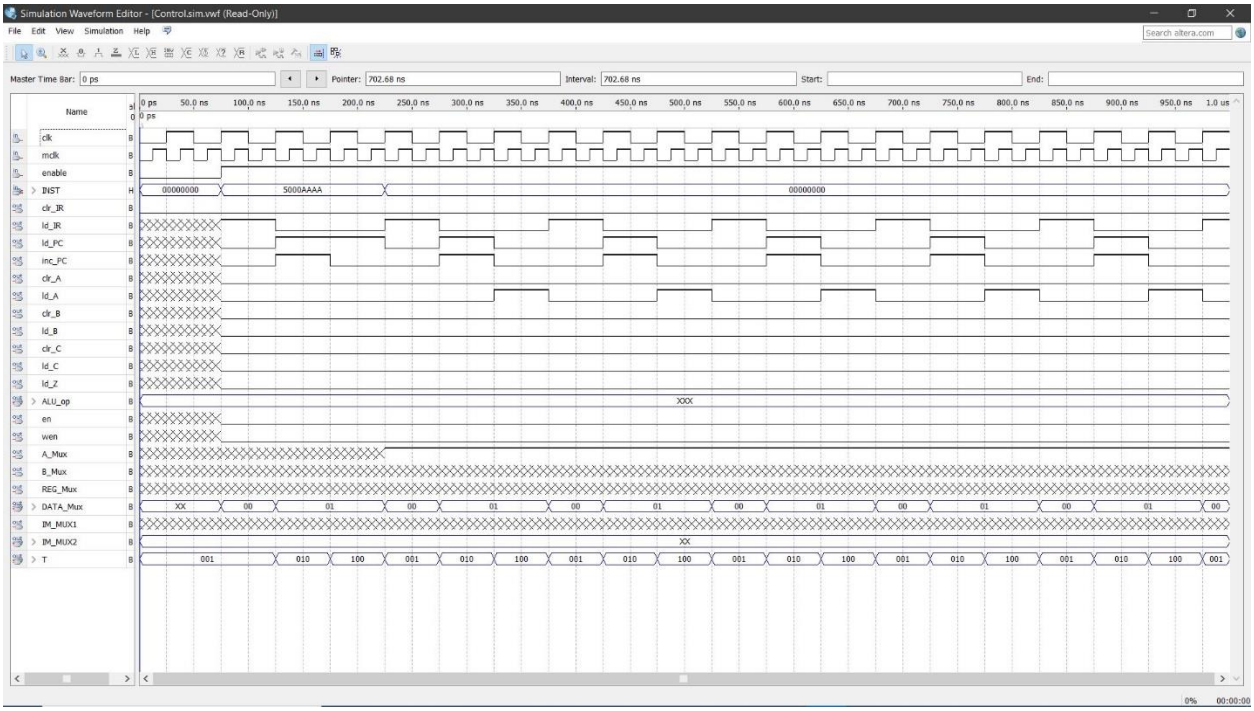
6. LDB


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 140.66 ns
Interval: 140.66 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
A0008888
00000000
XXX
XX

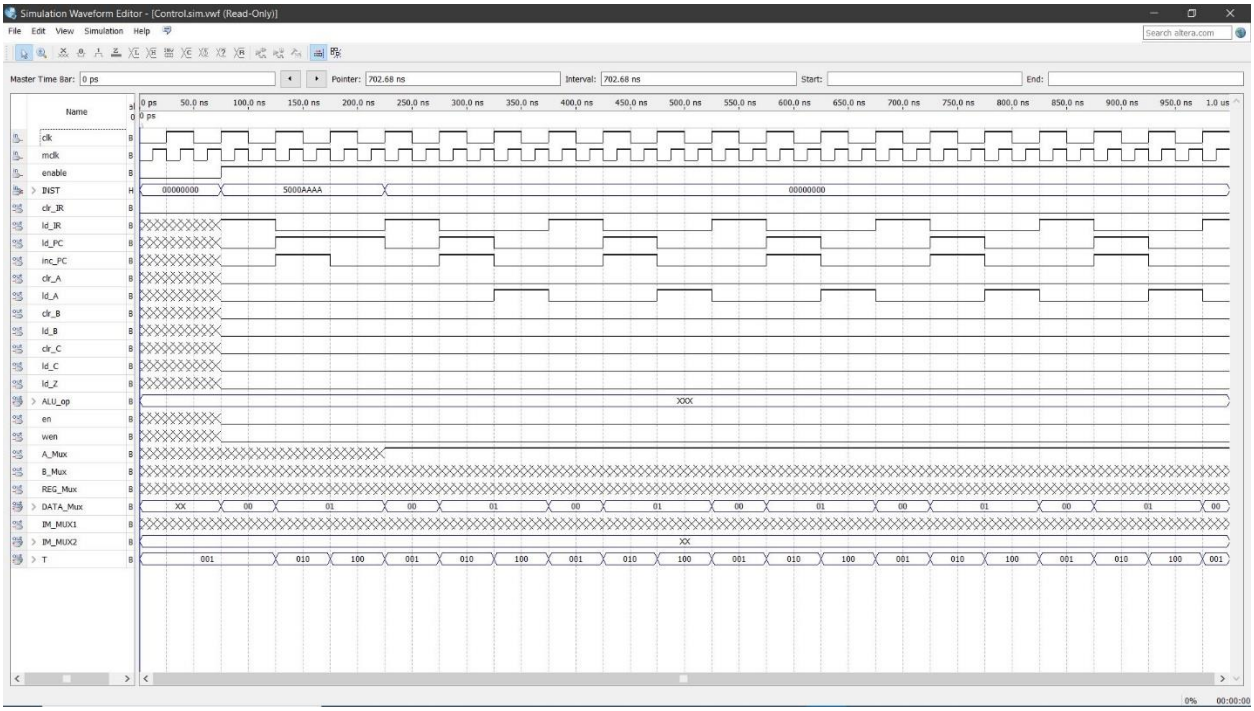
7. LUI


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 140.66 ns
Interval: 140.66 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
A0008888
00000000
XXX
XX

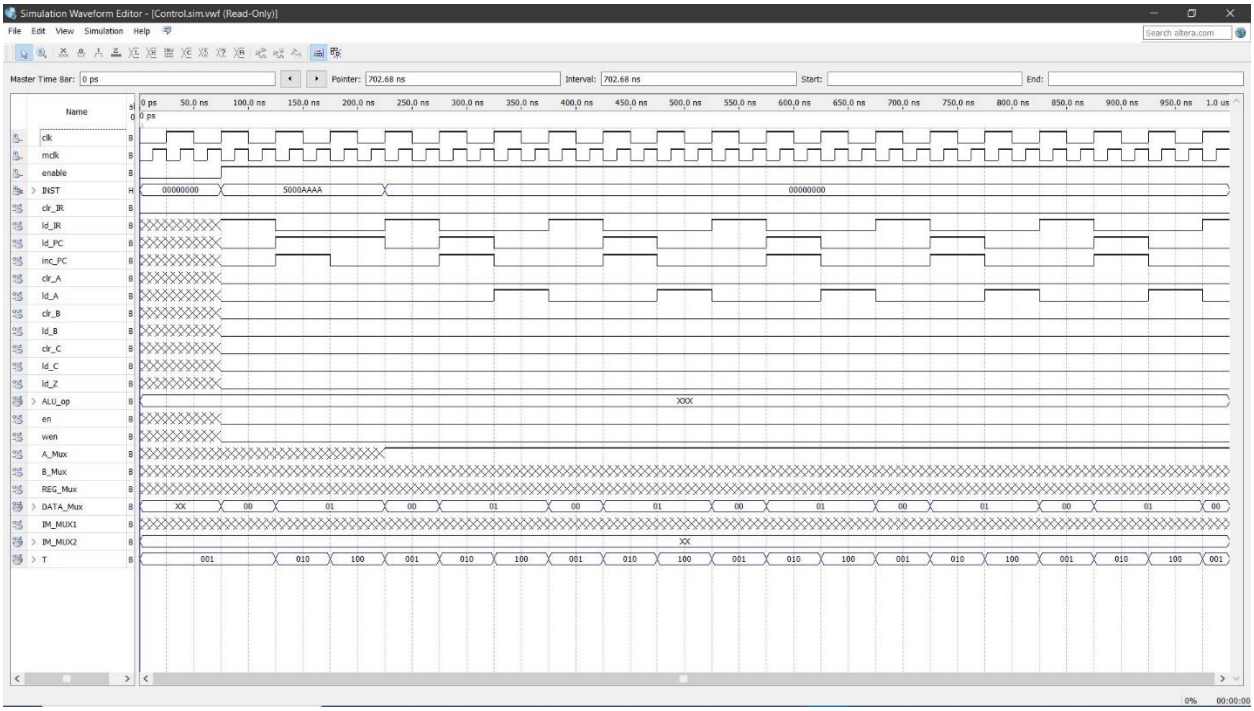
8. JMP


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 702.68 ns
Interval: 702.68 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
5000AAAA
00000000
XXX
XX

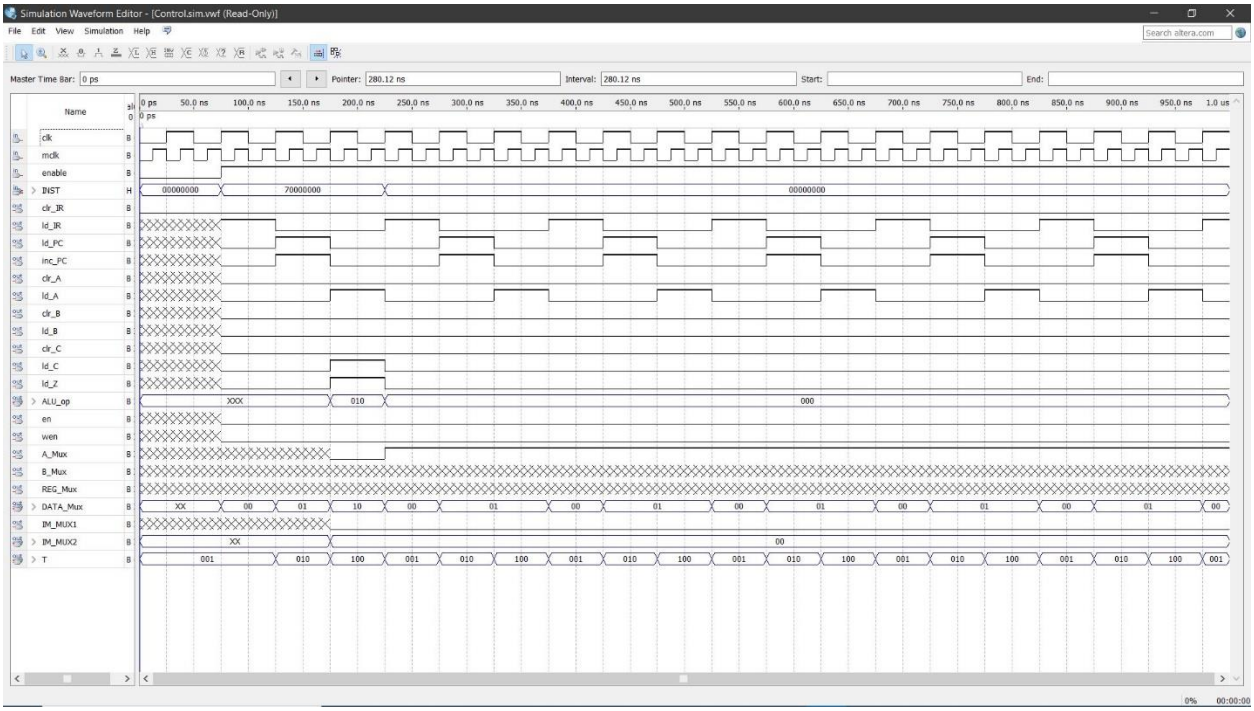
9. BEQ


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 702.68 ns
Interval: 702.68 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
5000AAAA
00000000
XXX
XX

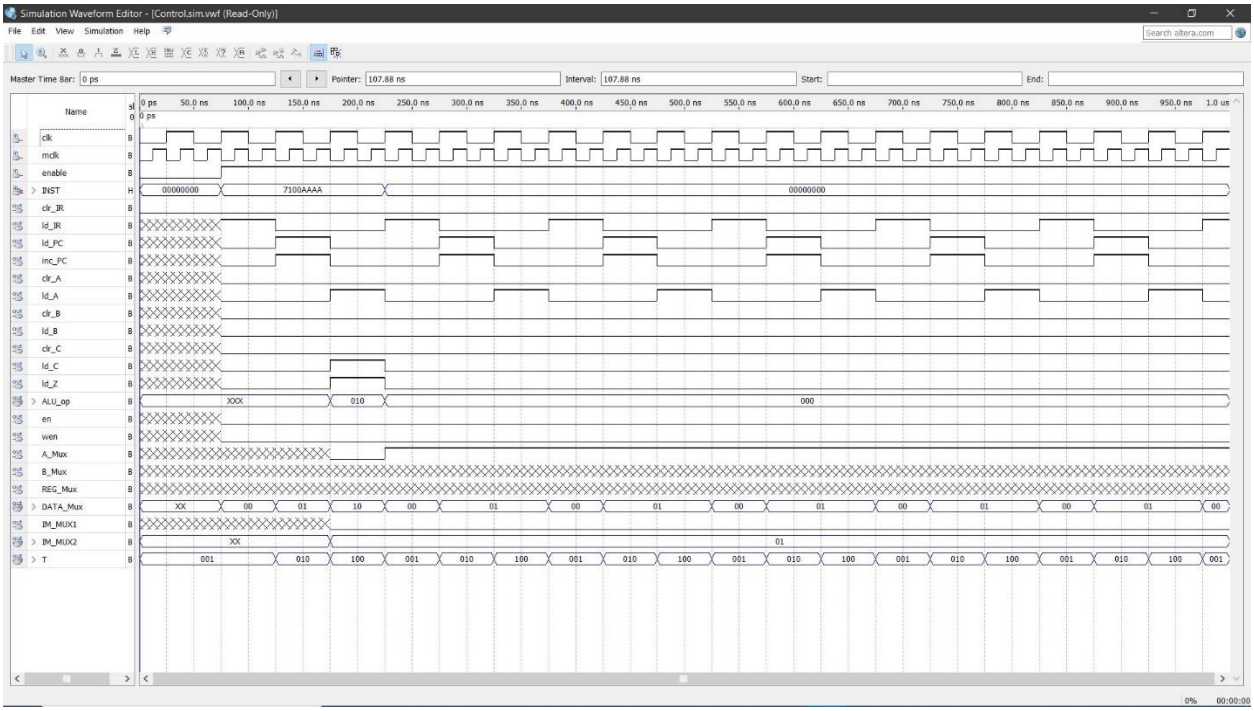
10. BNE


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 702.68 ns
Interval: 702.68 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
5000AAAA
00000000
XXX
XX

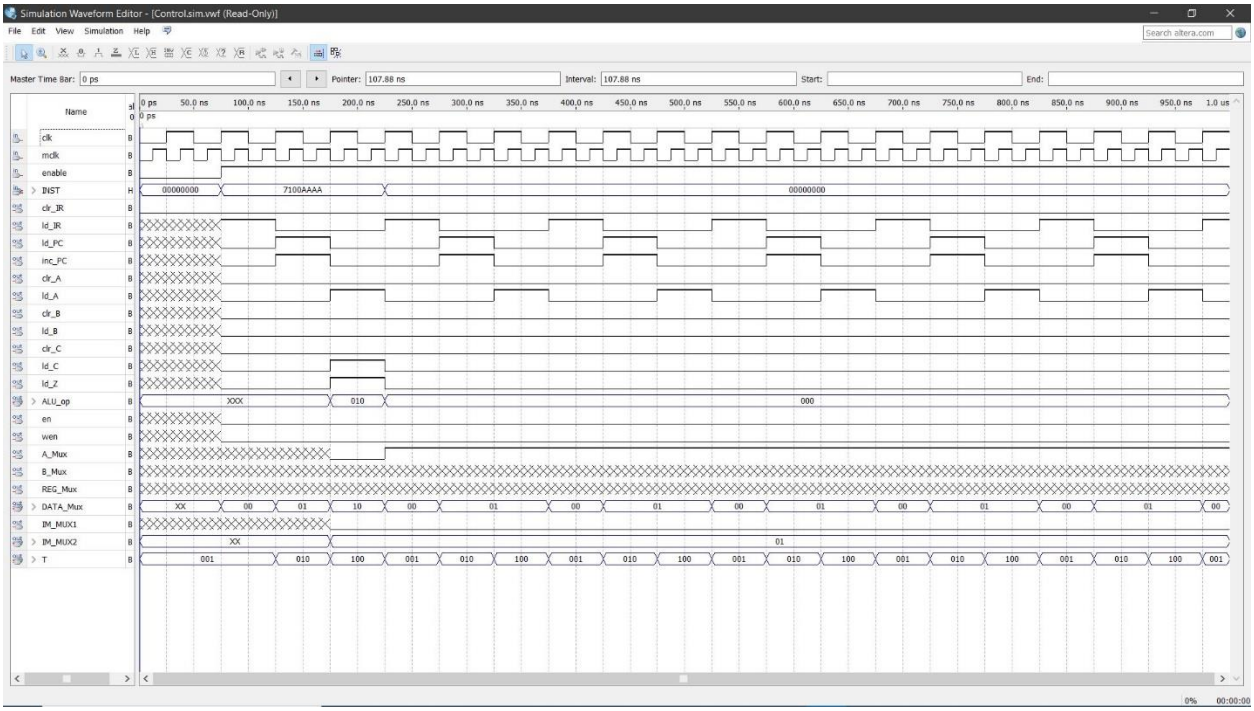
11. ADD


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 280.12 ns
Interval: 280.12 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

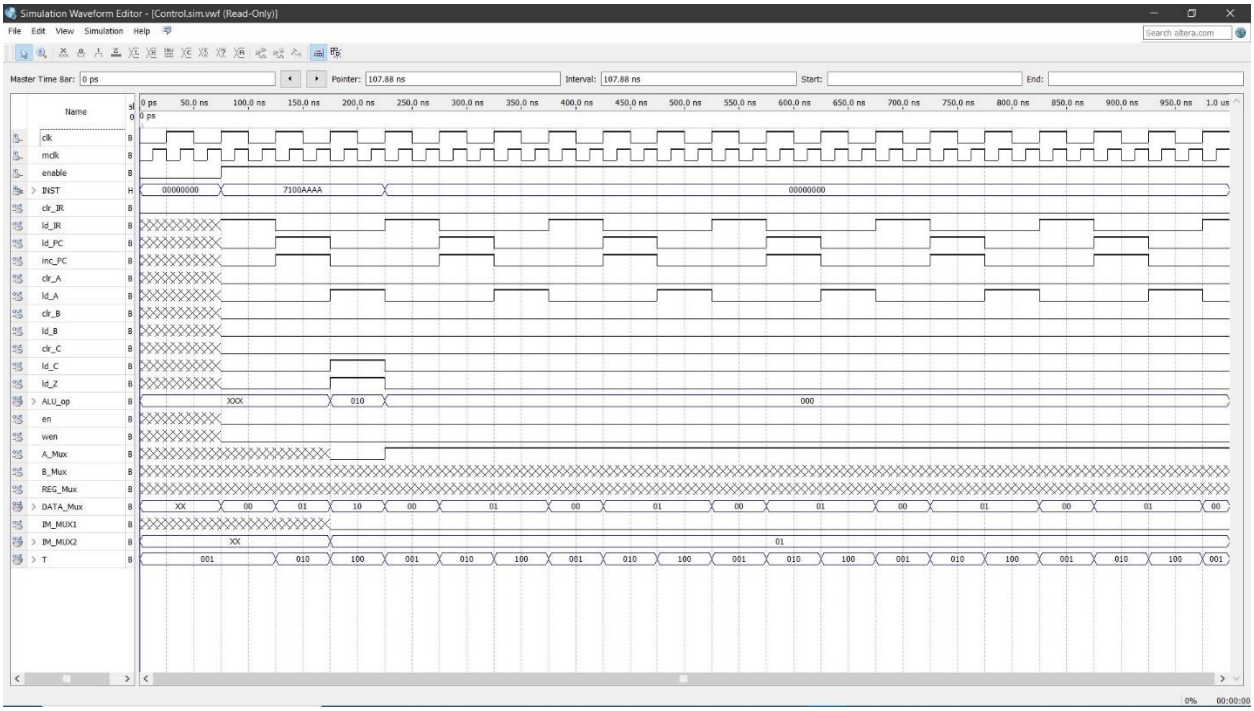
12. ADDI


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 107.88 ns
Interval: 107.88 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
7100AAAA
00000000
XXX
010
000

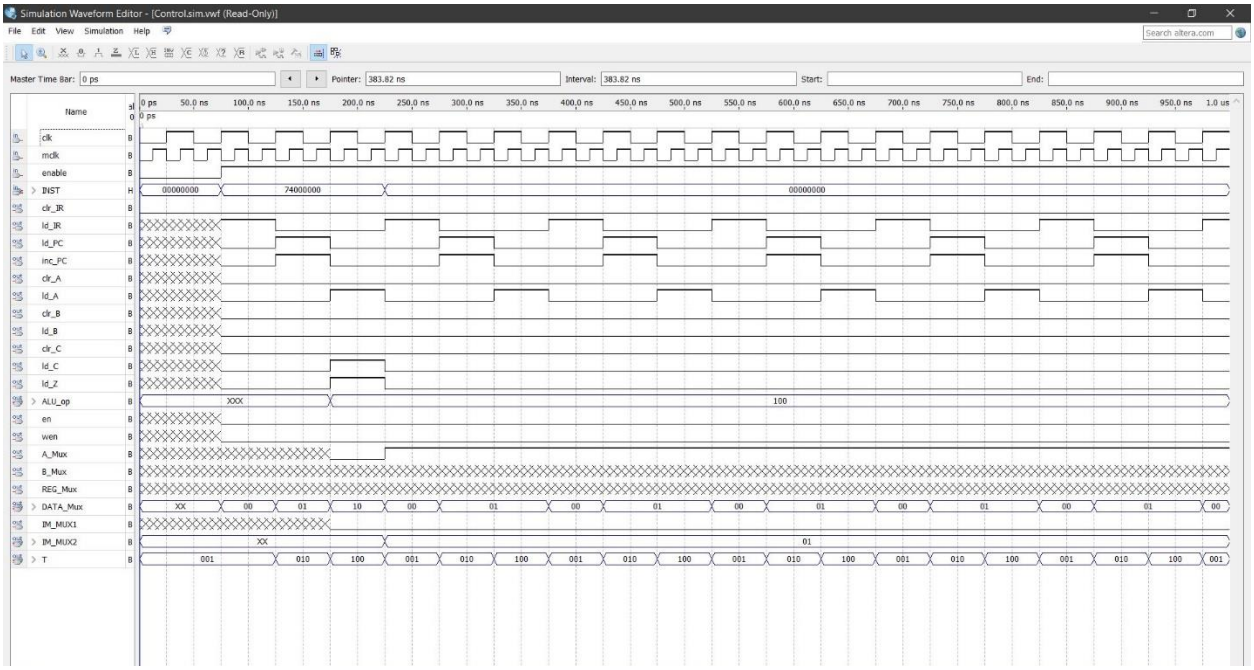
13. SUB


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 107.88 ns
Interval: 107.88 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
7100AAAA
00000000
XXX
010
000
XX
00
01
10
XX
01
001
010
100

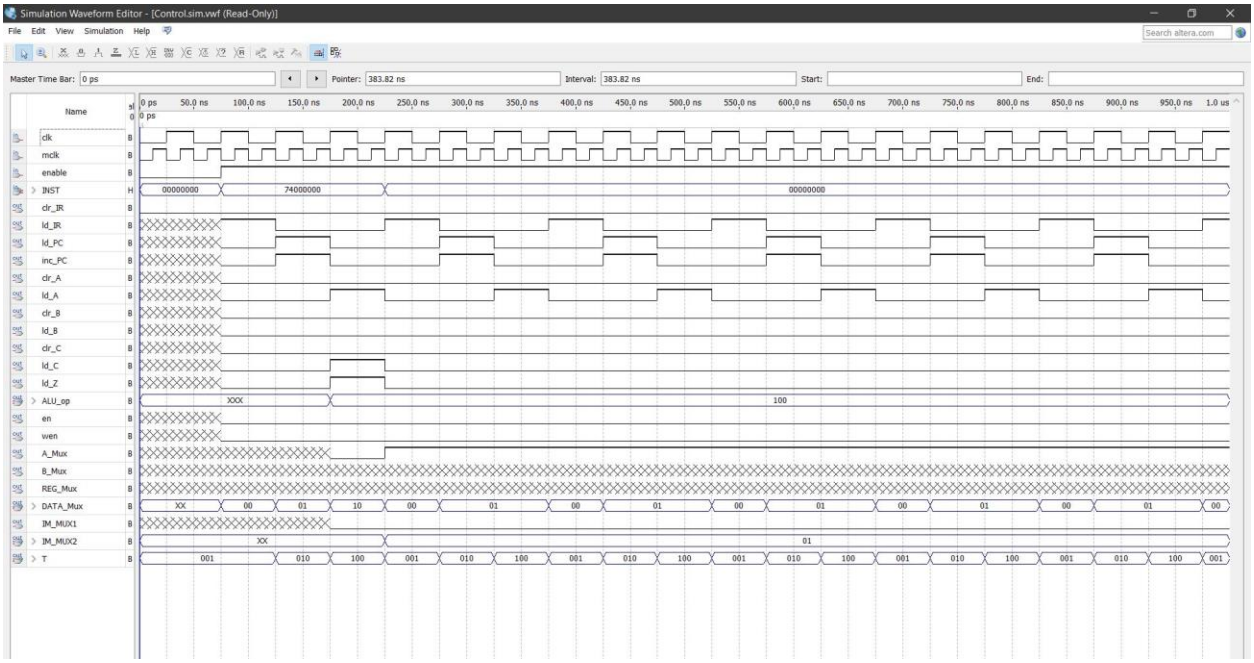
14. INCA


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 107.88 ns
Interval: 107.88 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
7100AAAA
00000000
XXX
010
000
XX
00
01
10
XX
01
001
010
100

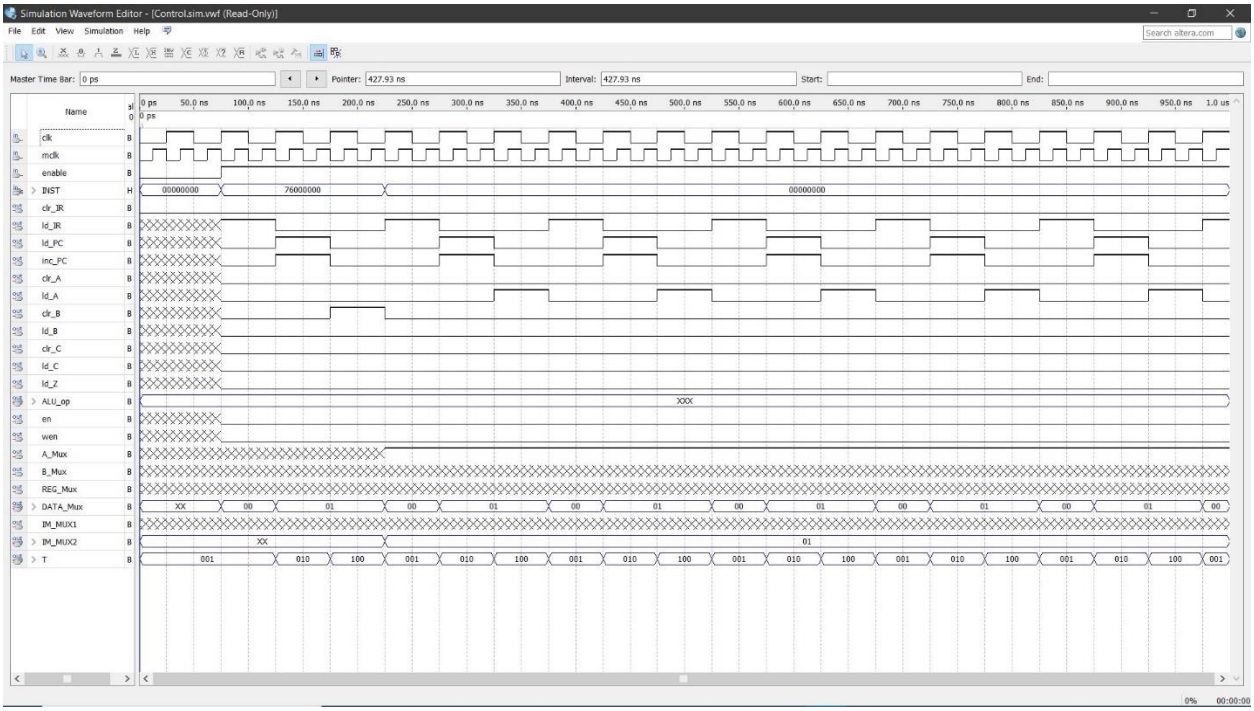
15. ROL


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 383.82 ns
Interval: 383.82 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

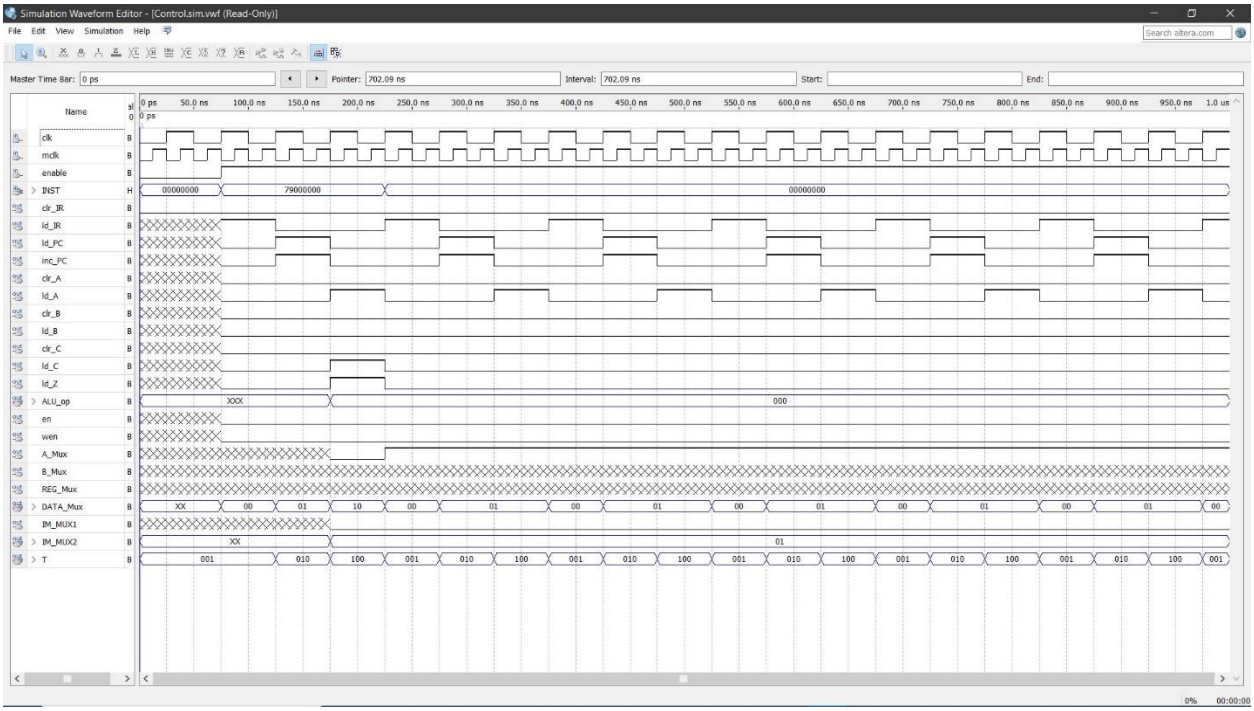
16. CLRA


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 383.82 ns
Interval: 383.82 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

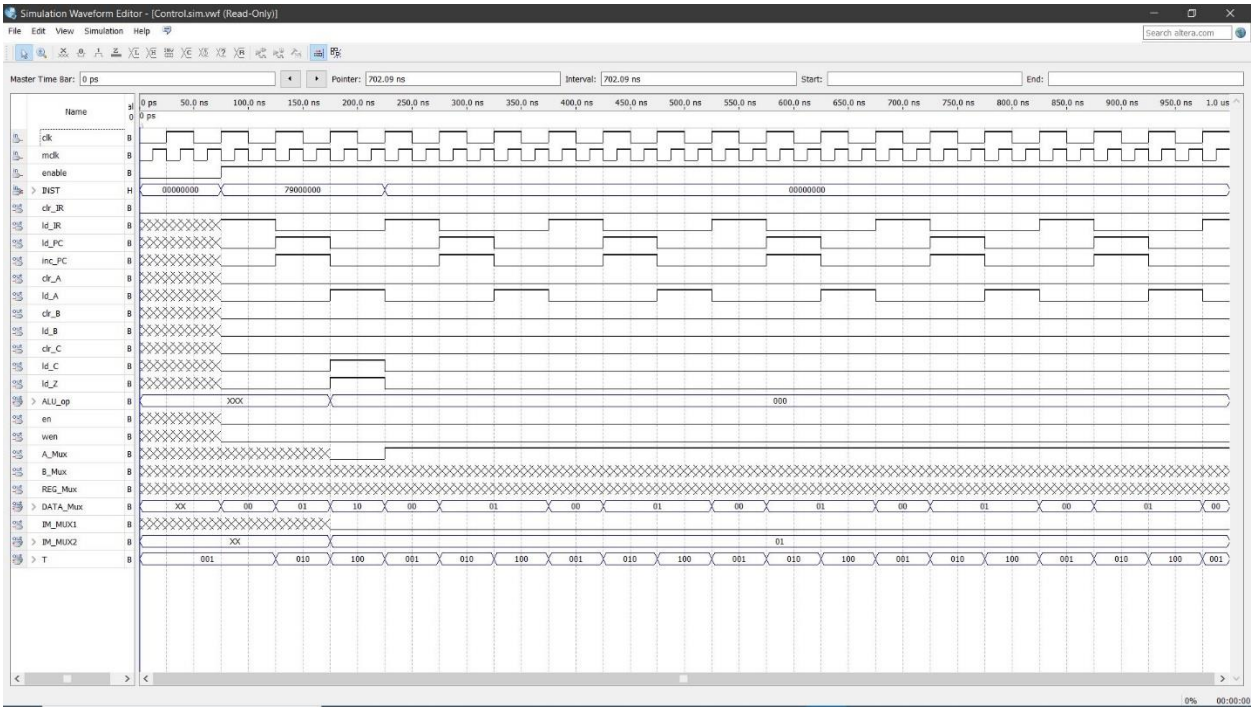
17. CLRB


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 427.93 ns
Interval: 427.93 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
76000000
00000000
XXX

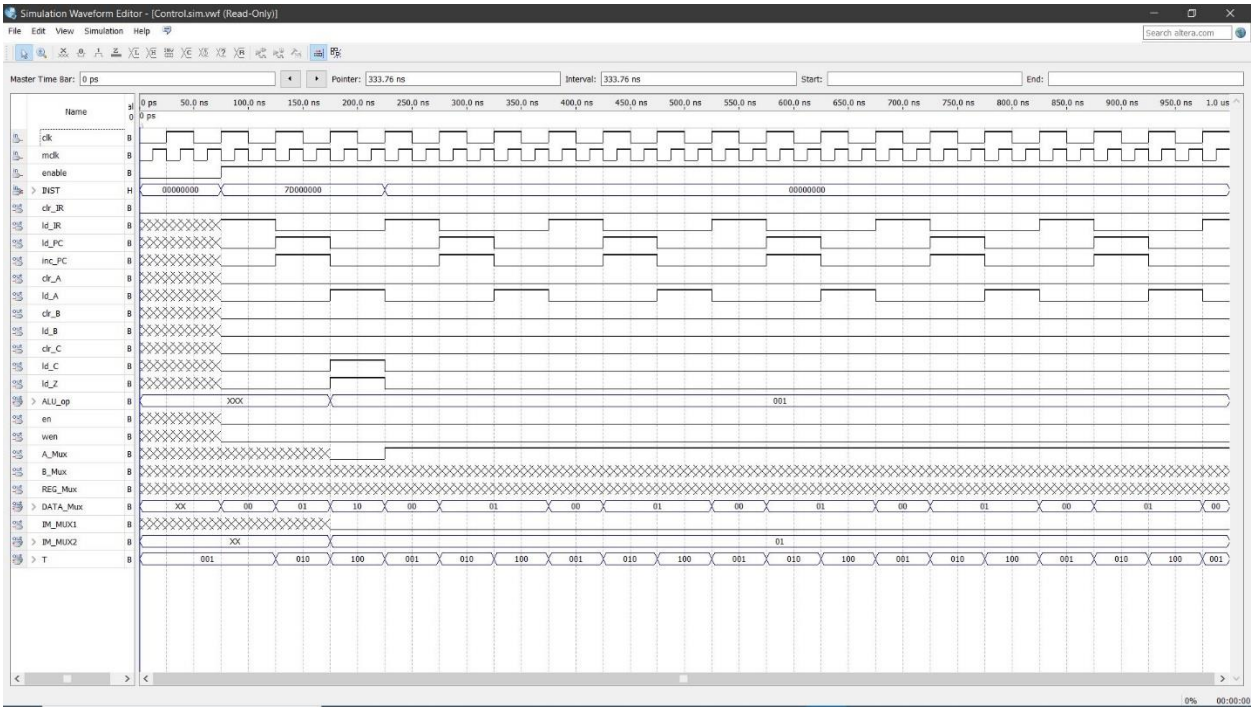
18. ANDI


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 702.09 ns
Interval: 702.09 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
79000000
00000000
XXX
000
XX
01
001
010
100

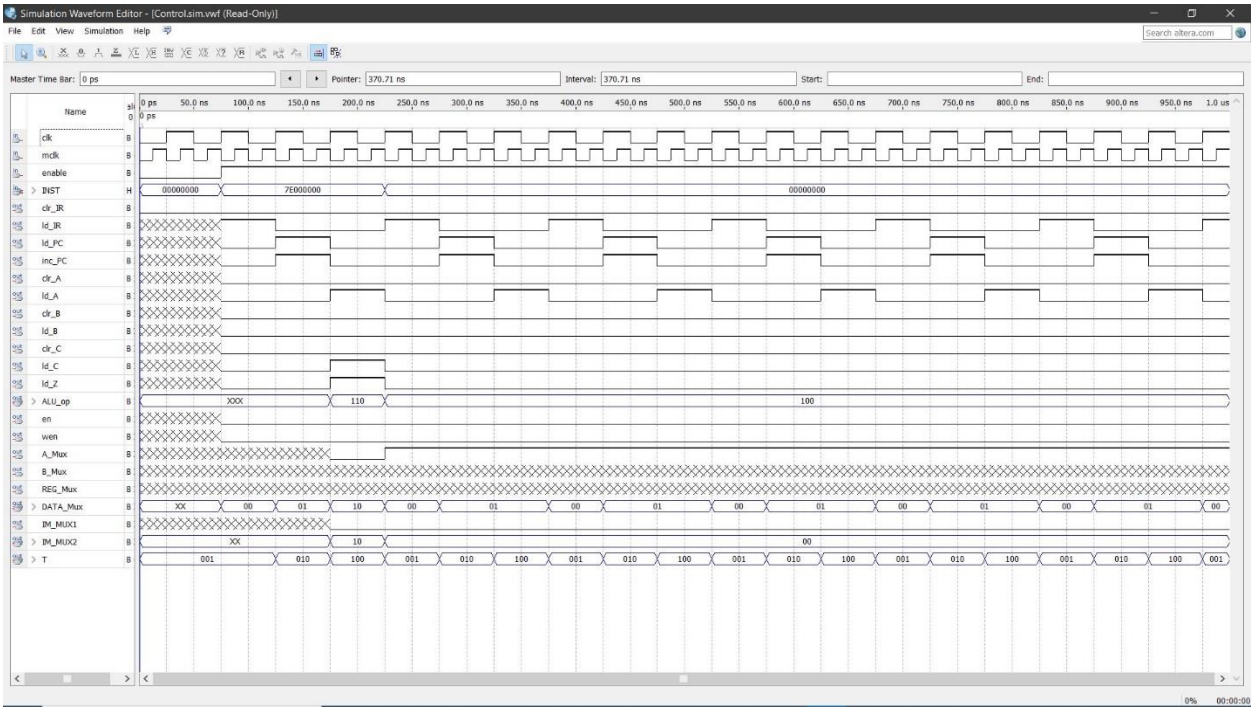
19. AND


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 702.09 ns
Interval: 702.09 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

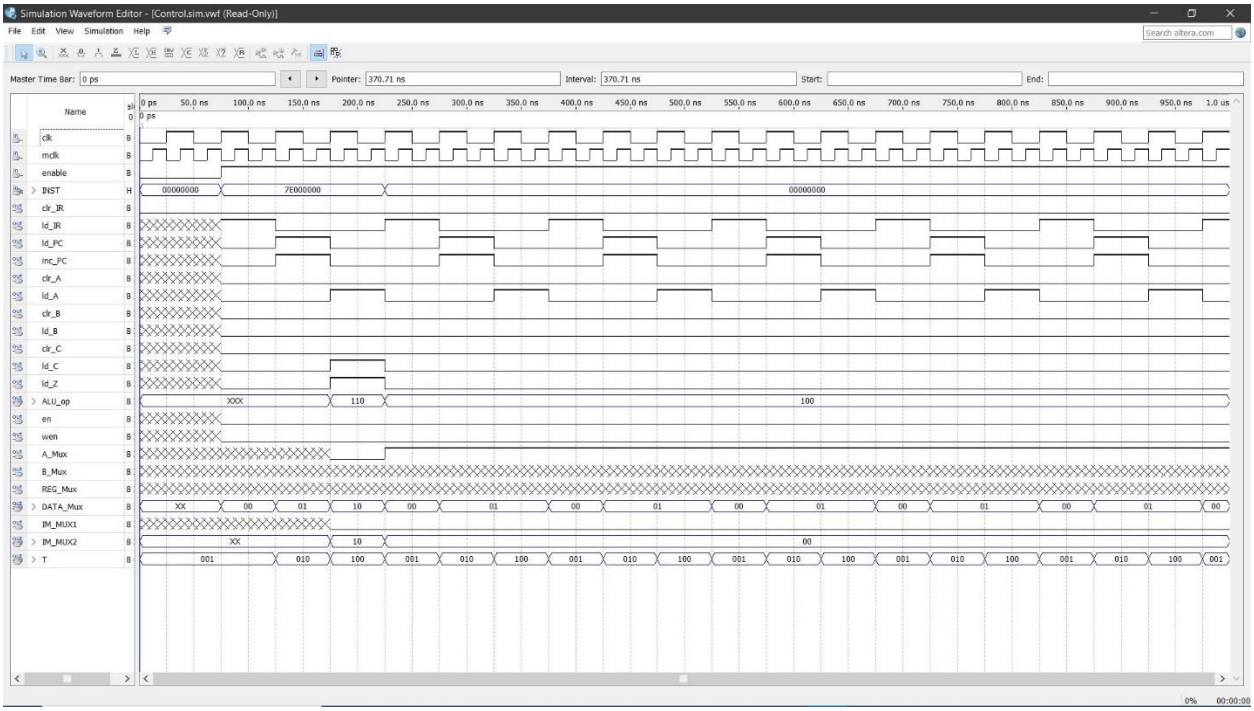
20. ORI


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 333.76 ns
Interval: 333.76 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

21. DECA


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 370.71 ns
Interval: 370.71 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T

22. ROR


Simulation Waveform Editor - [Control.sim.vwf (Read-Only)]
File Edit View Simulation Help
Master Time Bar: 0 ps
Pointer: 370.71 ns
Interval: 370.71 ns
Start:
End:
Name
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_Mux
DATA_Mux
IM_MUX1
IM_MUX2
T
00000000
7E000000
00000000
XXX
110
100
XX
00
01
10
XX
10
00
001
010
100