Lab 5: CPU Control Unit Design:

Introduction:

The objective of this lab(lab 5) is to successfully design a control unit to implement on the CPU to generate the appropriate control signals for the tasks to be running on the CPU data-path scheme(Lab 4B). The various signals controlled by the control unit include all of the MUXes, Registers and the ALU of the data-path scheme. The control unit takes status bits for "Zero" and "Carry" as inputs and takes the instruction. However, within the instruction, note that only the opcode(INST[31...28]) and function code(INST[27..24]) are taken, as only those are required to determine the task being executed(LDA, ADD, OR, etc.). As it will be seen from the waveforms, each instruction/task in a CPU data-path requires 3 clock cycles/stages (T0(Fetch), T1(Decode), and T2(Execute)).

The following component that is required for the Control Unit Design:

Implementation of the CPU Control Unit:

```
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
```

```
107                 elsif Instruction_sig = "1001" then -- LDA
108                    clr_IR <= '0';
109                    ld_IR <= '0';
110                    ld_PC <= '1';
111                    inc_PC <= '1';
112                    clr_A <= '0';
113                    ld_A <= '1';
114                    ld_B <= '0';
115                    clr_B <= '0';
116                    clr_C <= '0';
117                    ld_C <= '0';
118                    clr_Z <= '0';
119                    ld_Z <= '0';
120                    DATA_Mux <= "01";
121                    EN <= '1';
122                    WEN <= '0';
123
124                 elsif Instruction_sig = "1010" then -- LDB
125                    clr_IR <= '0';
126                    ld_IR <= '0';
127                    ld_PC <= '1';
128                    inc_PC <= '1';
129                    clr_A <= '0';
130                    ld_A <= '0';
131                    ld_B <= '1';
132                    clr_B <= '0';
133                    clr_C <= '0';
134                    ld_C <= '0';
135                    clr_Z <= '0';
136                    ld_Z <= '0';
137                    REG_Mux <= '0';
138                    DATA_Mux <= "01";
139                    EN <= '1';
140                    WEN <= '0';
141                 end if; -- END IF FOR LOAD STORE IN STAGE 1
142
143              elsif present_state = state_2 then
144
145                 if Instruction_sig = "0101" then -- JUMP
146                    clr_IR <= '0';
147                    ld_IR <= '0';
148                    ld_PC <= '1';
149                    inc_PC <= '0';
150                    clr_A <= '0';
151                    ld_A <= '0';
152                    ld_B <= '0';
153                    clr_B <= '0';
154                    clr_C <= '0';
155                    ld_C <= '0';
156                    clr_Z <= '0';
157                    ld_Z <= '0';
158
159                 elsif Instruction_sig = "0110" then -- BEQ
160                    clr_IR <= '0';
161                    ld_IR <= '0';
162                    ld_PC <= '1';
163                    inc_PC <= '0';
164                    clr_A <= '0';
165                    ld_A <= '0';
166                    ld_B <= '0';
167                    clr_B <= '0';
168                    clr_C <= '0';
169                    ld_C <= '0';
170                    clr_Z <= '0';
171                    ld_Z <= '0';
172
173                 elsif Instruction_sig = "1000" then -- BNE
174                    clr_IR <= '0';
175                    ld_IR <= '0';
176                    inc_PC <= '0';
177                    clr_A <= '0';
178                    ld_A <= '0';
179                    ld_B <= '0';
```

```
                clr_B <= '0';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';

            elsif Instruction_sig = "1001" then -- LDA
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '0';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '1';
                ld_B <= '0';
                clr_B <= '0';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';
                B_Mux <= '0';
                DATA_Mux <= "01";
                EN <= '1';
                WEN <= '0';

            elsif Instruction_sig = "1010" then -- LDB
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <='0';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '0';
                ld_B <= '1';
                clr_B <= '0';
                clr_C <= '0';
                ld_C <='0';
                clr_Z <= '0';
                ld_Z <= '0';
                B_Mux <= '0';
                DATA_MUX <= "01";
                EN <= '1';
                WEN <= '0';

            elsif Instruction_sig = "0010" then -- STA
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '0';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '0';
                ld_B <= '0';
                clr_B <= '0';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';
                REG_Mux <= '0';
                DATA_Mux<="00";
                EN <= '1';
                WEN <= '1';

            elsif Instruction_sig = "0011" then -- STB
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '0';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '0';
                ld_B <= '0';
                clr_B <= '0';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';
                clr_Z <= '0';
```

```
254                     ld_Z <= '0';
255                     REG_Mux <= '1';
256                     DATA_Mux <= "00";
257                     en <= '1';
258                     wen <= '1';
259 
260                 elsif Instruction_sig = "0000" then -- LDAI
261                     clr_IR <= '0';
262                     ld_IR <= '0';
263                     ld_PC <= '0';
264                     inc_pc <= '0';
265                     clr_A <= '0';
266                     ld_A <= '1';
267                     ld_B <= '0';
268                     clr_B <= '0';
269                     clr_C <= '0';
270                     ld_C <= '0';
271                     clr_Z <= '0';
272                     ld_Z <= '0';
273                     ALU_op <= "000";
274                     A_Mux <= '1';
275 
276                 elsif Instruction_sig = "0001" then -- LDBI
277                     clr_IR <= '0';
278                     ld_IR <= '0';
279                     ld_PC <= '0';
280                     inc_pc <= '0';
281                     clr_A <= '0';
282                     ld_A <= '0';
283                     ld_B <= '1';
284                     clr_B <= '0';
285                     clr_C <= '0';
286                     ld_C <= '0';
287                     clr_Z <= '0';
288                     ld_Z <= '0';
289                     B_Mux <= '1';
290 
291                 elsif Instruction_sig = "0100" then -- LUI
292                     clr_IR <= '0';
293                     ld_IR <= '0';
294                     ld_PC <= '0';
295                     inc_pc <= '0';
296                     clr_A <= '0';
297                     ld_A <= '1';
298                     ld_B <= '0';
299                     clr_B <= '1';
300                     clr_C <= '0';
301                     ld_C <= '0';
302                     clr_Z <= '0';
303                     ld_Z <= '0';
304                     ALU_op <= "001";
305                     A_Mux <= '1';
306                     DATA_Mux <= "10";
307                     IM_MUX1 <= '1';
308                 elsif Instruction_sig2 = "01111001" then --ANDI
309                     clr_IR <= '0';
310                     ld_IR <= '0';
311                     ld_PC <= '0';
312                     inc_PC <= '0';
313                     clr_A <= '0';
314                     ld_A <= '1';
315                     ld_B <= '0';
316                     clr_B <= '0';
317                     clr_C <= '0';
318                     ld_C <= '1';
319                     clr_Z <= '0';
320                     ld_Z <= '1';
321                     ALU_op <= "000";
322                     A_Mux <= '0';
323                     DATA_Mux <= "10";
324                     IM_MUX1 <= '0';
325                     IM_MUX2 <= "01";
326                 elsif Instruction_sig2 = "01111110" then -- DECA
327                     clr_IR <= '0';
```

```
328                     ld_IR <= '0';
329                     ld_PC <= '0';
330                     inc_PC <= '0';
331                     clr_A <= '0';
332                     ld_A <= '1';
333                     ld_B <= '0';
334                     clr_B <= '0';
335                     clr_C <= '0';
336                     ld_C <= '1';
337                     clr_Z <= '0';
338                     ld_Z <= '1';
339                     ALU_op <= "110";
340                     A_Mux <= '0';
341                     DATA_Mux <= "10";
342                     IM_MUX1 <= '0';
343                     IM_MUX2 <= "10";
344                 elsif Instruction_sig2 = "01110000" then -- ADD
345                     clr_IR <= '0';
346                     ld_IR <= '0';
347                     ld_PC <= '0';
348                     inc_PC <= '0';
349                     clr_A <= '0';
350                     ld_A <= '1';
351                     ld_B <= '0';
352                     clr_B <= '0';
353                     clr_C <= '0';
354                     ld_C <= '1';
355                     clr_Z <= '0';
356                     ld_Z <= '1';
357                     ALU_op <= "010";
358                     A_Mux <= '0';
359                     DATA_Mux <= "10";
360                     IM_MUX1 <= '0';
361                     IM_MUX2 <= "00";
362 
363                 elsif Instruction_sig2 = "01110010" then --SUB
364                     clr IR <= '0';
365                     ld_IR <= '0';
366                     ld_PC <= '0';
367                     inc_PC <= '0';
368                     clr_A <= '0';
369                     ld_A <= '1';
370                     ld_B <= '0';
371                     clr_B <= '0';
372                     clr_C <= '0';
373                     ld_C <= '1';
374                     clr_Z <= '0';
375                     ld_Z <= '1';
376                     ALU_op <= "110";
377                     A_Mux <= '0';
378                     DATA_Mux <= "10";
379                     IM_MUX1 <= '0';
380                     IM_MUX2 <= "00";
381                 elsif Instruction_sig2 = "01110011" then --INCA
382                     clr_IR <= '0';
383                     ld_IR <= '0';
384                     ld_PC <= '0';
385                     inc_PC <= '0';
386                     clr_A <= '0';
387                     ld_A <= '1';
388                     ld_B <= '0';
389                     clr_B <= '0';
390                     clr_C <= '0';
391                     ld_C <= '1';
392                     clr_Z <= '0';
393                     ld_Z <= '1';
394                     ALU_op <= "010";
395                     A_Mux <= '0';
396                     DATA_Mux <= "10";
397                     IM_MUX1 <= '0';
398                     IM_MUX2 <= "10";
399                 elsif Instruction_sig2 = "01111011" then -- AND
400                     clr_IR <= '0';
401                     ld_IR <= '0';
```

```
                    ld_PC <= '0';
                    inc_PC <= '0';
                    clr_A <= '0';
                    ld_A <= '1';
                    ld_B <= '0';
                    clr_B <= '0';
                    clr_C <= '0';
                    ld_C <= '1';
                    clr_Z <= '0';
                    ld_Z <= '1';
                    ALU_op <= "000";
                    A_Mux <= '0';
                    DATA_Mux <= "10";
                    IM_MUX1 <= '0';
                    IM_MUX2 <= "00";
                elsif Instruction_sig2 = "01110001" then -- ADDI
                    clr_IR <= '0';
                    ld_IR <= '0';
                    ld_PC <= '0';
                    inc_PC <= '0';
                    clr_A <= '0';
                    ld_A <= '1';
                    ld_B <= '0';
                    clr_B <= '0';
                    clr_C <= '0';
                    ld_C <= '1';
                    clr_Z <= '0';
                    ld_Z <= '1';
                    ALU_op <= "010";
                    A_Mux <= '0';
                    DATA_Mux <= "10";
                    IM_MUX1 <= '0';
                    IM_MUX2 <= "01";
                elsif Instruction_sig2 = "01111101" then -- ORI
                    clr_IR <= '0';
                    ld_IR <= '0';
                    ld_PC <= '0';
                    inc_PC <= '0';
                    clr_A <= '0';
                    ld_A <= '1';
                    ld_B <= '0';
                    clr_B <= '0';
                    clr_C <= '0';
                    ld_C <= '1';
                    clr_Z <= '0';
                    ld_Z <= '1';
                    ALU_op <= "001";
                    A_Mux <= '0';
                    DATA_Mux <= "10";
                    IM_MUX1 <= '0';
                    IM_MUX2 <= "01";
                elsif Instruction_sig2 = "01110100" then --ROL
                    clr_IR <= '0';
                    ld_IR <= '0';
                    ld_PC <= '0';
                    inc_PC <= '0';
                    clr_A <= '0';
                    ld_A <= '1';
                    ld_B <= '0';
                    clr_B <= '0';
                    clr_C <= '0';
                    ld_C <= '1';
                    clr_Z <= '0';
                    ld_Z <= '1';
                    ALU_op <= "100";
                    A_Mux <= '0';
                    DATA_Mux <= "10";
                    IM_MUX1 <= '0';

                elsif Instruction_sig2 = "01111111" then -- ROR
                    clr_IR <= '0';
                    ld_IR <= '0';
                    ld_PC <= '0';
                    inc_PC <= '0';
```

```
476                     clr_A <= '0';
477                     ld_A <= '1';
478                     ld_B <= '0';
479                     clr_B <= '0';
480                     clr_C <= '0';
481                     ld_C <= '1';
482                     clr_Z <= '0';
483                     ld_Z <= '1';
484                     ALU_op <= "101";
485                     A_Mux <= '0';
486                     DATA_Mux <= "10";
487                     IM_MUX1 <= '0';
488
489                 elsif Instruction_sig2 = "01110101" then -- CLR_A
490                     clr_IR <= '0';
491                     ld_IR <= '0';
492                     ld_PC <= '0';
493                     inc_PC <= '0';
494                     clr_A <= '1';
495                     ld_A <= '0';
496                     ld_B <= '0';
497                     clr_B <= '0';
498                     clr_C <= '0';
499                     ld_C <= '0';
500                     clr_Z <= '0';
501                     ld_Z <= '0';
502
503                 elsif Instruction_sig2 = "01110110" then --CLR_B
504                     clr_IR <= '0';
505                     ld_IR <= '0';
506                     ld_PC <= '0';
507                     inc_PC <= '0';
508                     clr_A <= '0';
509                     ld_A <= '0';
510                     ld_B <= '0';
511                     clr_B <= '1';
512                     clr_C <= '0';
513                     ld_C <= '0';
514                     clr_Z <= '0';
515                     ld_Z <= '0';
516
517                 elsif Instruction_sig2 = "01110111" then --CLR_C
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 <= '1';
527                     ld_C <= '0';
528                     clr_Z <= '0';
529                     ld_Z <= '0';
530
531                 elsif Instruction_sig2 = "01111000" then -- CLR_Z
532                     clr_IR <= '0';
533                     ld_IR <= '0';
534                     ld_PC <= '0';
535                     inc_PC <= '0';
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 <= '1';
543                     ld_Z <= '0';
544
545                 elsif Instruction_sig2 = "01111010" then --TSTZ
546                     if(statusZ = '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';
                    clr_B <= '0';
                    clr_C <= '0';
                    ld_C <= '0';
                    clr_Z <= '0';
                    ld_Z <= '0';
                end if;
            elsif Instruction_sig2 = "01111100" then --TSTC
                if(statusC = '1') then
                    clr_IR <= '0'; -- INCREMENT PC COUNTER
                    ld_IR <= '0';
                    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;
```

Figure 1: Code Implementation for the CPU Control Unit Design Component

Control Unit Functional Simulation Waveforms - Testing Portion:

1)

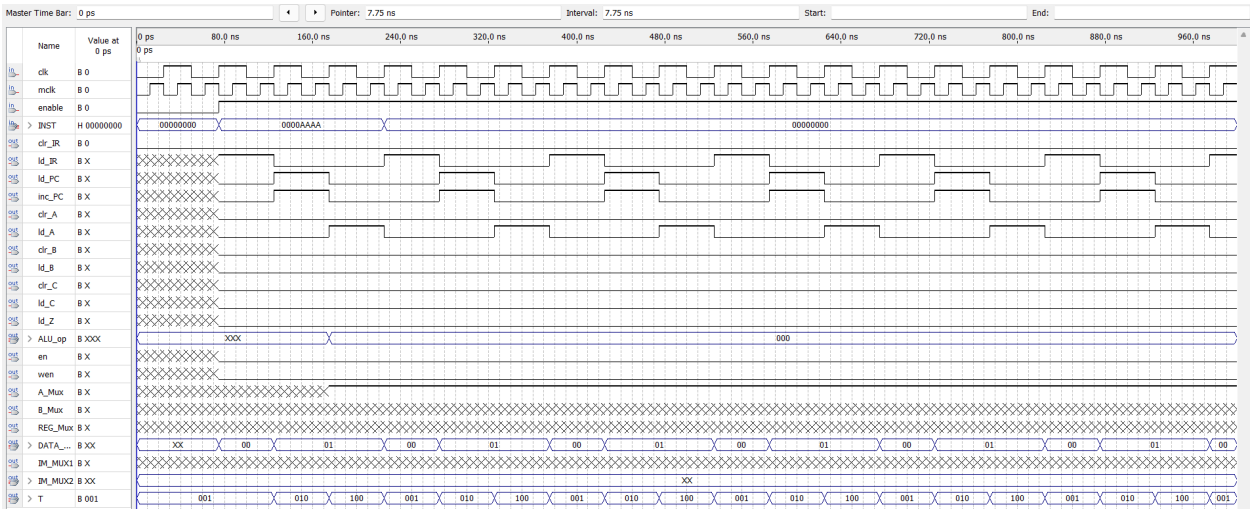

Figure 2: Waveform of the CPU Control Unit for operation: LDAI(Load Intermediate)

2)

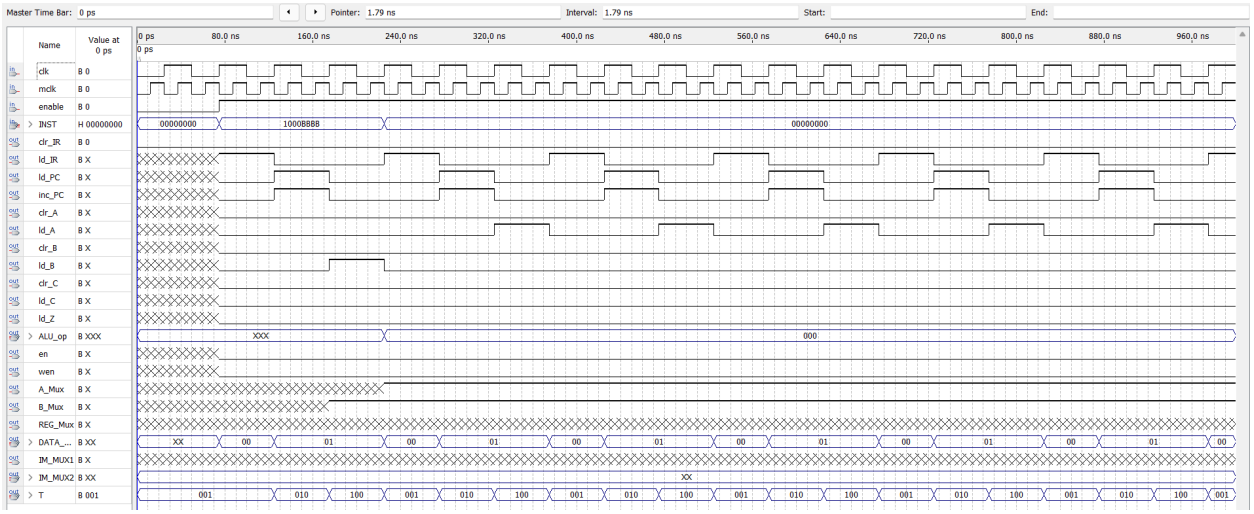

Figure 3: Waveform of the CPU Control Unit for operation: LDBI(Load Intermediate)

3)

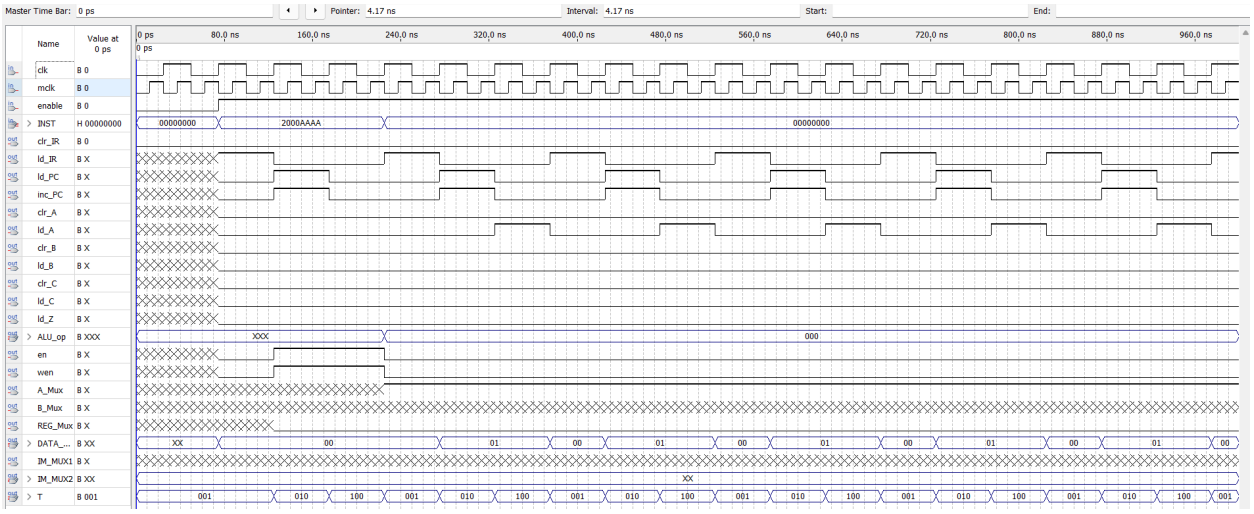

Figure 4: Waveform of the CPU Control Unit Component for operation: STA(Store from A)

4)

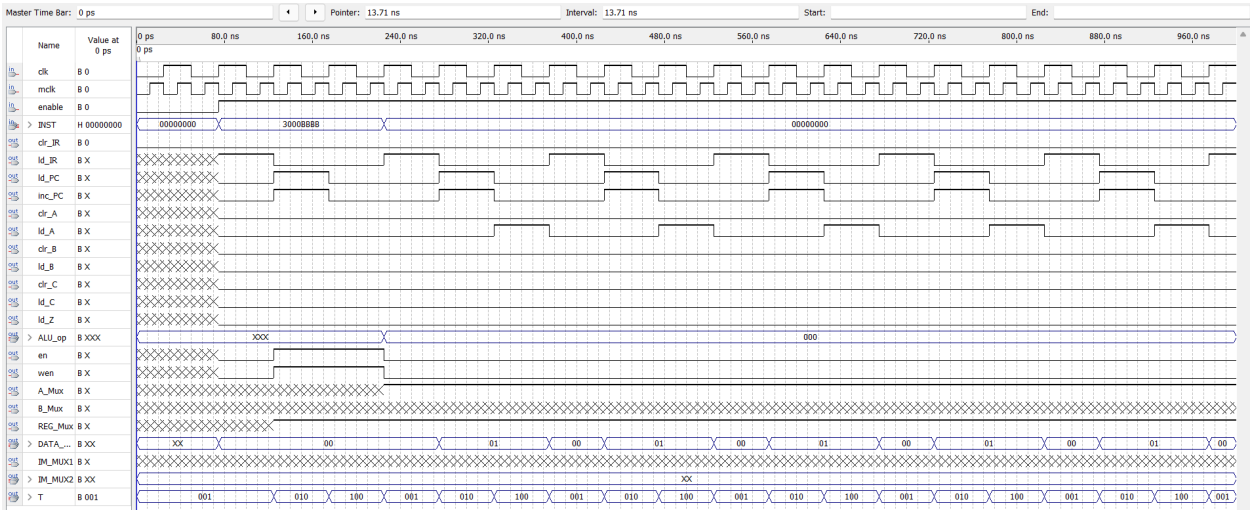

Figure 5: Waveform of the CPU Control Unit Component for operation: STB(Store from B)

5)

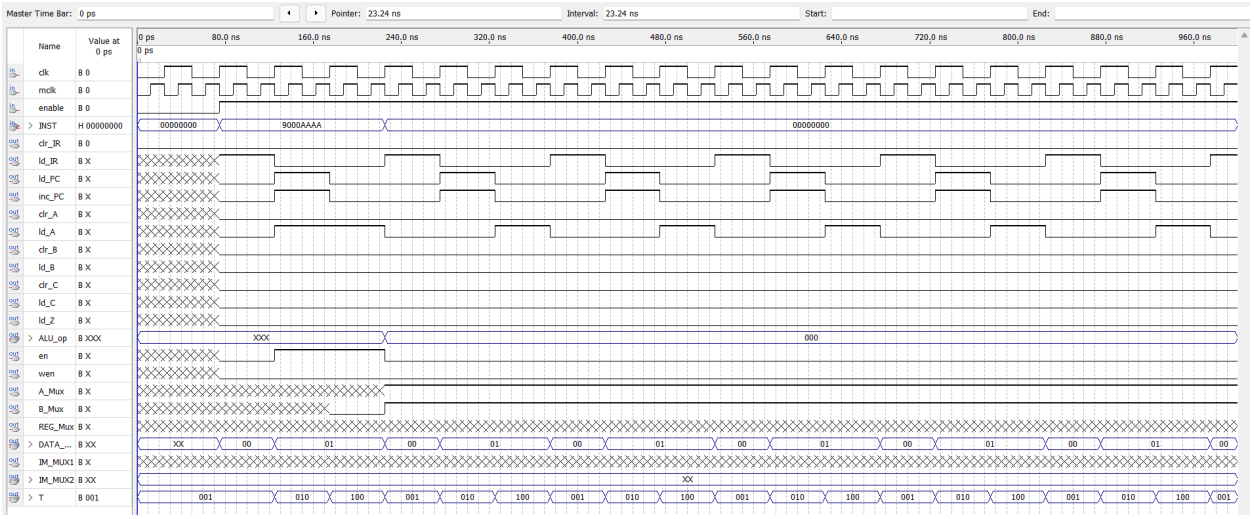

Figure 6: Waveform of the CPU Control Unit Component for operation: LDA(Load into A)

6)

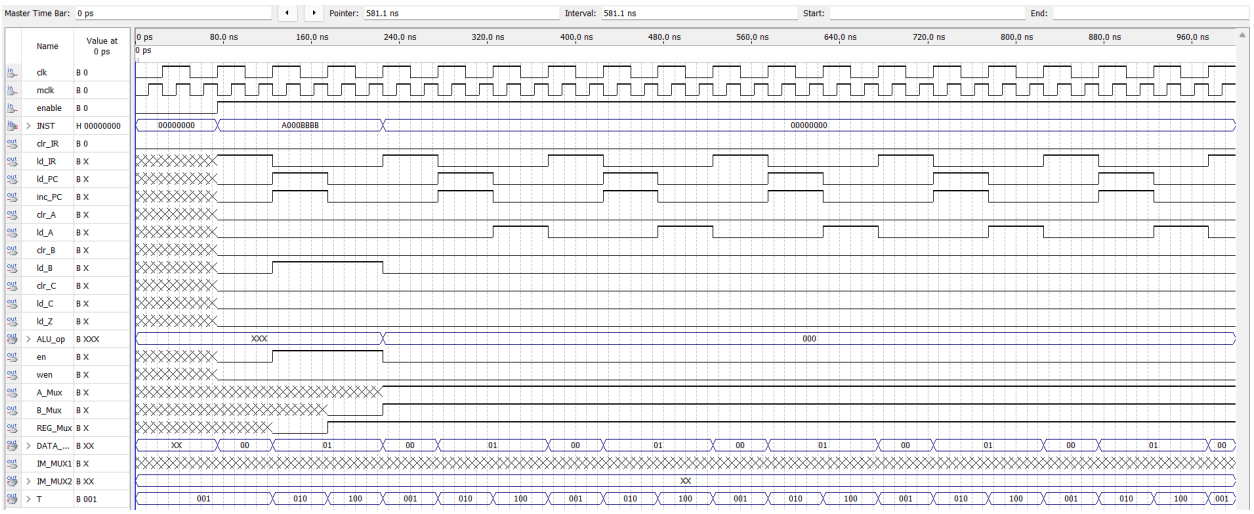

Figure 7: Waveform of the CPU Control Unit Component for operation: LDB(Load into B)

7)

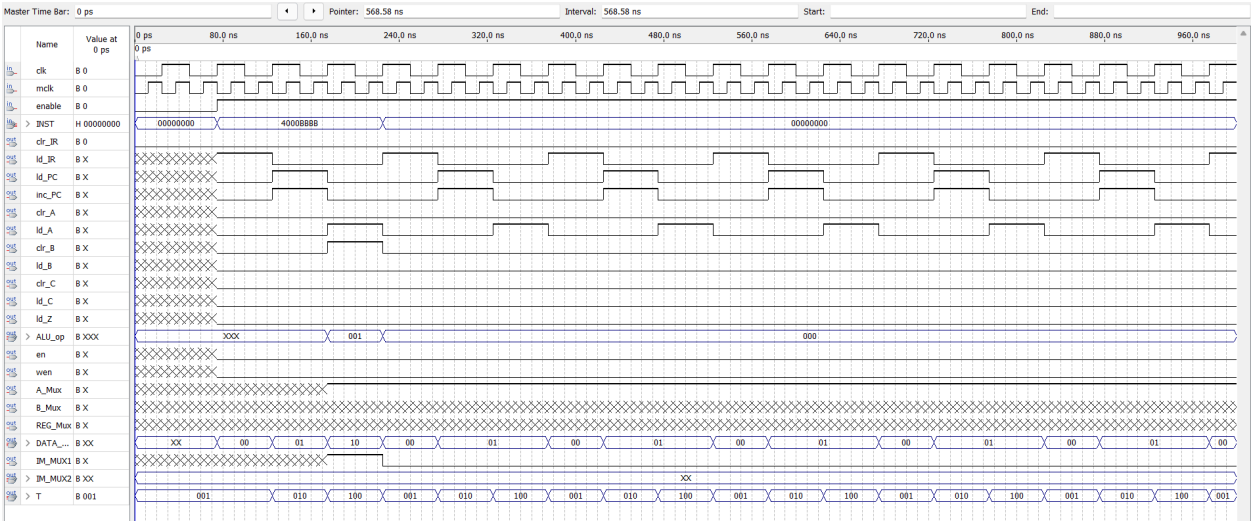

Figure 8: Waveform of the CPU Control Unit Component for operation: LUI(Load Instruction)

8)

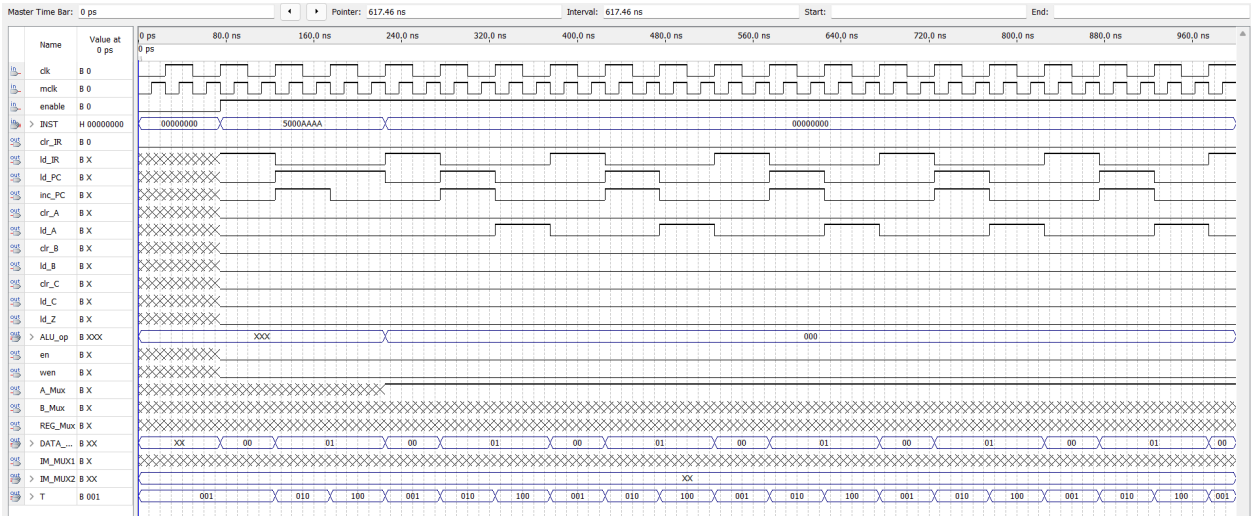

Figure 9: Waveform of the CPU Control Unit Component for operation: JMP(Jump Instruction)

9)

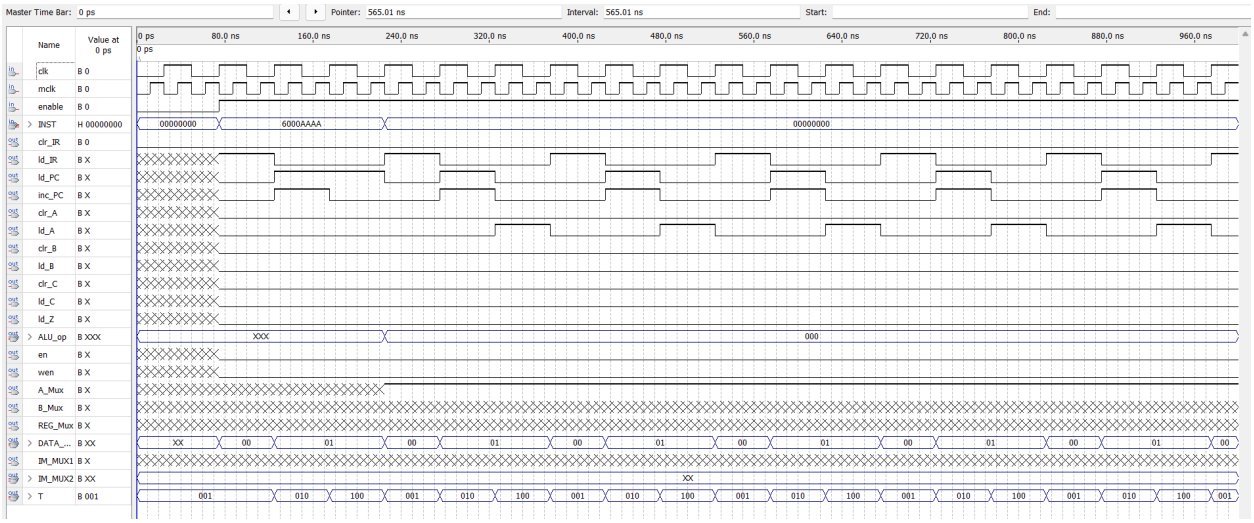

Figure 10: Waveform of the CPU Control Unit Component for operation: BEQ(Branch if-equal)

10)

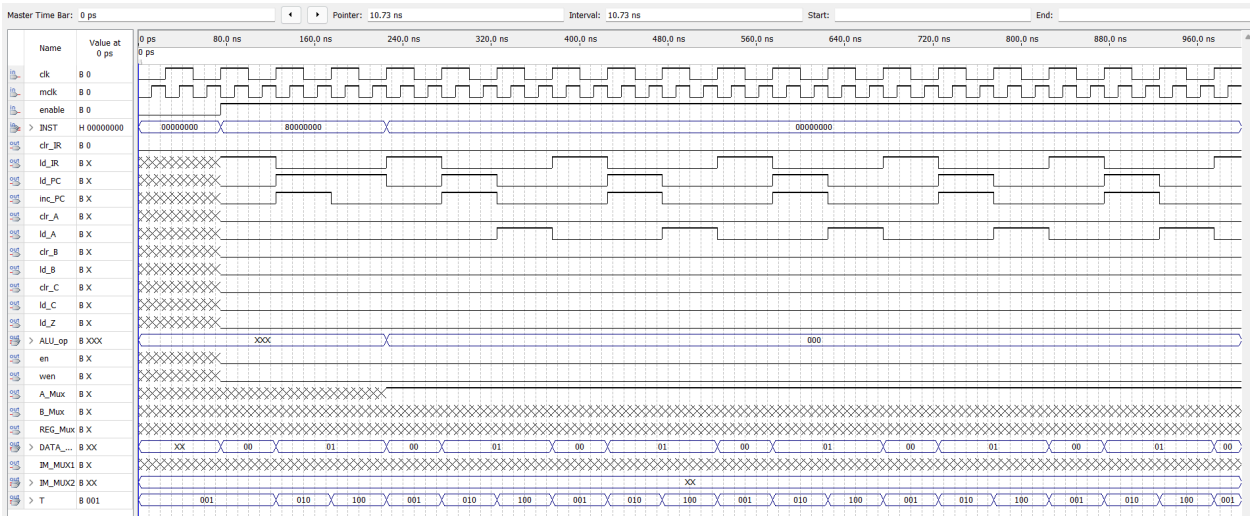

Figure 11: Waveform of the CPU Control Unit for operation: BNE(Branch if-Not-equal)

11)

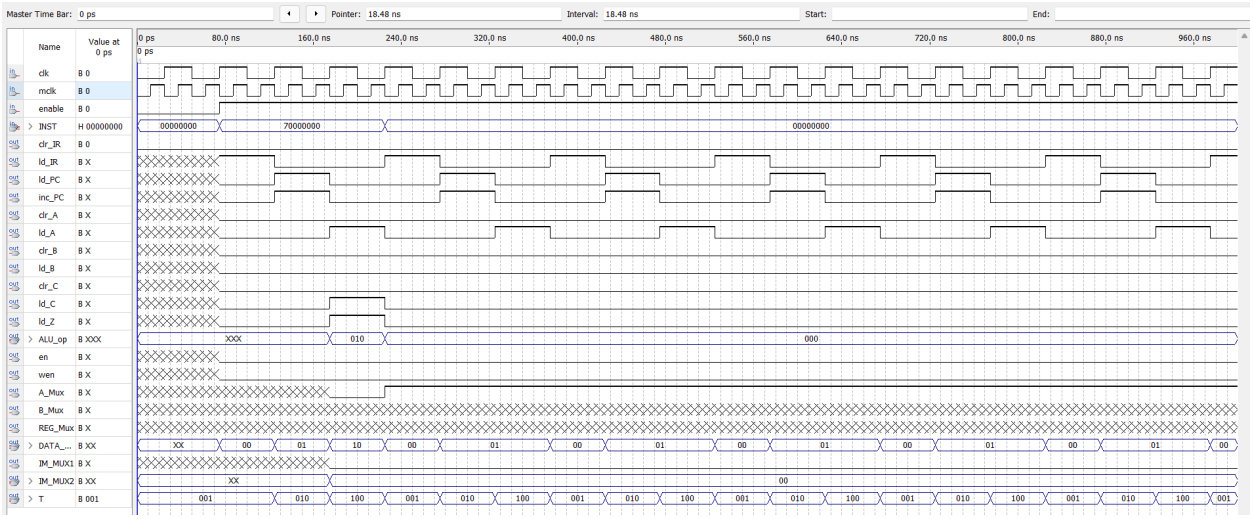

Figure 12: Waveform of the CPU Control Unit Component for operation: ADD

12)

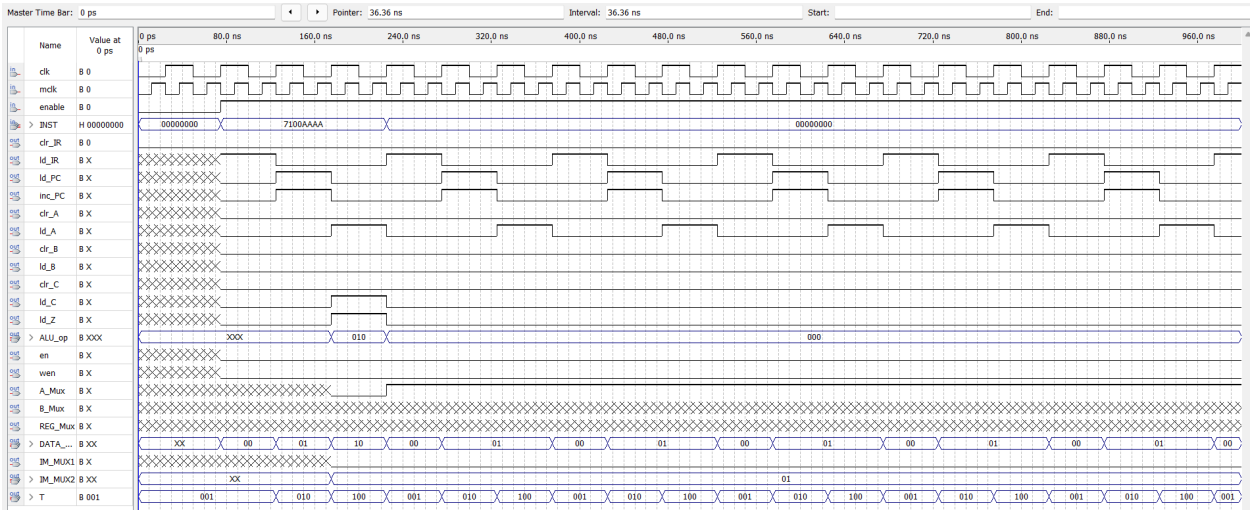

Figure 13: Waveform of the CPU Control Unit for operation: ADDI(Add intermediate)

13)

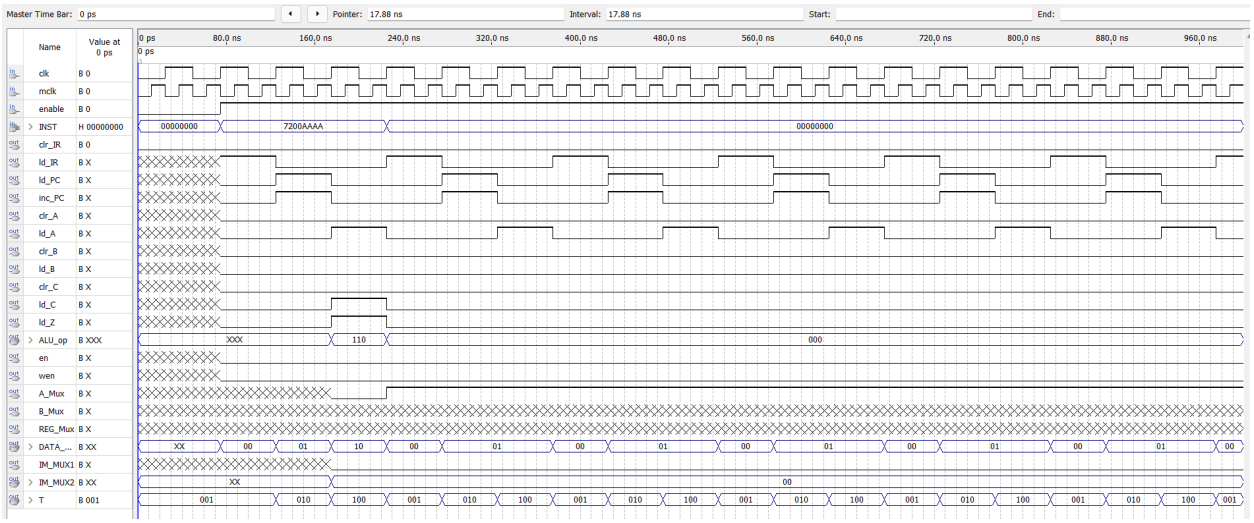

Figure 14: Waveform of the CPU Control Unit Component for operation: SUB

14)

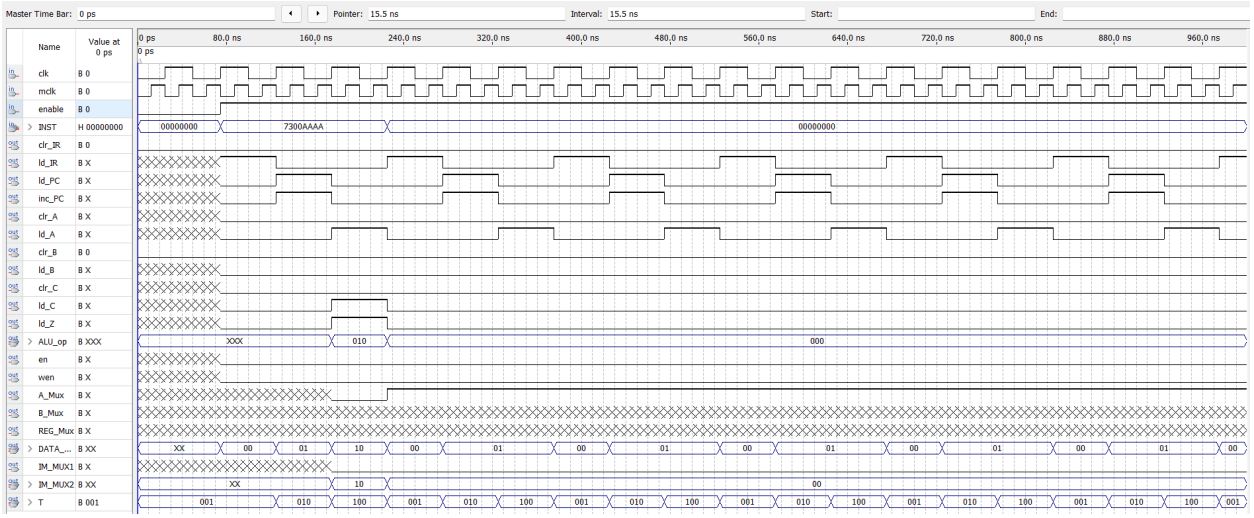

Figure 15: Waveform of the CPU Control Unit Component for operation: INCA(Increment)

15)

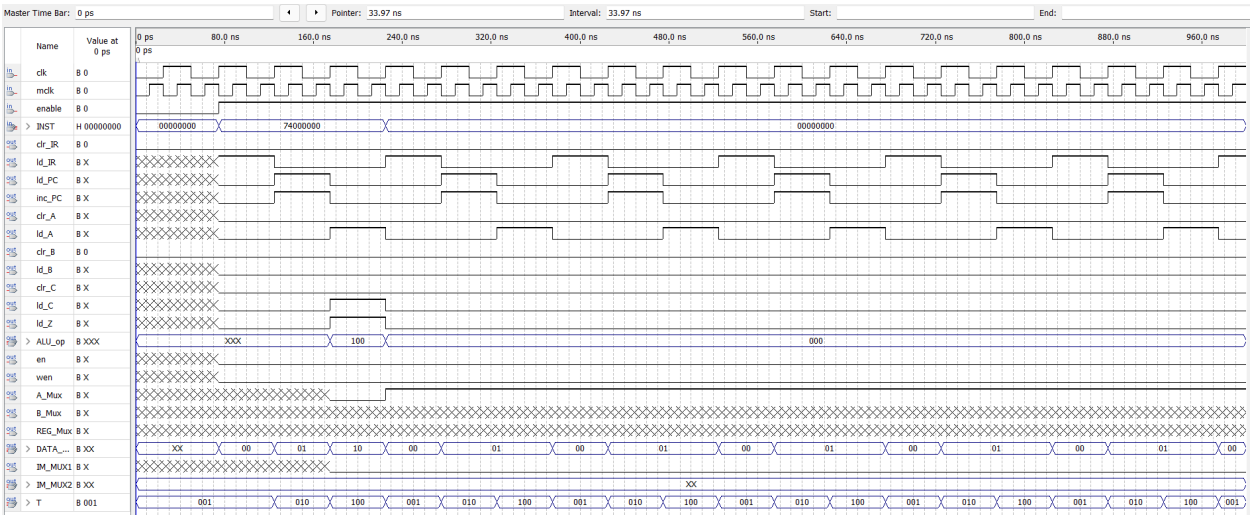

Figure 16: Waveform of the CPU Control Unit Component for operation: ROL(Rotate Left)

16)

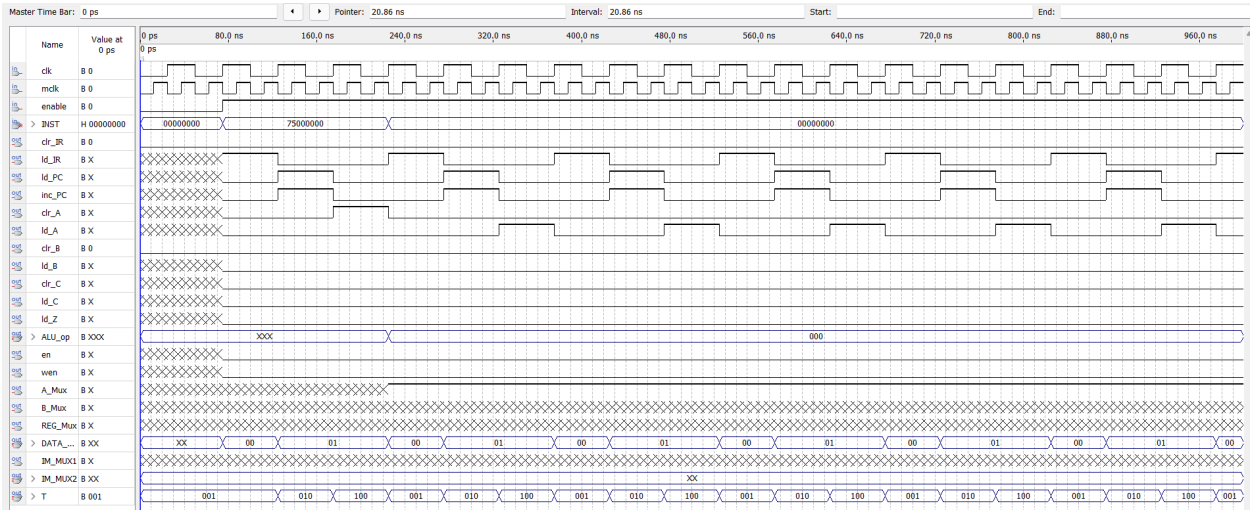

Figure 17: Waveform of the CPU Control Unit Component for operation: CLRA(Clear Reg. A)

17)

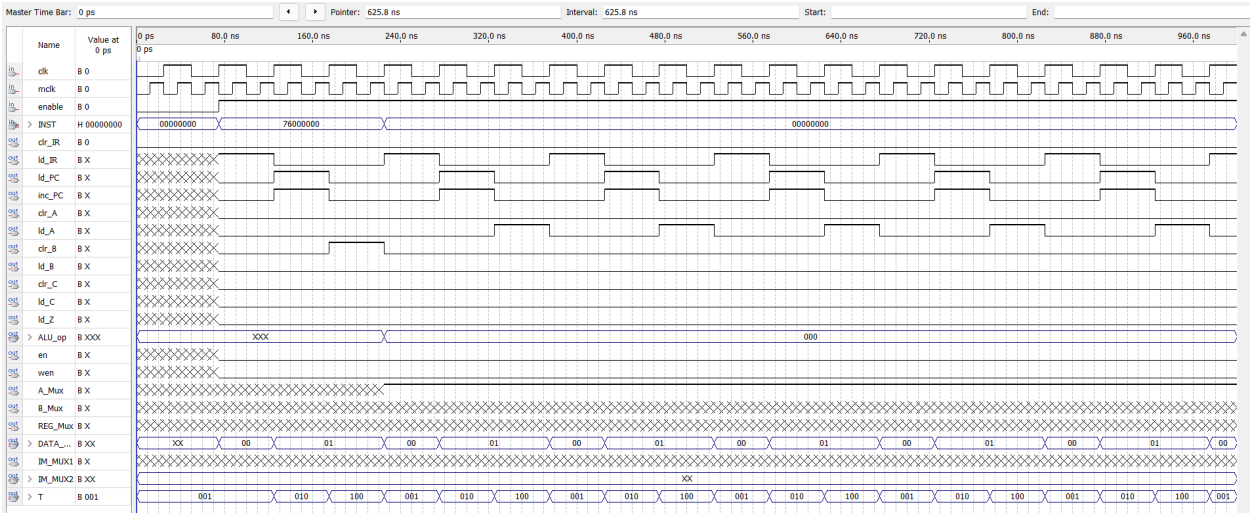

Figure 18: Waveform of the CPU Control Unit Component for operation: CLRB(Clear Reg. B)

18)

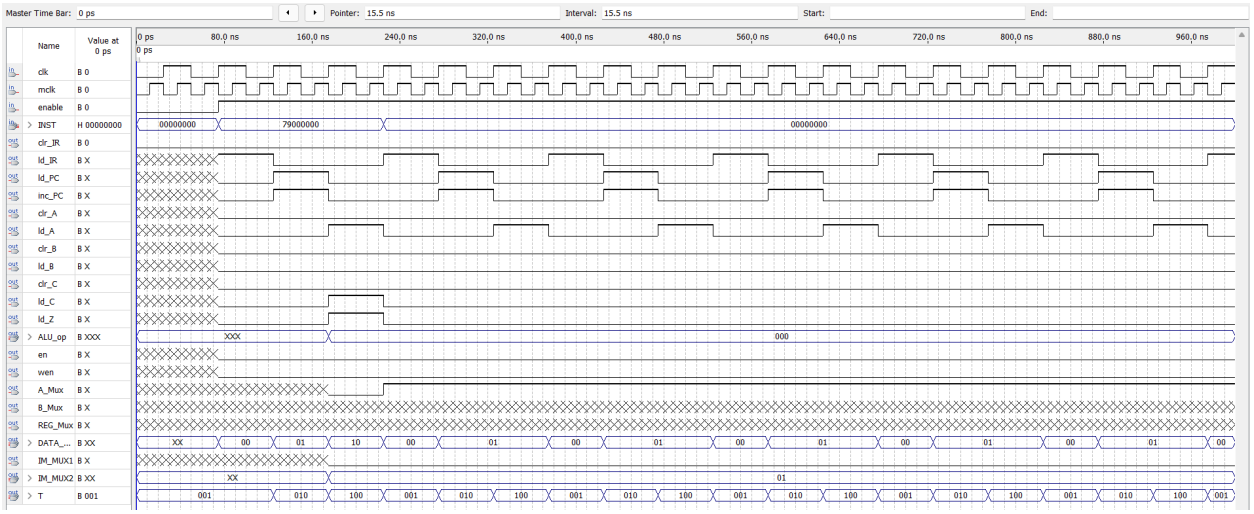

Figure 19: Waveform of the CPU Control Unit for operation: ANDI(And with intermediate)

19)

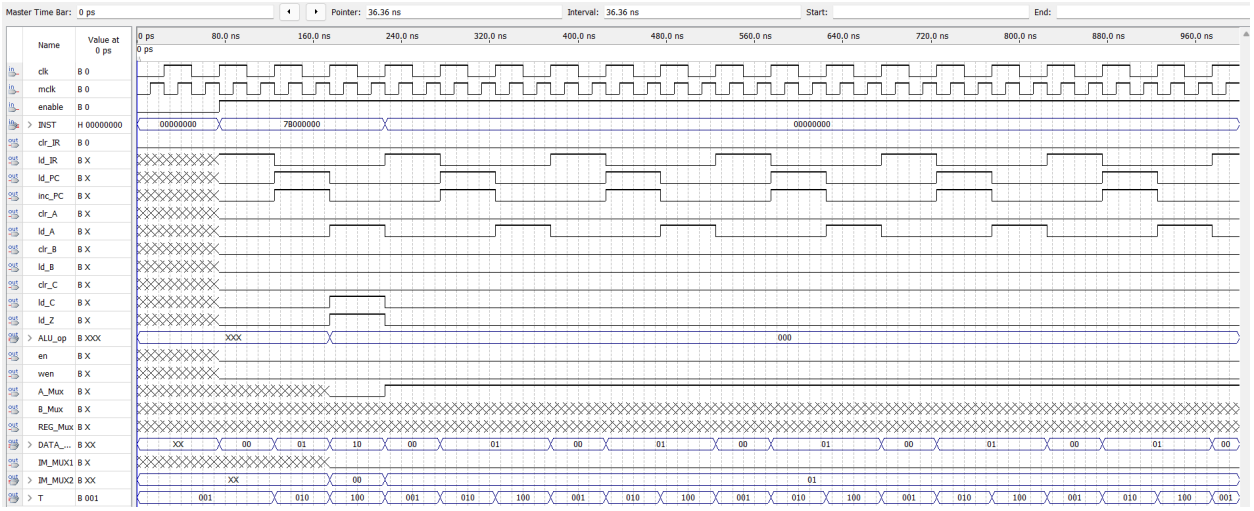

Figure 20: Waveform of the CPU Control Unit Component for operation: AND

20)

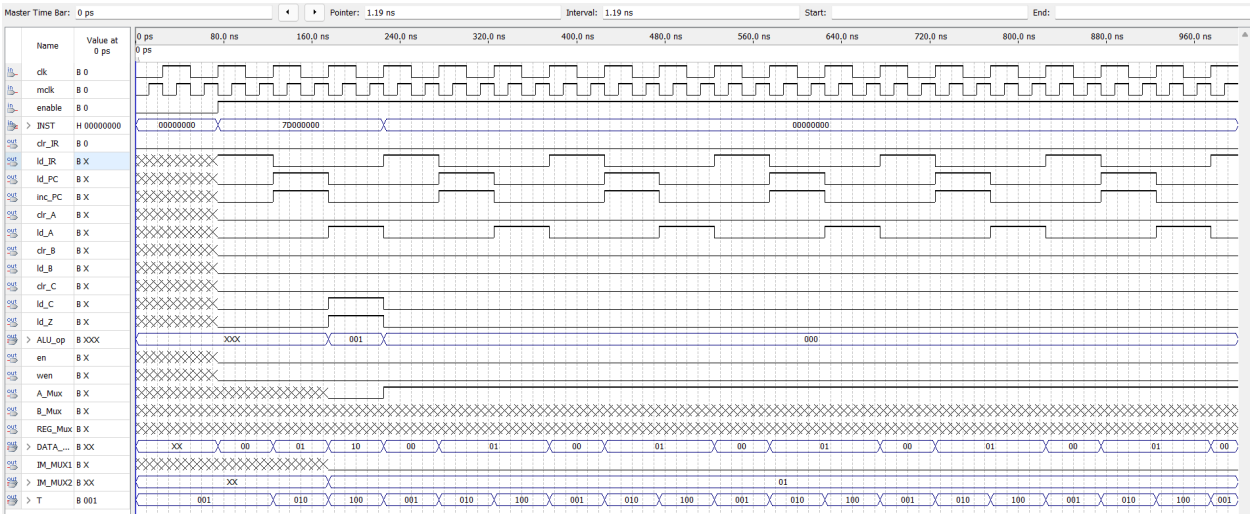

Figure 21: Waveform of the CPU Control Unit for operation: ORI(OR with Intermediate)

21)

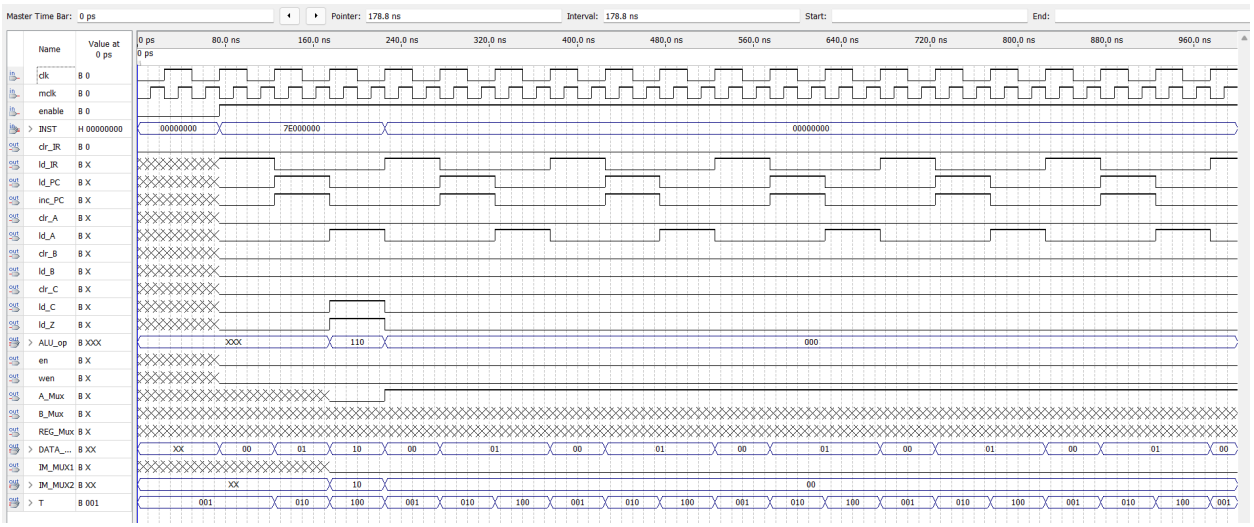

Figure 22: Waveform of the CPU Control Unit Component for operation: DECA(Decrement)

22)

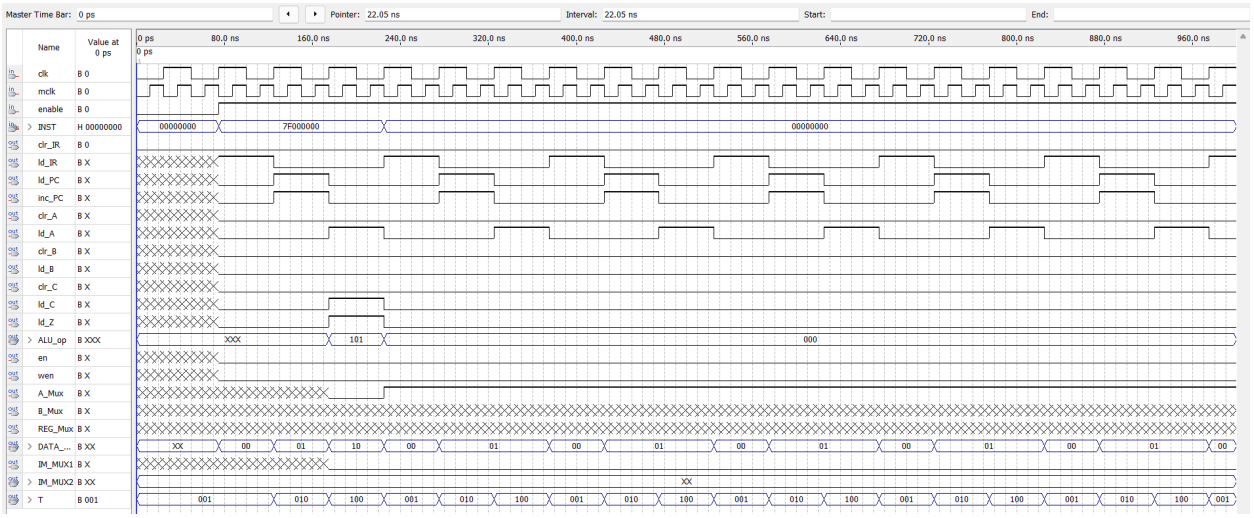

Figure 23: Waveform of the CPU Control Unit Component for operation: ROR(Rotate Right)

In conclusion, the control unit is an essential component of the CPU, responsible for generating the correct control signals to execute instructions appropriately. By automating the coordination of MUXes, Registers, and the ALU, the control unit eliminates manual signal handling, and thus prevents redundancy and eliminates inefficiency. As observed in the waveforms, it systematically directs each instruction through the necessary clock cycle stages, enabling seamless operation of the CPU data-path.