Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Computer Organization and Architecture
Course Number:	COE628
Semester/Year (e.g.F2016)	W2024

Instructor:	Khalid Abdel Hafeez

Assignment/Lab Number:	5
Assignment/Lab Title:	CPU Control Unit Design

Submission Date:	2024/03/25
Due Date:	2024/03/28

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Sarim	Shahwar	501109286	12	SS

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.

Lab Objective:

A data path is essentially the network of hardware parts inside devices like computers or digital processors that helps data move and get processed. It's made up of various components and the links between them that transport data around the system and perform tasks on the data. Some of the components in a data path include Registers, an Arithmetic Logic Unit (ALU), Multiplexers (MUX), a Bus, a Control Unit, and a Memory interface.

Component	Description
LDAI	Load the Accumulator A immediately. Loads value directly into the accumulator register.
LDBI	Load the Accumulator B immediately. Loads value directly into the accumulator register.
STA	Store the value of the register in the memory for A.
STB	Store the value of the register in the memory for B.
LDA	Load Accumulator A. Loads value into memory.
LDB	Load Accumulator B. Loads value into memory.
LUI	Load upper immediately. Loads value into the upper half of the register.
JMP	Alter the flow of execution by setting the program counter to the address specified by the operand.
ANDI	“AND” immediate period operation between “AND” and immediate value
ADDI	Adds an immediate value to the register by using a register file adder and another separate register file to store the immediate value
ORI	“OR” immediate. “OR” operation between register and immediate value
ADD	The function to add contents to registers usually registers “A” and “B”
SUB	The function to subtract contents of two registers usually registers “A” and “B”
DECA	Decrement accumulator. decrease the value in the accumulator by 1
INCA	Increments a register value by using a registered file and an adder
ROL	Rotate left. Shifts all bits in a register/memory to the Left
ROR	Rotate right, Shifts all bits in a register/ memory to the Right
CLRA	Clears accumulator “A”, sets value to 0
CLRB	Clears accumulator “B”, sets value to 0
BEQ	Branch if equal
BNE	Branch if not equal.
RED	Gets 8-bits from 32-bits

Experiment Details:

Data-Path VHDL:

```
library ieee;
use ieee.std_logic_1164.ALL;

ENTITY Control IS
   PORT(
         clk, mlck                  : IN  STD_LOGIC;
         enable                     : IN  STD_LOGIC;
         statusC, statusZ           : IN  STD_LOGIC;
         INST                       : IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
         A_MUX, B_MUX               : OUT STD_LOGIC;
         IM_MUX1, REG_Mux           : OUT STD_LOGIC;
         IM_MUX2, DATA_Mux          : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
         ALU_op                     : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
         inc_PC, ld_PC              : OUT STD_LOGIC;
         clr_IR                     : OUT STD_LOGIC;
         ld_IR                      : OUT STD_LOGIC;
         clr_A, clr_B, clr_C, clr_Z : OUT STD_LOGIC;
         ld_A, ld_B, ld_C, ld_Z     : OUT STD_LOGIC;
         T                          : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
         wen, en                    : OUT STD_LOGIC
   );
END Control;

ARCHITECTURE description OF Control IS
   TYPE STATETYPE IS (state_0, state_1, state_2);
   SIGNAL present_state: STATETYPE;
   SIGNAL Instruction_sig: STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL Instruction_sig2: STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
   Instruction_sig <= INST(31 DOWNTO 28);
   Instruction_sig2 <= INST(31 DOWNTO 24);

   ------- OPERATION DECODER -------
   PROCESS (present_state, INST, statusC, statusZ, enable, Instruction_sig, Instruction_sig2)
   BEGIN
      if enable = '1' then
         if present_state = state_0 then
            DATA_Mux <= "00"; --Fetch address of next instruction
            clr_IR <= '0';
            ld_IR <= '1';
            ld_PC <= '0';
            inc_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';
            en <= '0';
            wen <= '0';

         elsif present_state = state_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';
            en <= '0';
            wen <= '0';

            if Instruction_sig = "0010" then --STA
               clr_IR <= '0';
               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';
               REG_Mux <= '0';
```

```
                REG_Mux <= '0';
                DATA_Mux <= "00";
                en <= '1';
                wen <= '1';

            elsif Instruction_sig = "0011" then --STB
                clr_IR <= '0';
                --ld_Z <= '0';
                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';
                REG_Mux <= '1';
                DATA_Mux <= "00";
                en <= '1';
                wen <= '1';

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

            elsif Instruction_sig = "1010" then --LDB
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '1';
                inc_PC <= '1';
                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';  --EN
                wen <= '0'; --WEN
            end if;  --END IF FOR LOAD STORE IN STAGE 1

        elsif present_state = state_2 then
            if Instruction_sig = "0101" then --JUMP
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '1';
                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';

            elsif Instruction_sig = "0110" then --BEQ
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '1';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '0';
```

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

        elsif Instruction_sig = "1000" then --BNE
            clr_IR <= '0';
            ld_IR <= '0';
            ld_PC <= '1';
            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';

        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';
            A_Mux <= '0';
            DATA_Mux <= "01";
            en <= '1';
            wen <= '0';

        elsif Instruction_sig = "1010" then --LDB
            clr_IR <= '0';
```

```
            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_A <= '0';
                ld_B <= '0';
                clr_B <= '0';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';
                REG_Mux <= '1';
                DATA_Mux <= "00";
                en <= '1';
                wen <= '1';

            elsif Instruction_sig = "0000" then --LDAI
                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';
                A_Mux <= '1';

            elsif Instruction_sig = "0000" then --LDBI
                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 <= '1';

            elsif Instruction_sig = "0100" then --LUI
                clr_IR <= '0';
                ld_IR <= '0';
                ld_PC <= '0';
                inc_PC <= '0';
                clr_A <= '0';
                ld_A <= '1';
                ld_B <= '0';
                clr_B <= '1';
                clr_C <= '0';
                ld_C <= '0';
                clr_Z <= '0';
                ld_Z <= '0';
                ALU_op <= "001";
                A_Mux <= '0';
                DATA_Mux <= "10";
                IM_MUX1 <= '1';

            elsif Instruction_sig2 = "01111001" then --ANDI
                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';
```

```
                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 <= "01";

            elsif Instruction_sig2 = "01111110" then --DECA
                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 <= "110";
                A_Mux <= '0';
                DATA_Mux <= "10";
                IM_MUX1 <= '0';
                IM_MUX2 <= "10";

            elsif Instruction_sig2 = "01110000" then --ADD
                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 <= "00";

            elsif Instruction_sig2 = "01110010" then --SUB
                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 <= "110";
                A_Mux <= '0';
                DATA_Mux <= "00";
                IM_MUX1 <= '0';
                IM_MUX2 <= "00";

            elsif Instruction_sig2 = "01110011" then --INCA
                clr_IR <= '0';
```

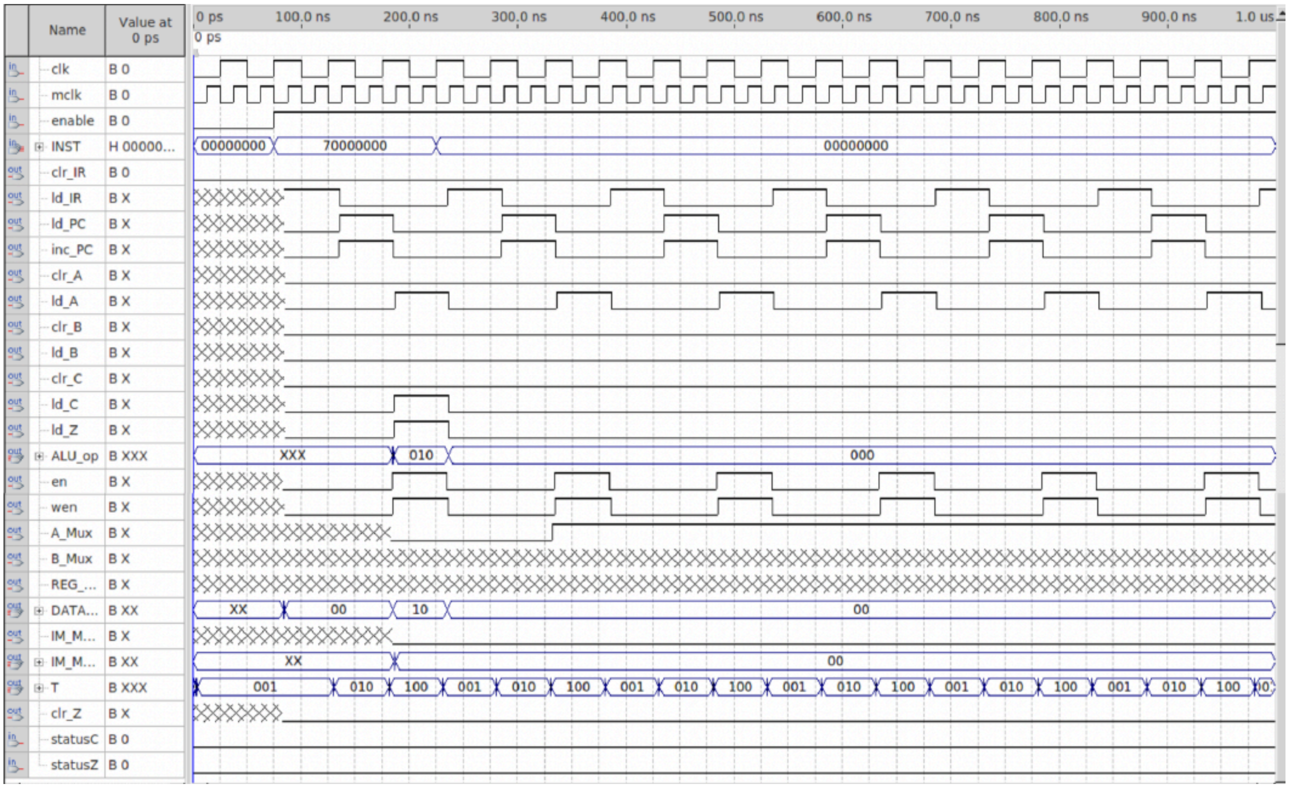
```
383             elsif Instruction_sig2 = "01110011" then --INCA
384                 clr_IR <= '0';
385                 ld_IR <= '0';
386                 ld_PC <= '0';
387                 inc_PC <= '0';
388                 clr_A <= '0';
389                 ld_A <= '1';
390                 ld_B <= '0';
391                 clr_B <= '0';
392                 clr_C <= '0';
393                 ld_C <= '1';
394                 clr_Z <= '0';
395                 ld_Z <= '1';
396                 ALU_op <= "010";
397                 A_Mux <= '0';
398                 DATA_Mux <= "10";
399                 IM_MUX1 <= '0';
400                 IM_MUX2 <= "10";
401
402             elsif Instruction_sig2 = "01111011" then --AND
403                 clr_IR <= '0';
404                 ld_IR <= '0';
405                 ld_PC <= '0';
406                 inc_PC <= '0';
407                 clr_A <= '0';
408                 ld_A <= '1';
409                 ld_B <= '0';
410                 clr_B <= '0';
411                 clr_C <= '0';
412                 ld_C <= '1';
413                 clr_Z <= '0';
414                 ld_Z <= '1';
415                 ALU_op <= "000";
416                 A_Mux <= '0';
417                 DATA_Mux <= "10";
418                 IM_MUX1 <= '0';
419                 IM_MUX2 <= "00";
420
421             elsif Instruction_sig2 = "01110001" then --ADDI
422                 clr_IR <= '0';
423                 ld_IR <= '0';
424                 ld_PC <= '0';
425                 inc_PC <= '0';
426                 clr_A <= '0';
427                 ld_A <= '1';
428                 ld_B <= '0';
429                 clr_B <= '0';
430                 clr_C <= '0';
431                 ld_C <= '1';
432                 clr_Z <= '0';
433                 ld_Z <= '1';
434                 ALU_op <= "010";
435                 A_Mux <= '0';
436                 DATA_Mux <= "10";
437                 IM_MUX1 <= '0';
438                 IM_MUX2 <= "01";
439
440             elsif Instruction_sig2 = "01111101" then --ORI
441                 clr_IR <= '0';
442                 ld_IR <= '0';
443                 ld_PC <= '0';
444                 inc_PC <= '0';
445                 clr_A <= '0';
446                 ld_A <= '1';
447                 ld_B <= '0';
448                 clr_B <= '0';
449                 clr_C <= '0';
450                 ld_C <= '1';
451                 clr_Z <= '0';
452                 ld_Z <= '1';
```

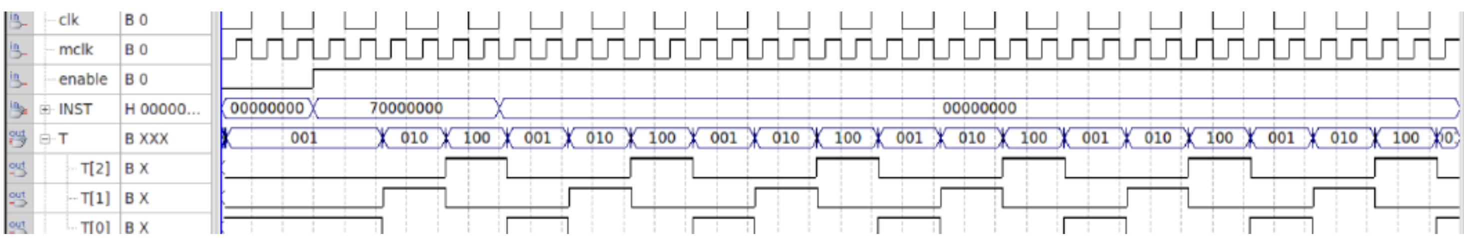
```
452                 ld_Z <= '1';
453                 ALU_op <= "001";
454                 A_Mux <= '0';
455                 DATA_Mux <= "10";
456                 IM_MUX1 <= '0';
457                 IM_MUX2 <= "01";
458
459             elsif Instruction_sig2 = "01110100" then --ROL
460                 clr_IR <= '0';
461                 ld_IR <= '0';
462                 ld_PC <= '0';
463                 inc_PC <= '0';
464                 clr_A <= '0';
465                 ld_A <= '1';
466                 ld_B <= '0';
467                 clr_B <= '0';
468                 clr_C <= '0';
469                 ld_C <= '1';
470                 clr_Z <= '0';
471                 ld_Z <= '1';
472                 ALU_op <= "100";
473                 A_Mux <= '0';
474                 DATA_Mux <= "10";
475                 IM_MUX1 <= '0';
476
477             elsif Instruction_sig2 = "01111111" then --ROR
478                 clr_IR <= '0';
479                 ld_IR <= '0';
480                 ld_PC <= '0';
481                 inc_PC <= '0';
482                 clr_A <= '0';
483                 ld_A <= '1';
484                 ld_B <= '0';
485                 clr_B <= '0';
486                 clr_C <= '0';
487                 ld_C <= '1';
488                 clr_Z <= '0';
489                 ld_Z <= '1';
490                 ALU_op <= "101";
491                 A_Mux <= '0';
492                 DATA_Mux <= "10";
493                 IM_MUX1 <= '0';
494
495             elsif Instruction_sig2 = "01110101" then --CLR_A
496                 clr_IR <= '0';
497                 ld_IR <= '0';
498                 ld_PC <= '0';
499                 inc_PC <= '0';
500                 clr_A <= '1';
501                 ld_A <= '0';
502                 ld_B <= '0';
503                 clr_B <= '0';
504                 clr_C <= '0';
505                 ld_C <= '0';
506                 clr_Z <= '0';
507                 ld_Z <= '0';
508
509             elsif Instruction_sig2 = "01110110" then --CLR_B
510                 clr_IR <= '0';
511                 ld_IR <= '0';
512                 ld_PC <= '0';
513                 inc_PC <= '0';
514                 clr_A <= '0';
515                 ld_A <= '0';
516                 ld_B <= '0';
517                 clr_B <= '1';
518                 clr_C <= '0';
519                 ld_C <= '0';
520                 clr_Z <= '0';
521                 ld_Z <= '0';
```

```
            ld_Z <= '0';

         elsif Instruction_sig2 = "01110111" then --CLR_C
            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 <= '1';
            ld_C <= '0';
            clr_Z <= '0';
            ld_Z <= '0';

         elsif Instruction_sig2 = "01111000" then --CLR_Z
            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 <= '1';
            ld_Z <= '0';

         elsif Instruction_sig2 = "01111010" then --TSTZ
            if (statusZ = '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;

            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;
```

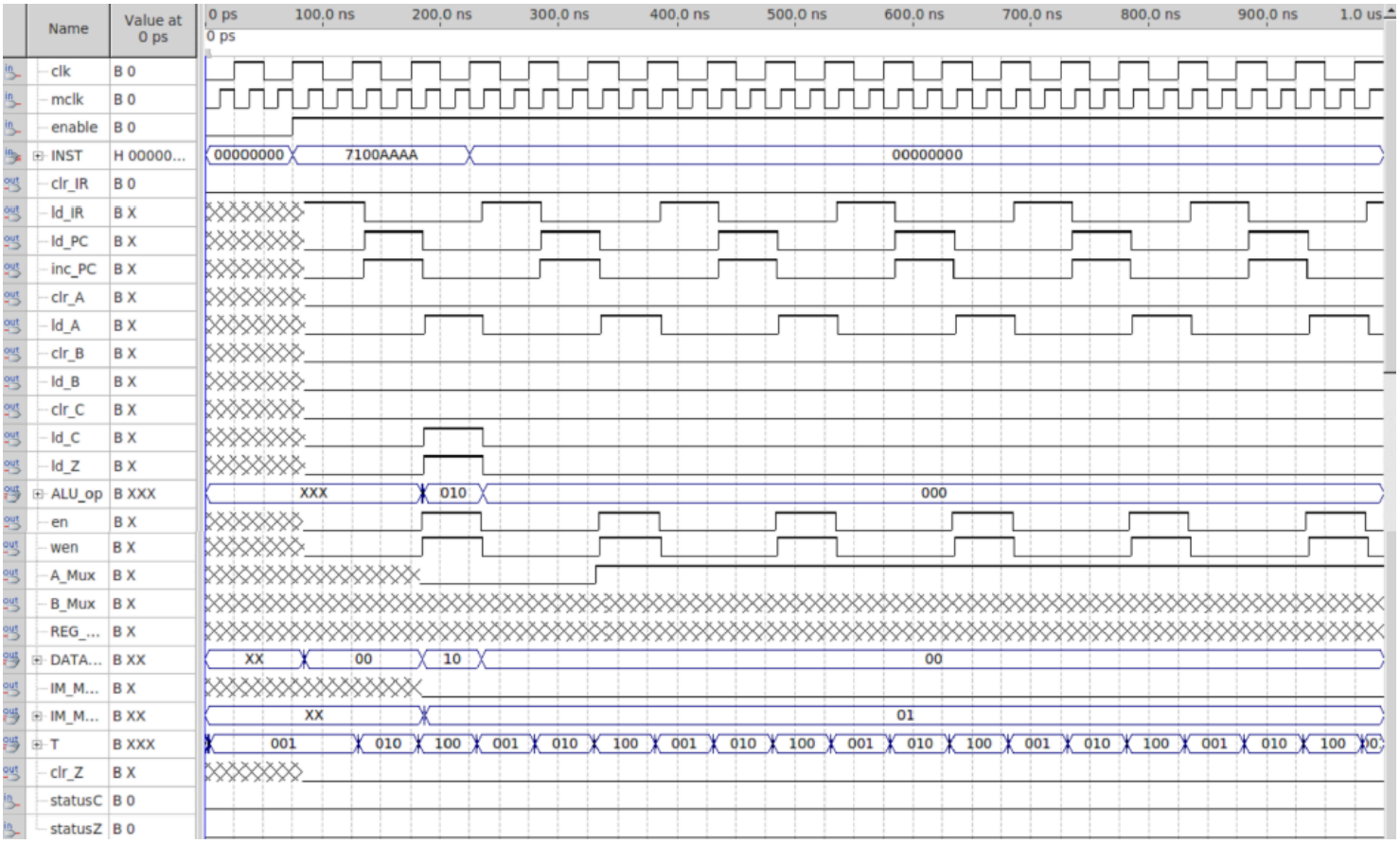
Results: The waveform below displays the function of the Data path VHDL code. The waveform includes the clock, the data address, the data input, the enable signal input, the write enable signal and the data output.

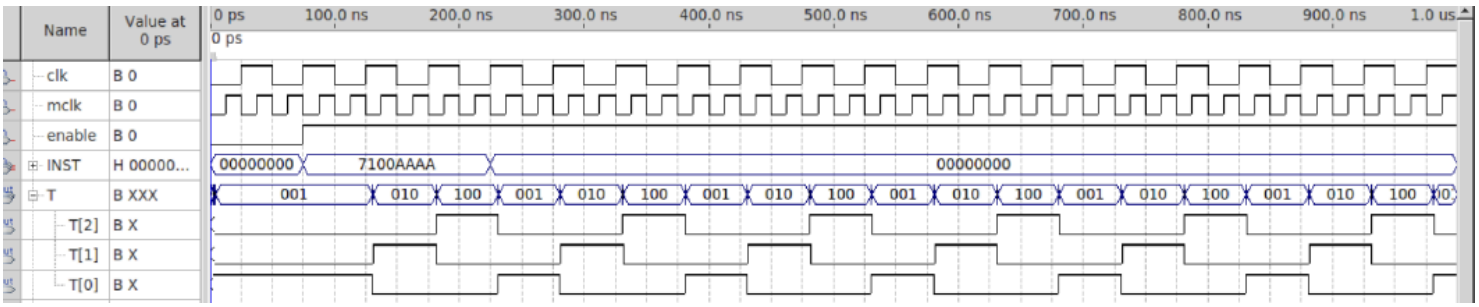
ADD:



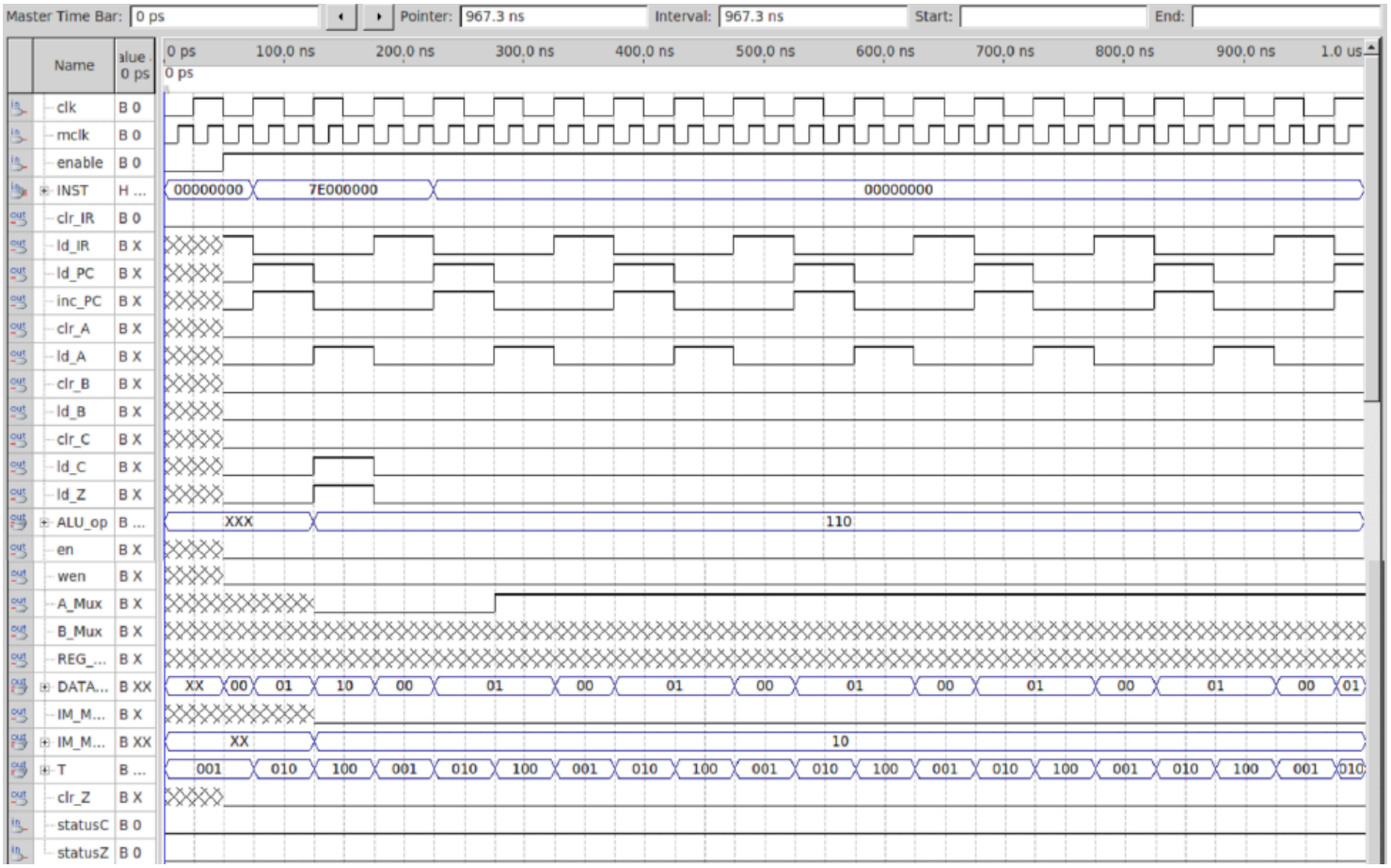


ADDI:

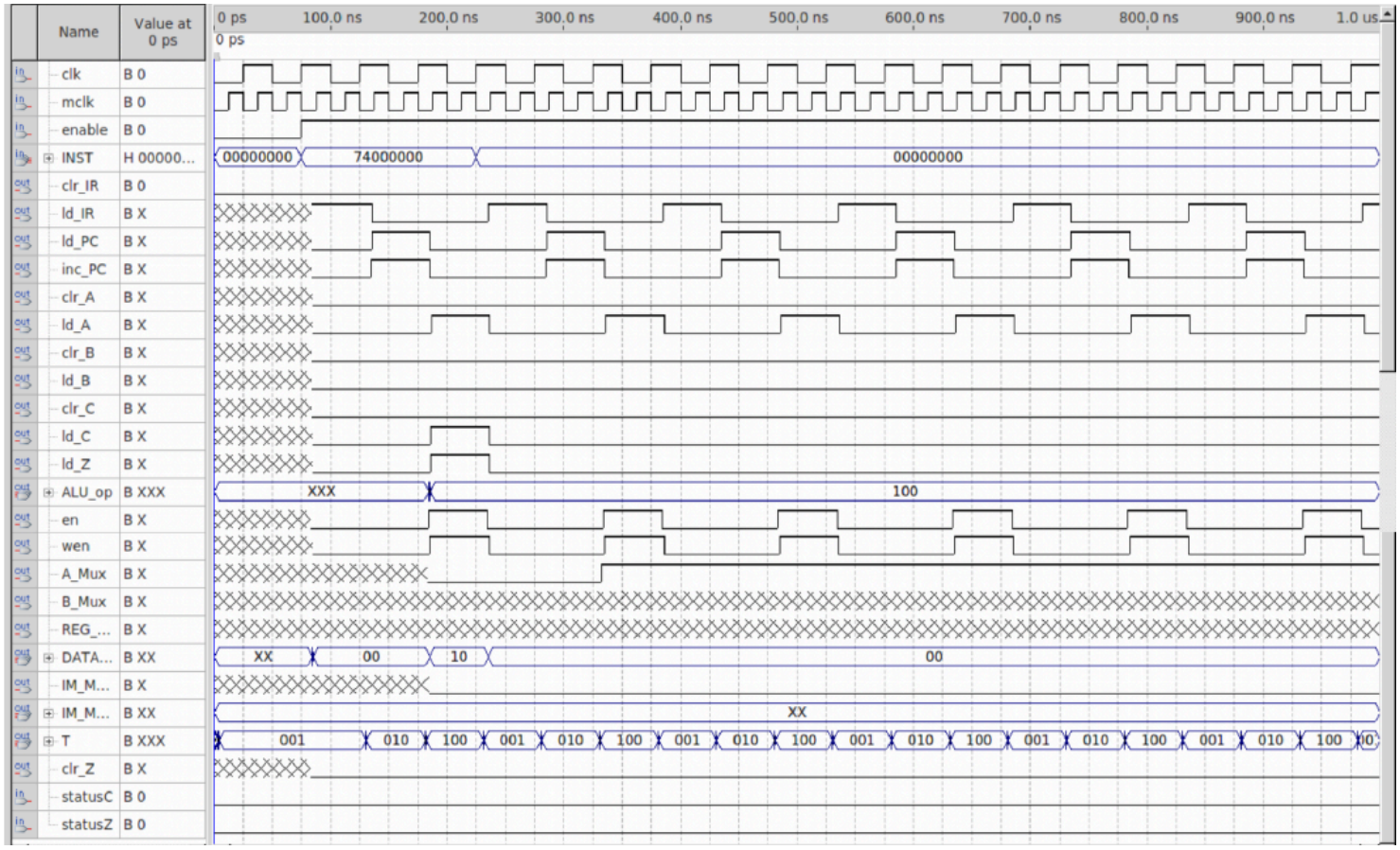

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
clr_IR
ld_iR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
7100AAAA
00000000
XXX
010
000
XX
00
10
00
XX
01
001
010
100

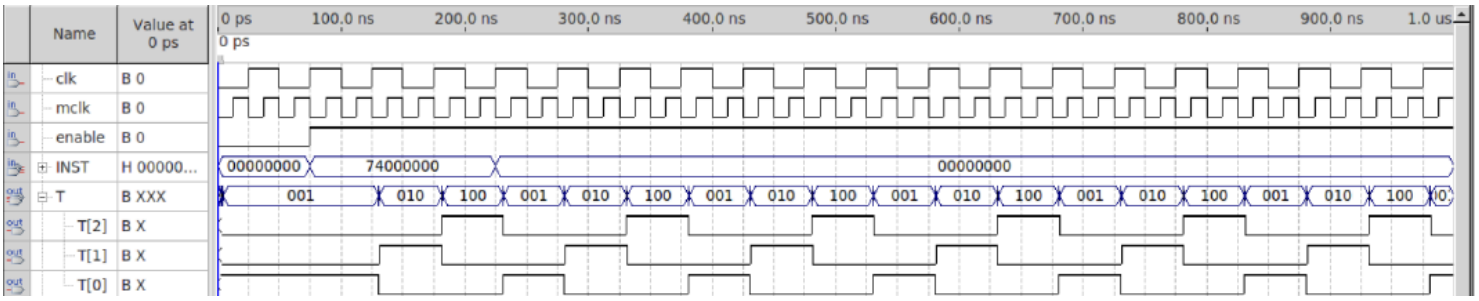

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
T
T[2]
T[1]
T[0]
00000000
7100AAAA
00000000
001
010
100

ANDI:

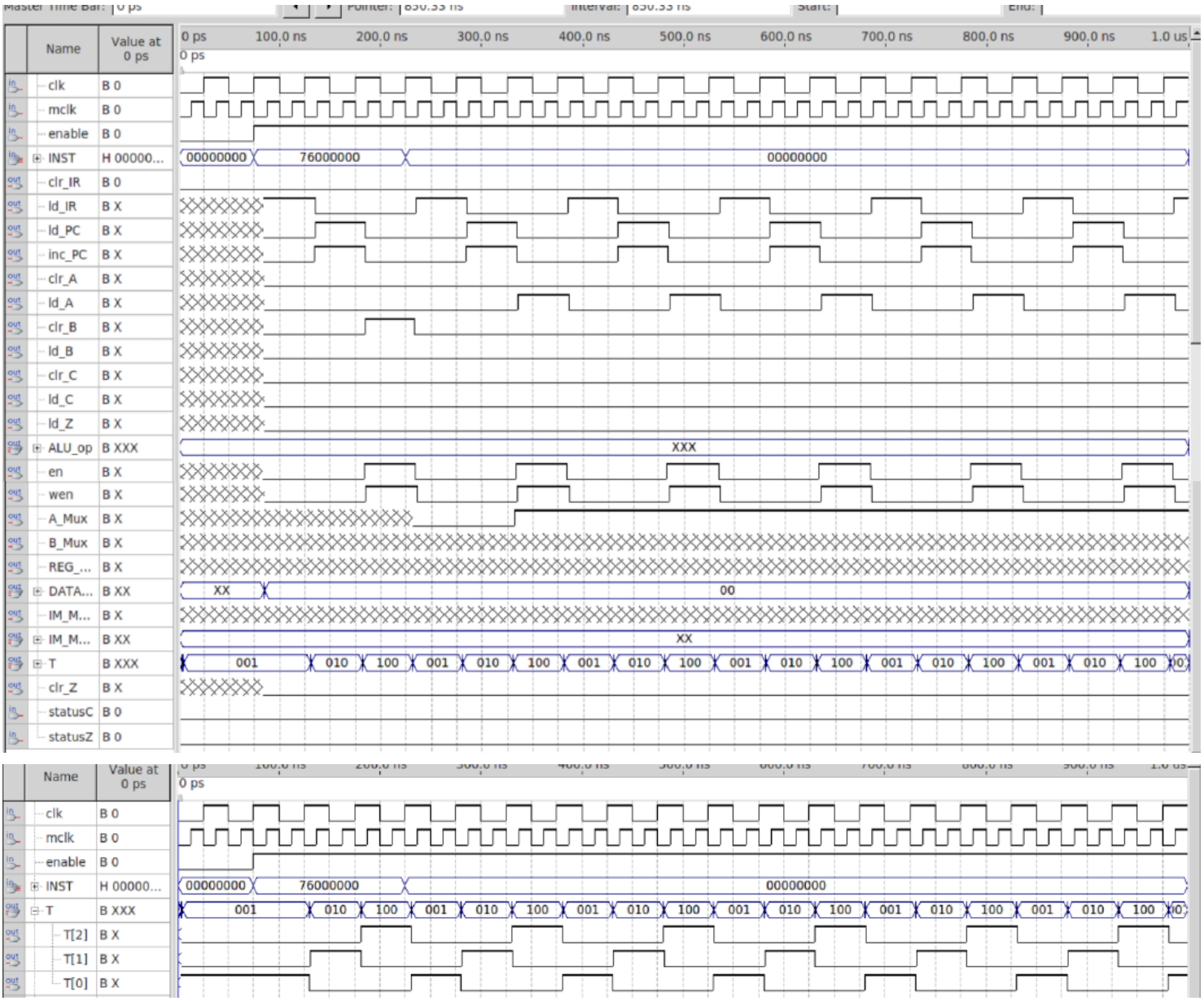

Master Time Bar: 0 ps
Pointer: 967.3 ns
Interval: 967.3 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
7E000000
00000000
XXX
110
XX
10
79000000
T[2]
T[1]
T[0]

CLRA:

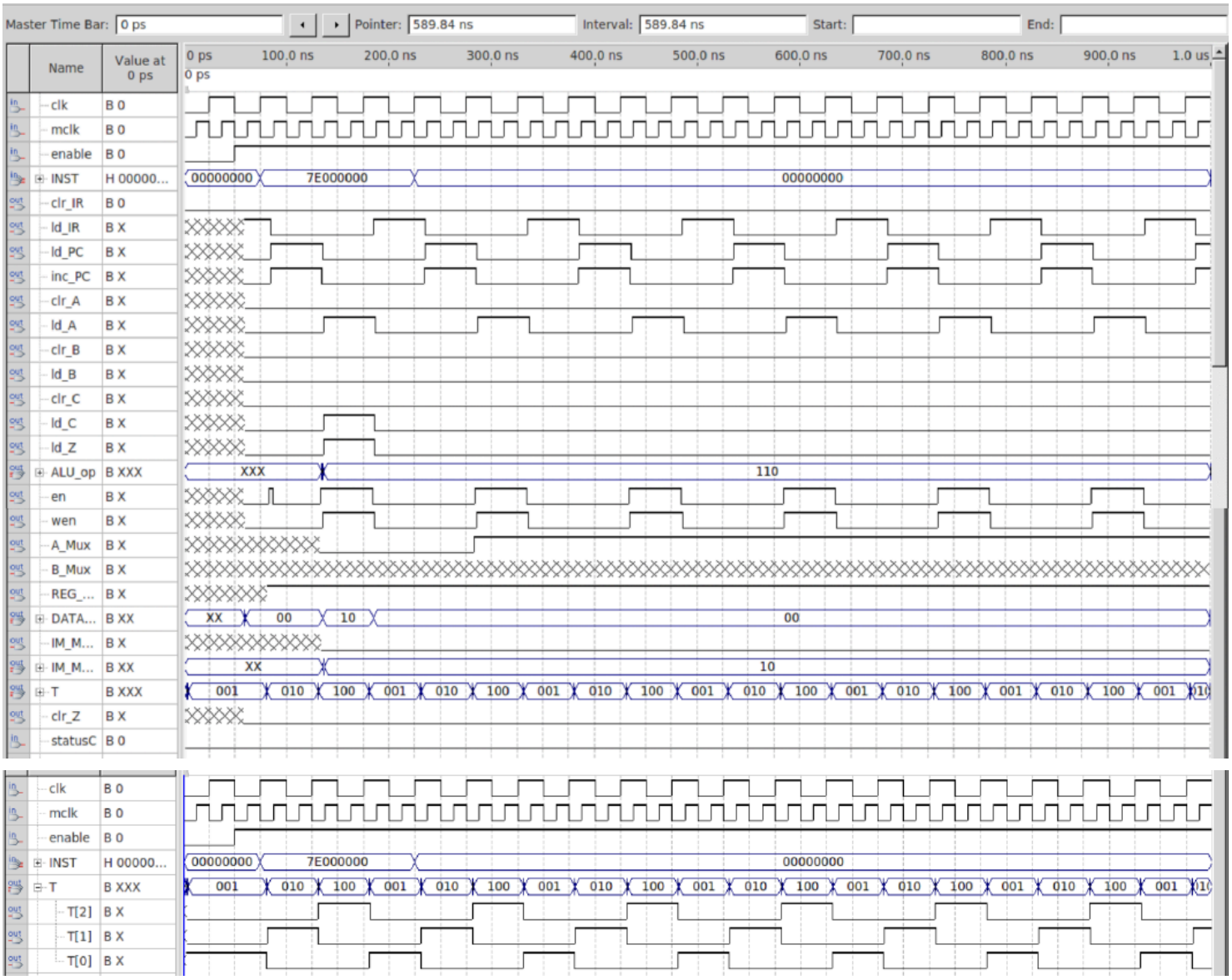

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
74000000
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
100
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
10
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
74000000
00000000
T B XXX
001
010
100
T[2] B X
T[1] B X
T[0] B X

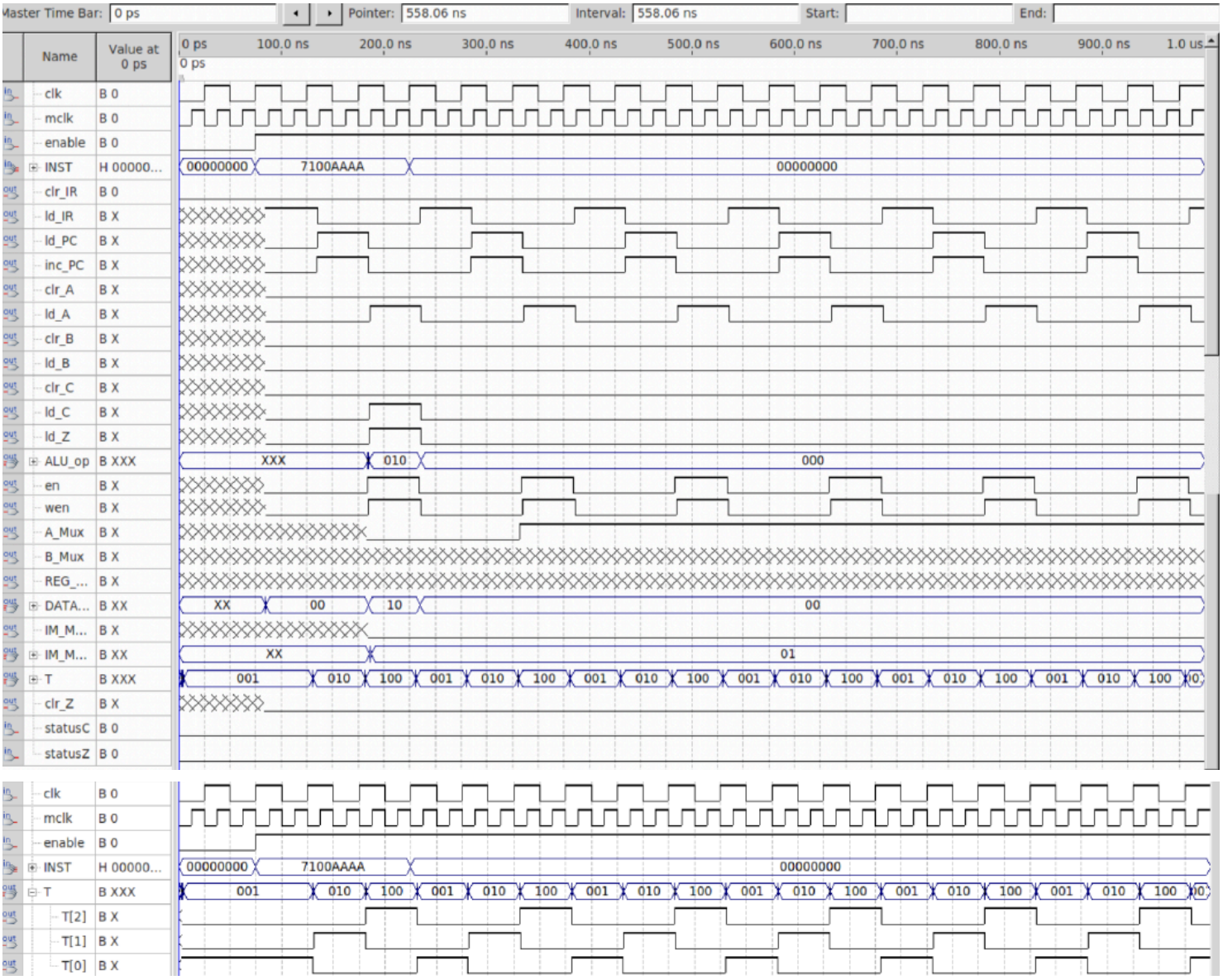
CLRB:


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
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_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
76000000
00000000
XXX
XX
00
XX
001
010
100
T[2]
T[1]
T[0]

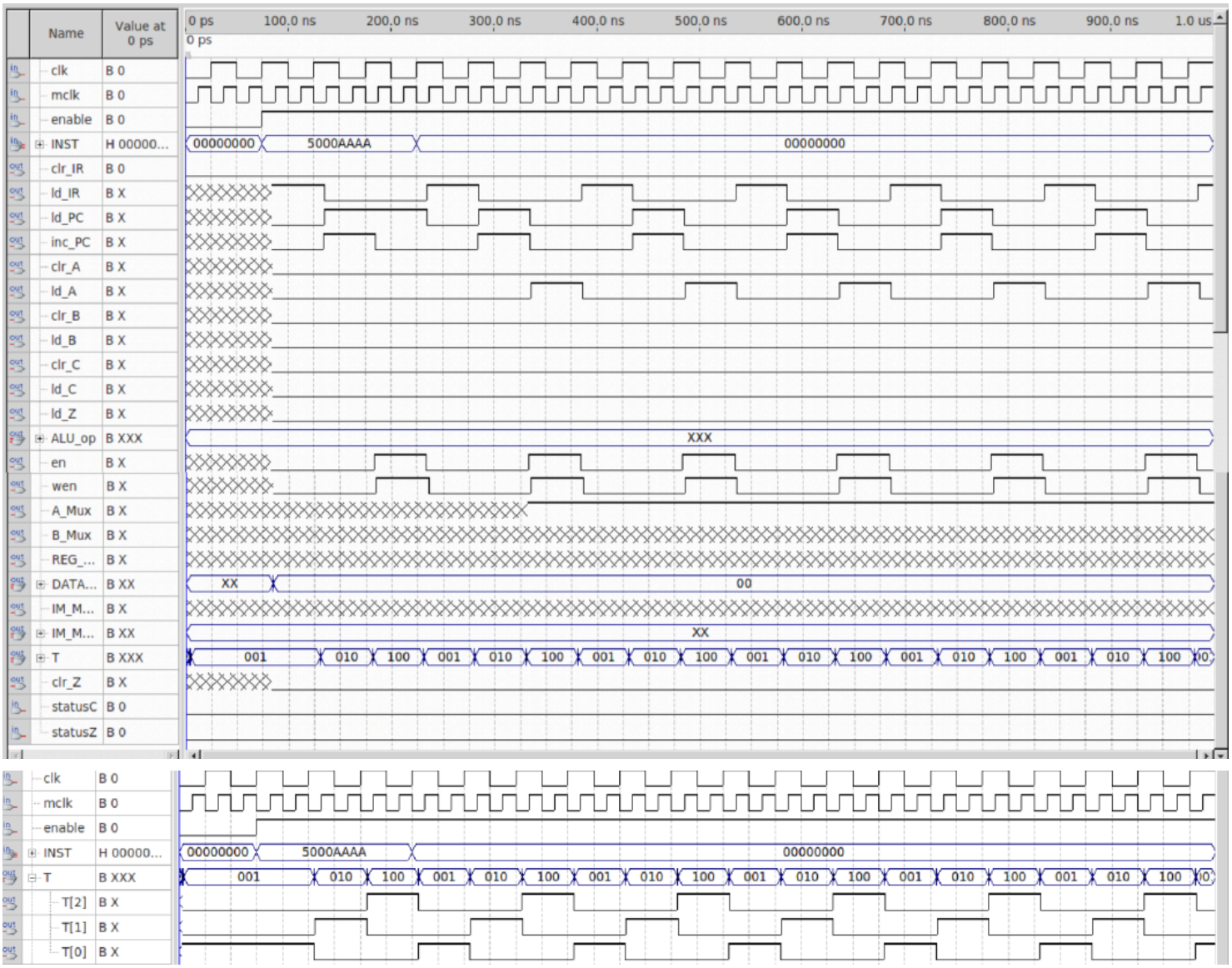
DECA:


Master Time Bar: 0 ps
Pointer: 589.84 ns
Interval: 589.84 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
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_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
00000000
7E000000
00000000
XXX
110
XX
00
10
00
XX
10
001
010
100
T[2]
T[1]
T[0]

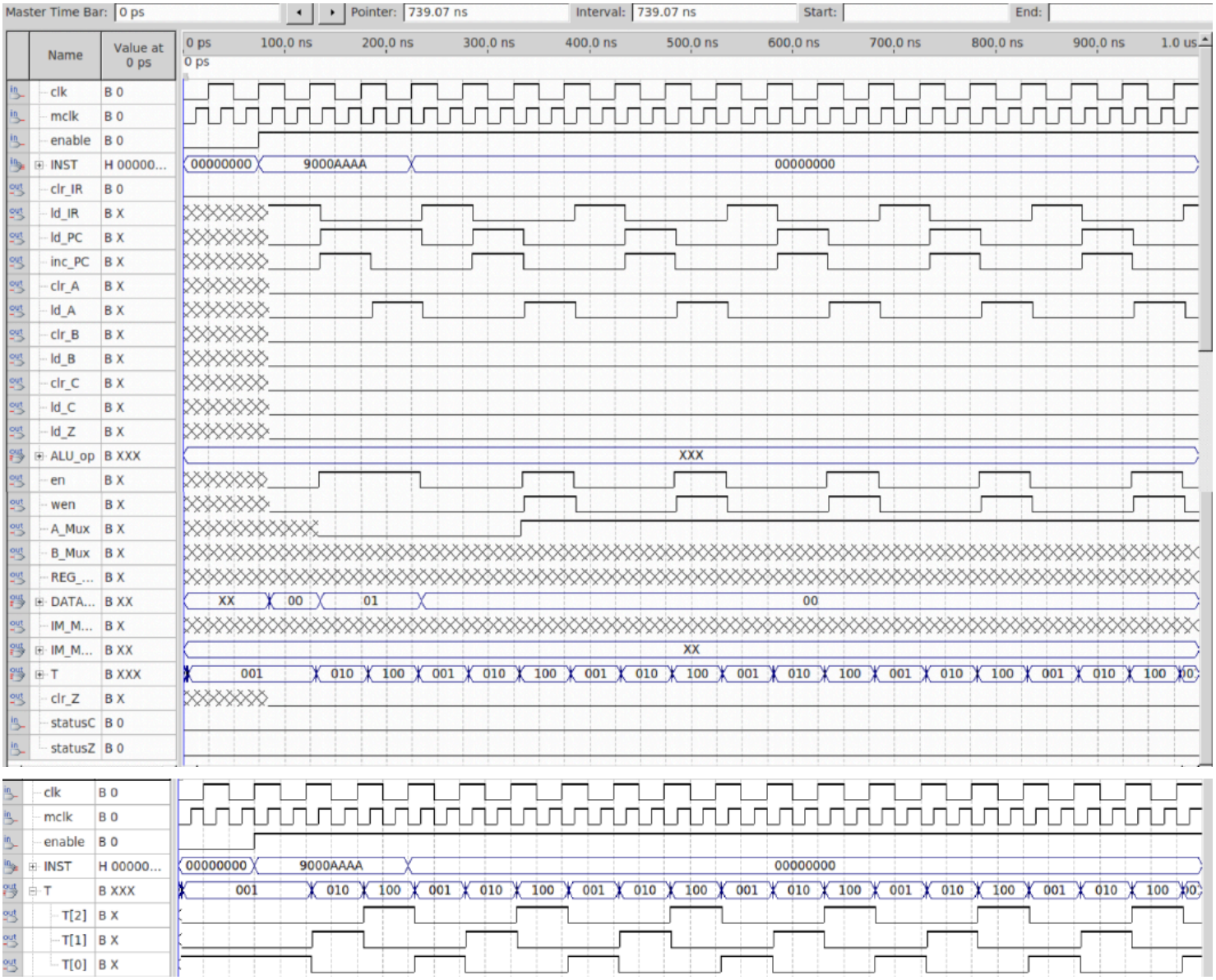
INCA:


Master Time Bar: 0 ps
Pointer: 558.06 ns
Interval: 558.06 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
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_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
7100AAAA
00000000
XXX
010
000
XX
00
10
00
XX
01
001
010
100
T[2]
T[1]
T[0]

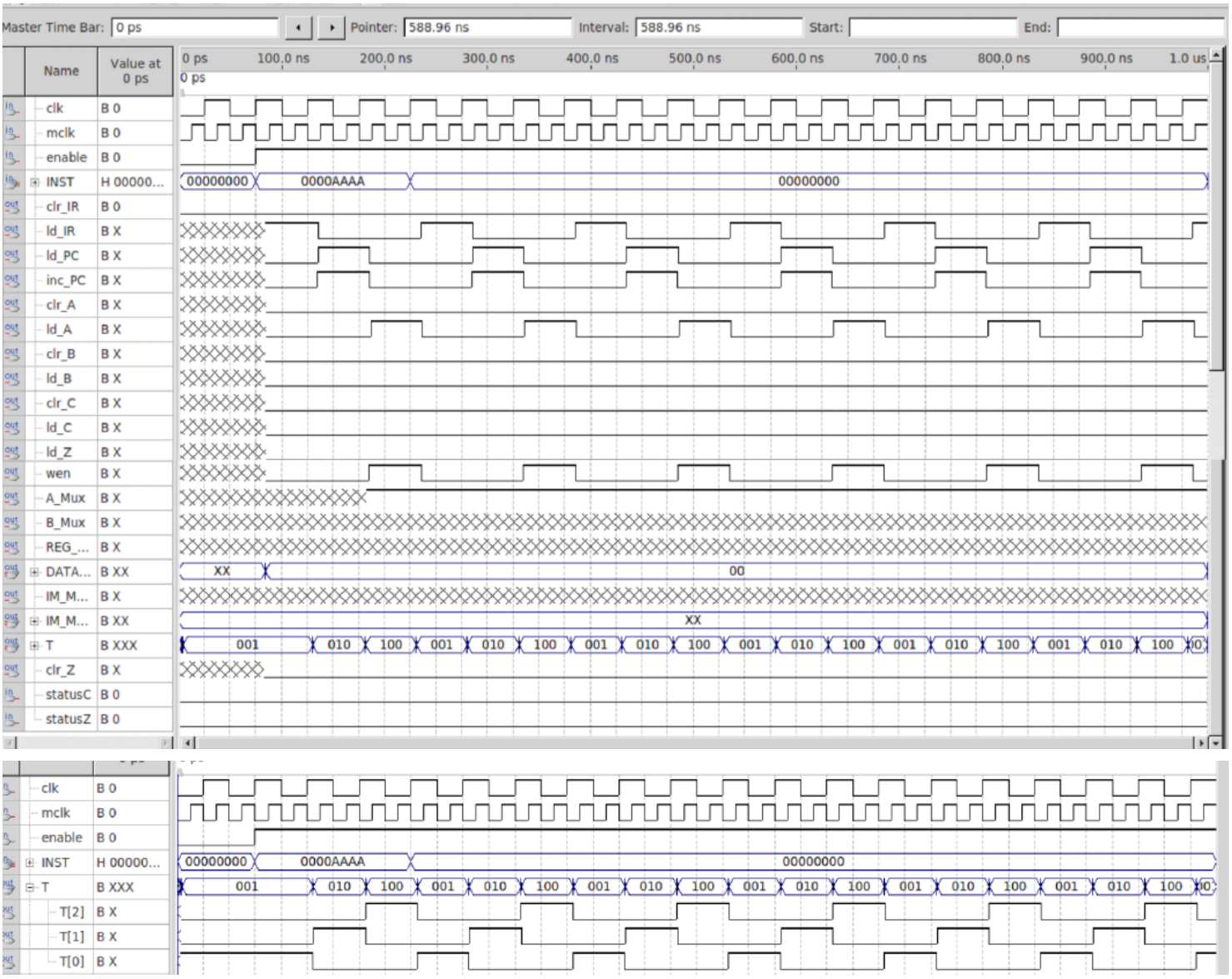
JMP:


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
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_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
5000AAAA
00000000
XXX
XX
00
XX
001
010
100
T[2]
T[1]
T[0]

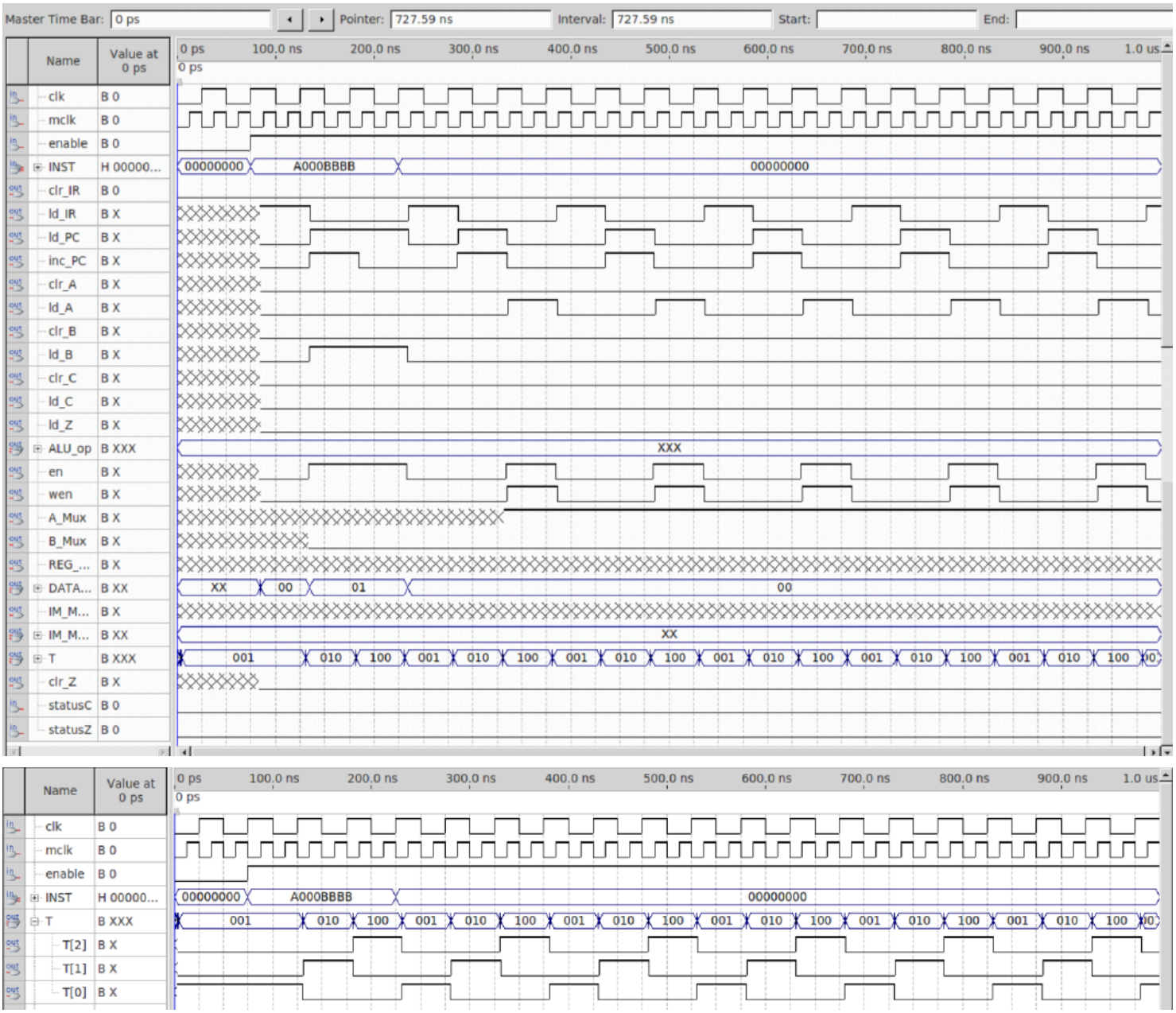
LDA:


Master Time Bar: 0 ps
Pointer: 739.07 ns
Interval: 739.07 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
9000AAAA
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
01
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0
T[2] B X
T[1] B X
T[0] B X

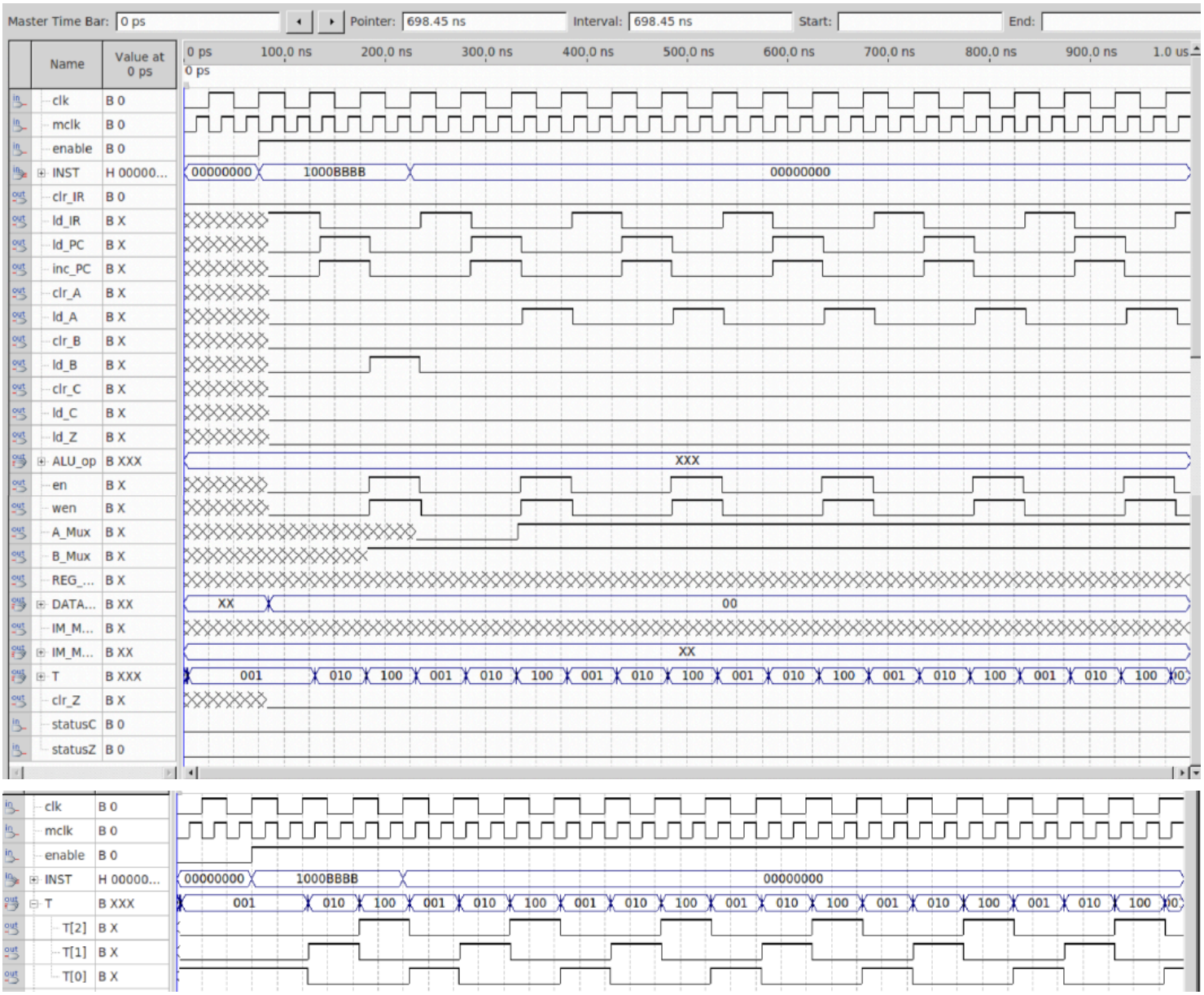
LDAI:


Master Time Bar: 0 ps
Pointer: 588.96 ns
Interval: 588.96 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
0000AAAA
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0
T[2] B X
T[1] B X
T[0] B X

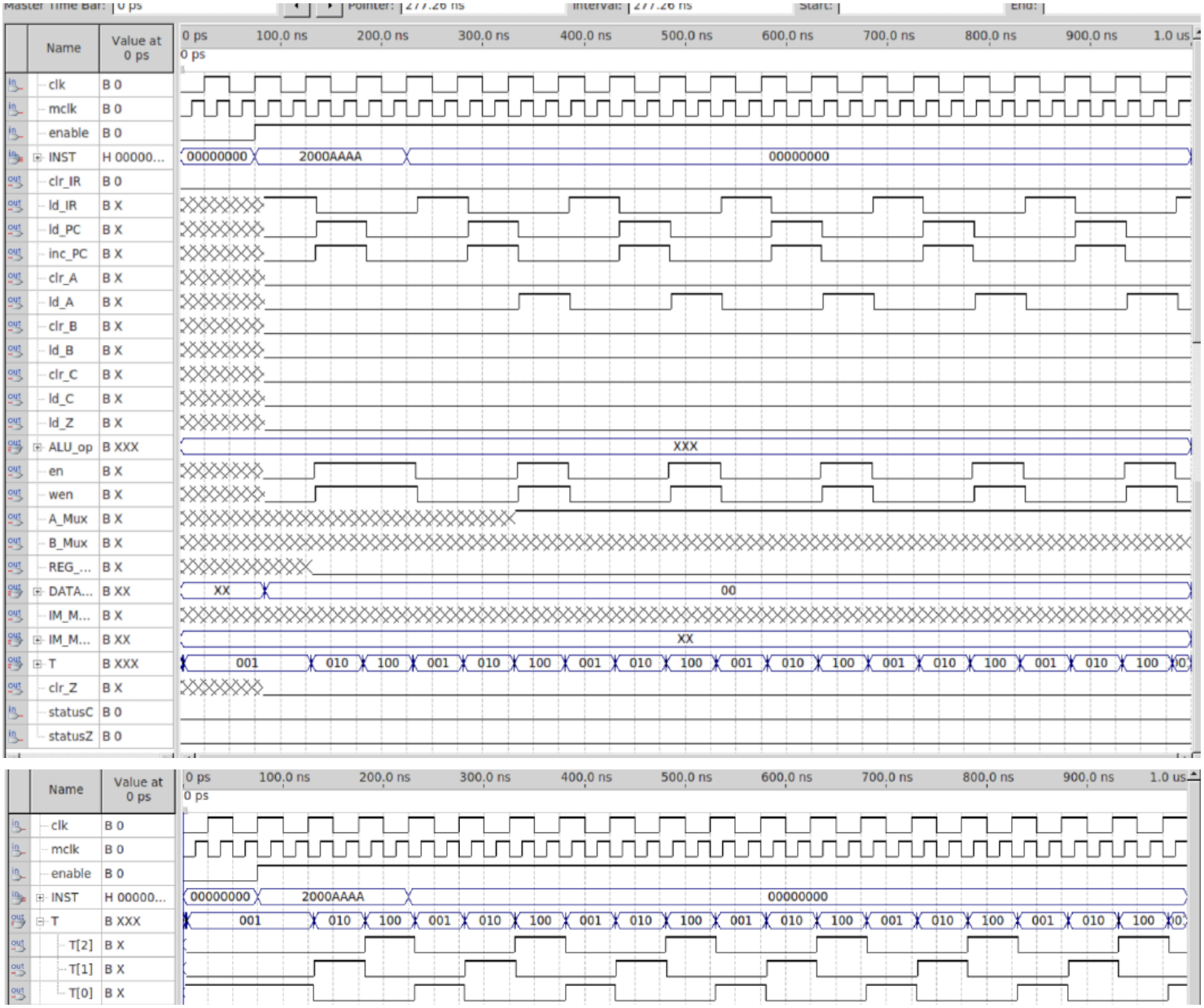
LDB:


Master Time Bar: 0 ps
Pointer: 727.59 ns
Interval: 727.59 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
IM_M... B X
IM_M... B XX
T B XXX
clr_Z B X
statusC B 0
statusZ B 0
00000000
A000BBBB
00000000
XXX
XX
00
01
00
XX
001
010
100
clk B 0
mclk B 0
enable B 0
INST H 00000...
T B XXX
T[2] B X
T[1] B X
T[0] B X

LDBI:

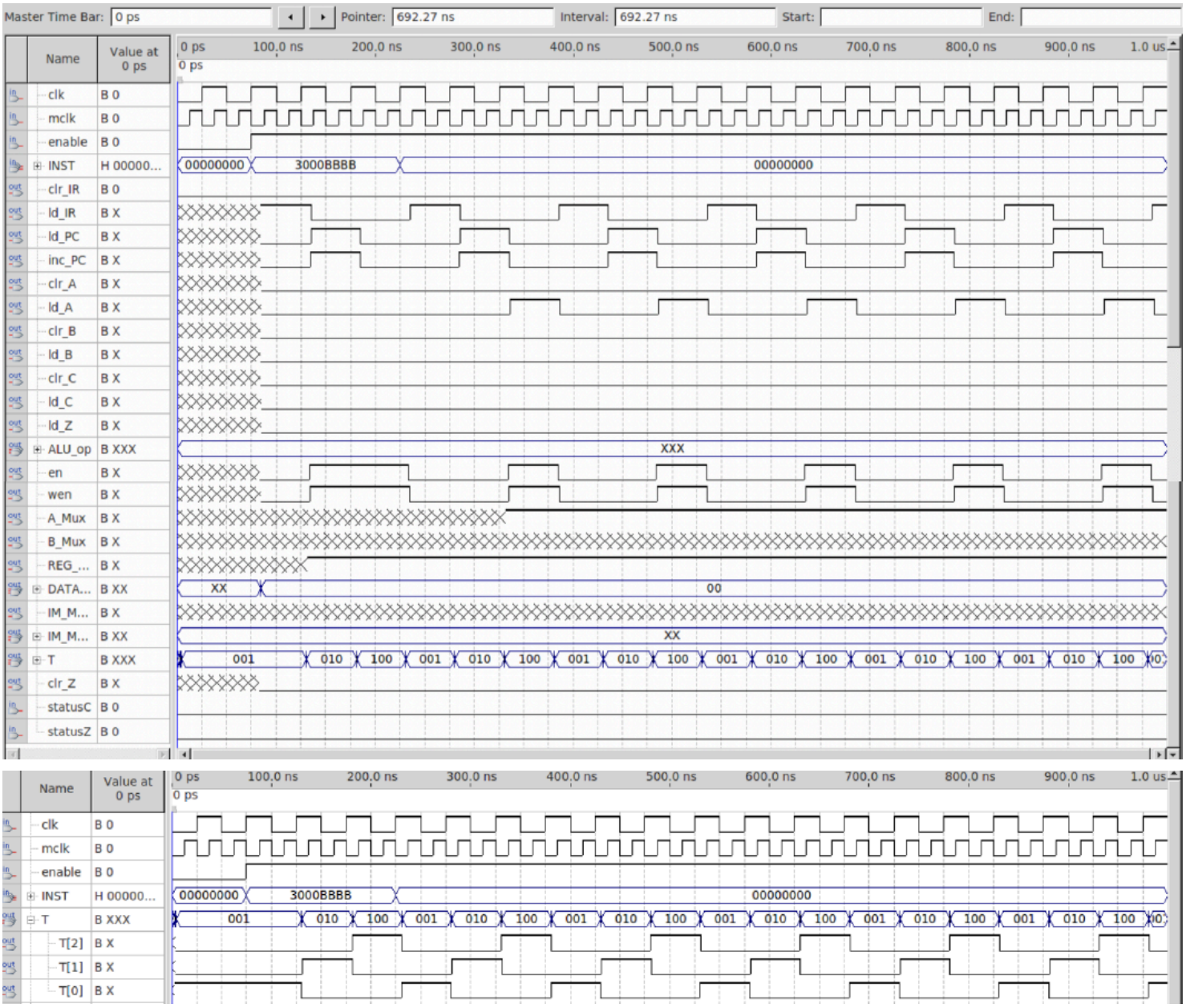

Master Time Bar: 0 ps
Pointer: 698.45 ns
Interval: 698.45 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
IM_M... B X
IM_M... B XX
T B XXX
clr_Z B X
statusC B 0
statusZ B 0
00000000
1000BBBB
00000000
XXX
XX
00
XX
001
010
100
clk B 0
mclk B 0
enable B 0
INST H 00000...
T B XXX
T[2] B X
T[1] B X
T[0] B X

STA:


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
2000AAAA
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 0
clr_Z B X
statusC B 0
statusZ B 0

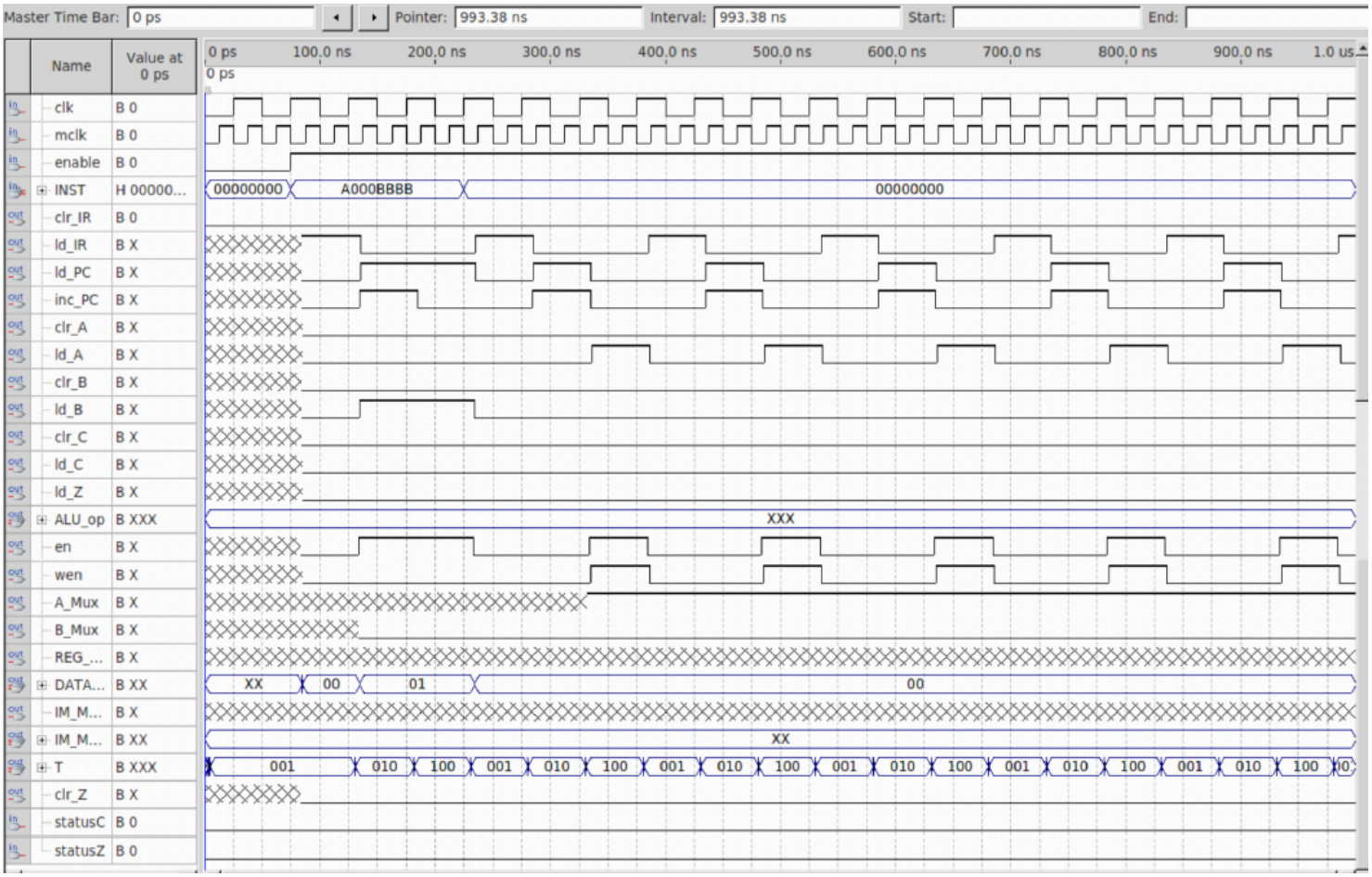
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
2000AAAA
00000000
T B XXX
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 0
T[2] B X
T[1] B X
T[0] B X

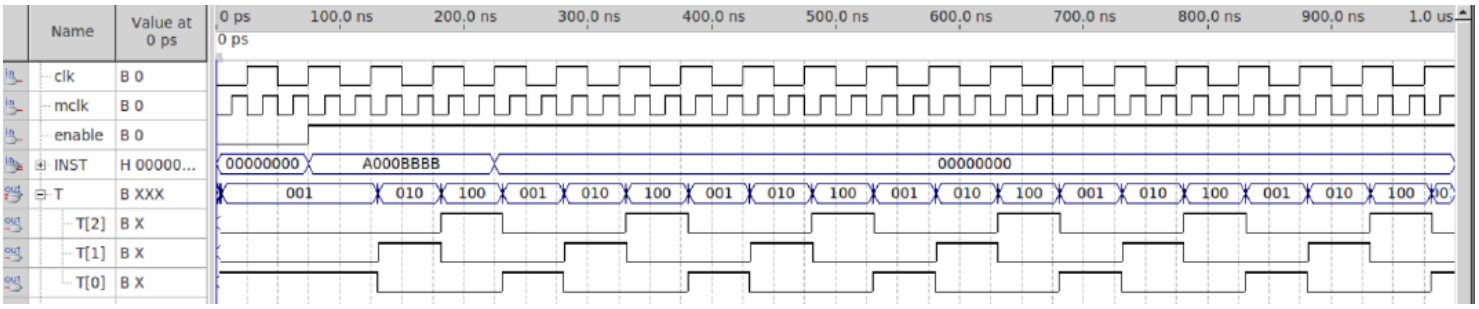
STB:


Master Time Bar: 0 ps
Pointer: 692.27 ns
Interval: 692.27 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
3000BBBB
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100
clr_Z B X
statusC B 0
statusZ B 0

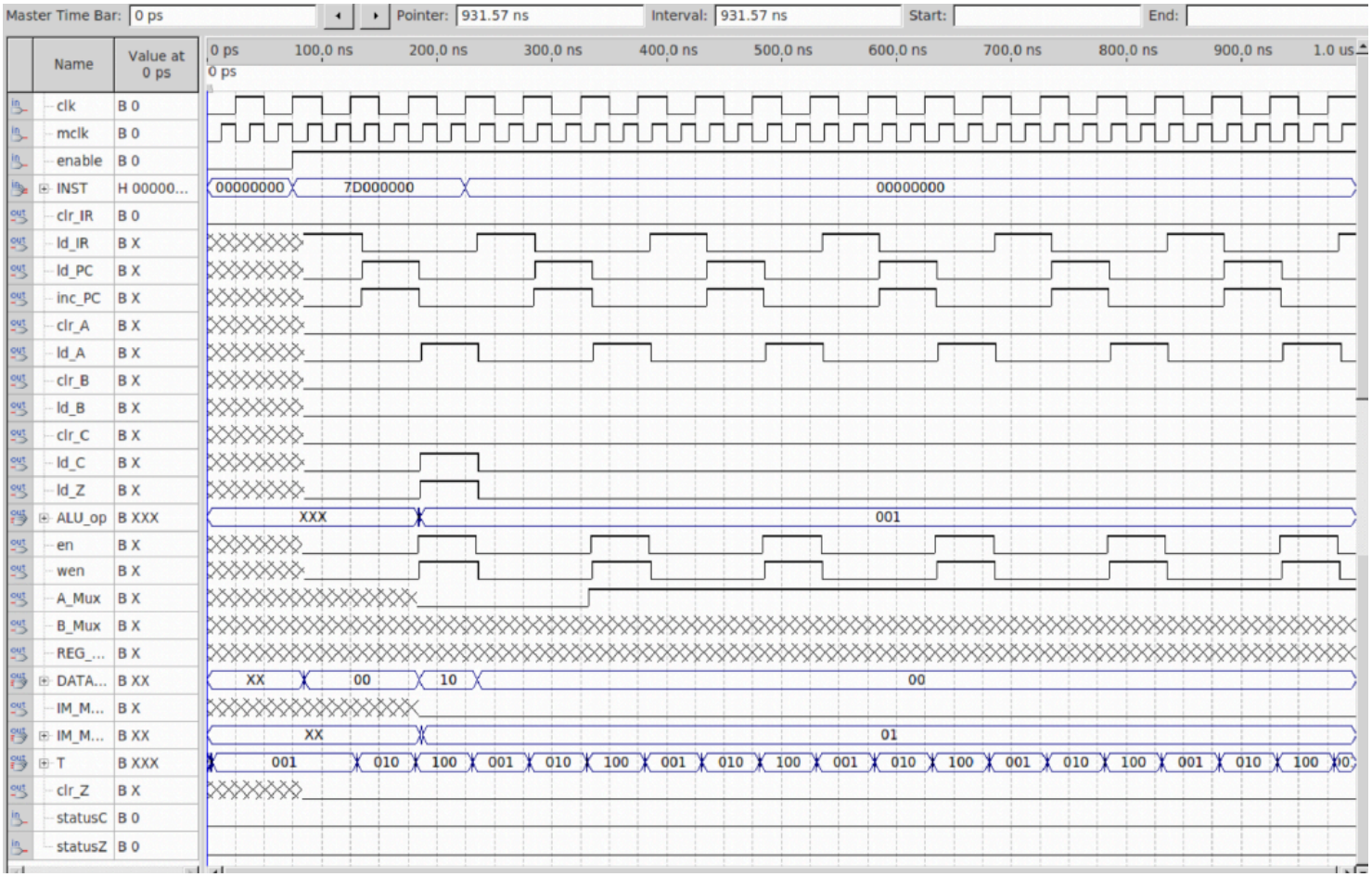
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
3000BBBB
00000000
T B XXX
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100
T[2] B X
T[1] B X
T[0] B X

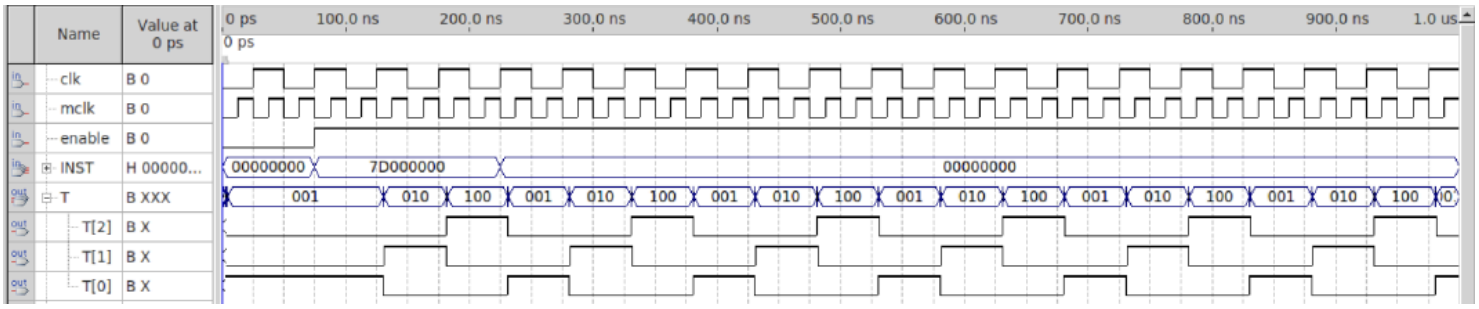
LUI:


Master Time Bar: 0 ps
Pointer: 993.38 ns
Interval: 993.38 ns
Start:
End:
Name
Value at 0 ps
clk B 0
mclk B 0
enable B 0
INST H 00000...
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
IM_M... B X
IM_M... B XX
T B XXX
clr_Z B X
statusC B 0
statusZ B 0
00000000
A000BBBB
00000000
XXX
XX
00
01
00
XX
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 00

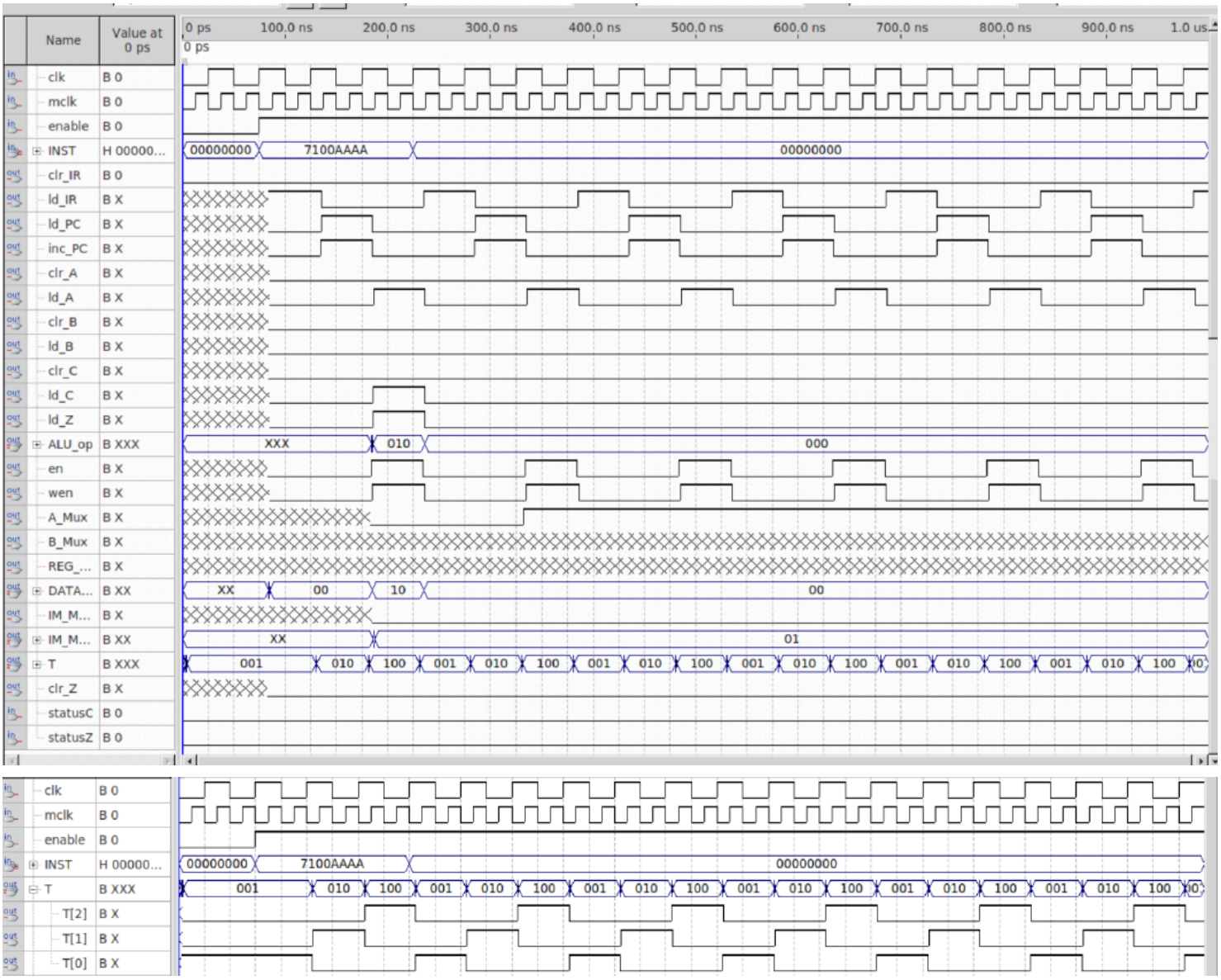

Name
Value at 0 ps
clk B 0
mclk B 0
enable B 0
INST H 00000...
T B XXX
T[2] B X
T[1] B X
T[0] B X
00000000
A000BBBB
00000000
001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 001 010 100 00

ORI:

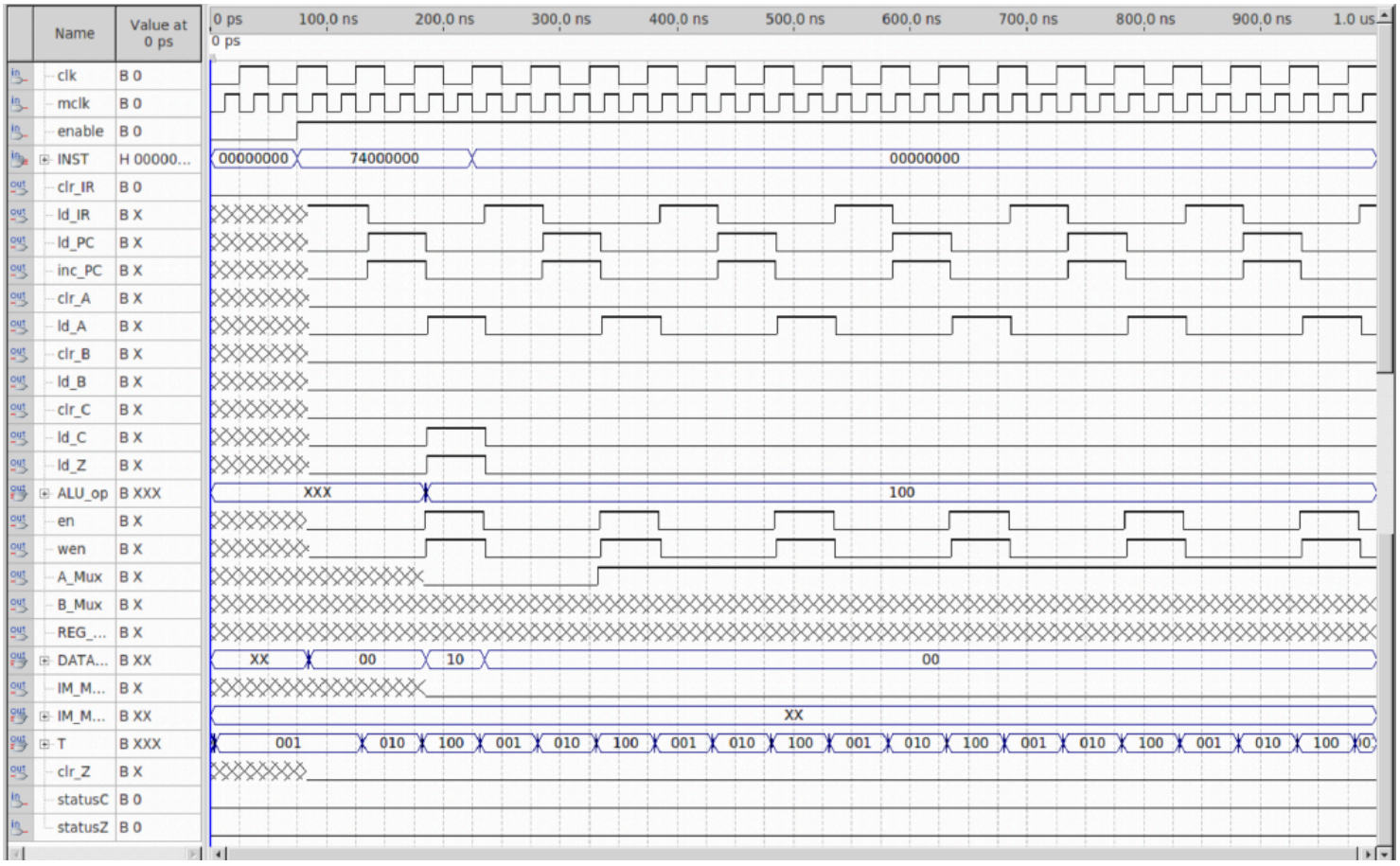

Master Time Bar: 0 ps
Pointer: 931.57 ns
Interval: 931.57 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
7D000000
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
001
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
10
00
IM_M... B X
IM_M... B XX
XX
01
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
7D000000
00000000
T B XXX
001
010
100
T[2] B X
T[1] B X
T[0] B X

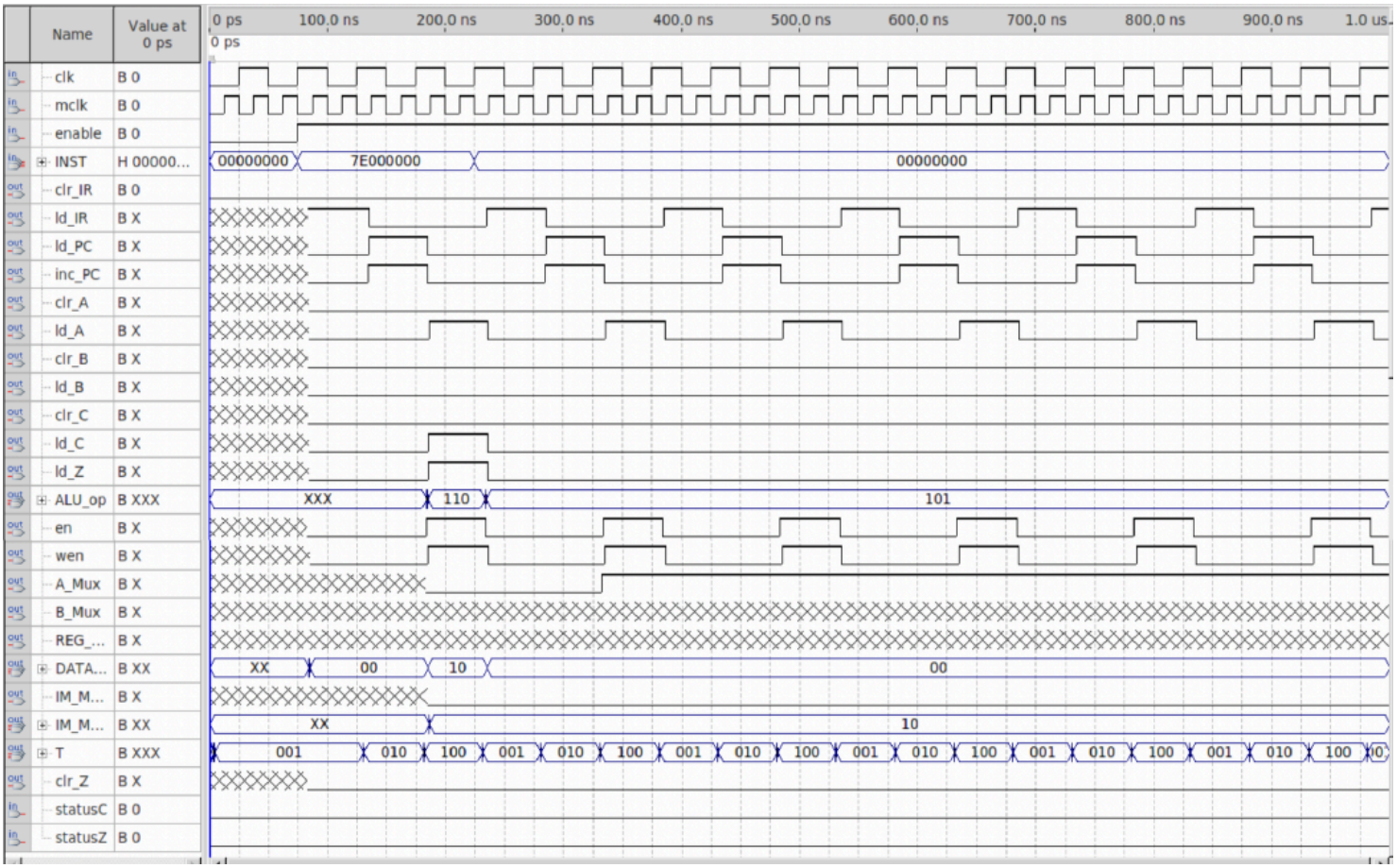
SUB:

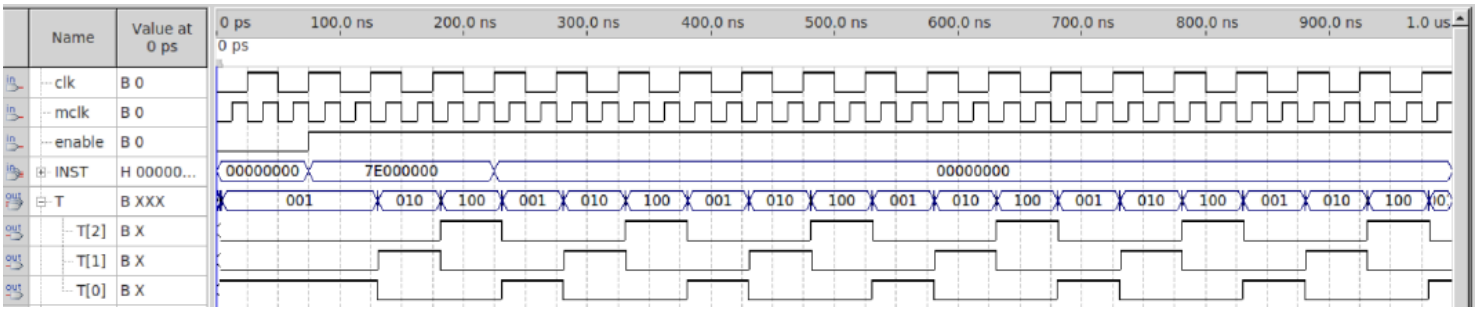

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
7100AAAA
00000000
XXX
010
000
XX
00
10
00
XX
01
001
010
100
T[2]
T[1]
T[0]

ROL:

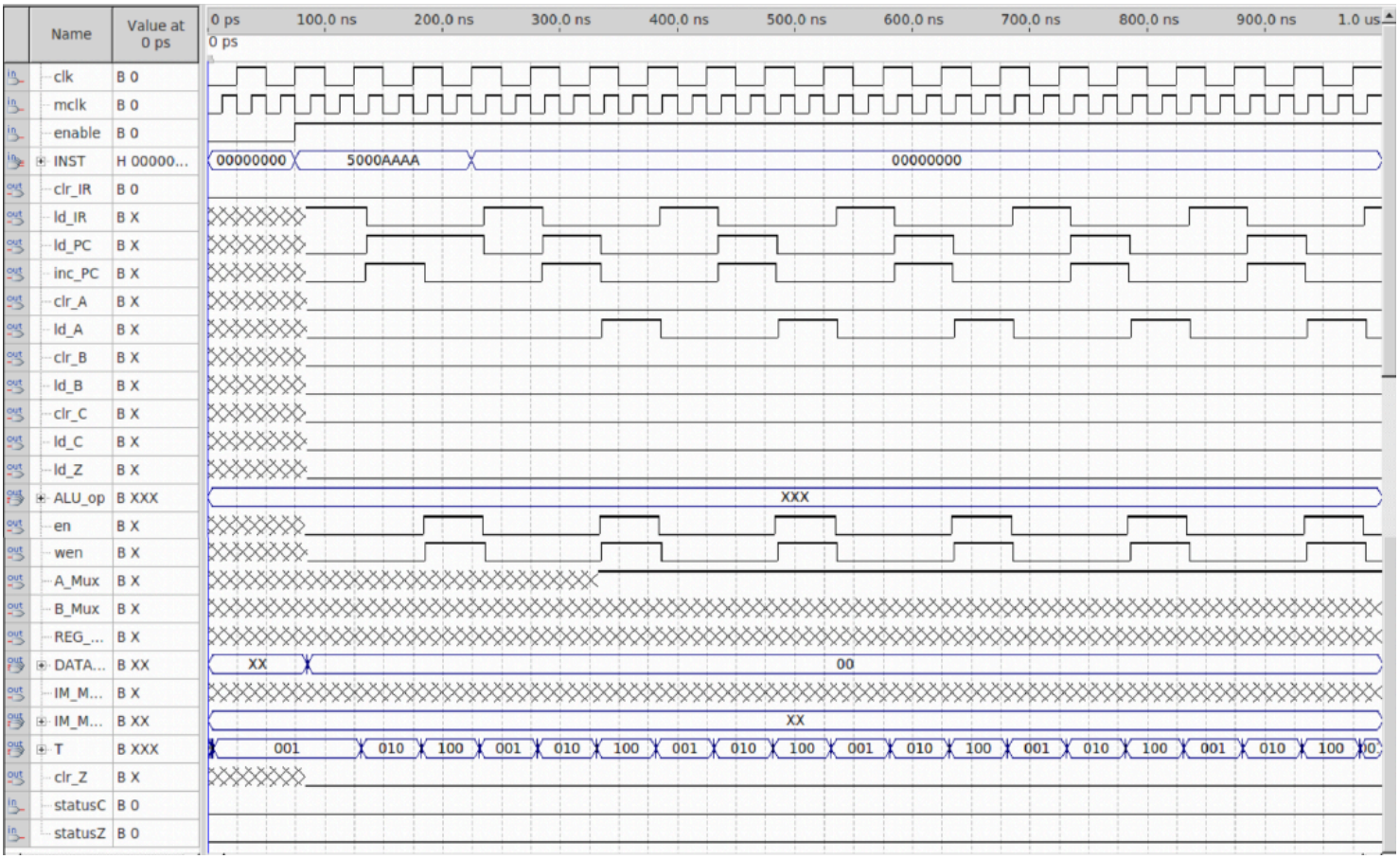

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
clr_IR
ld_IR
ld_PC
inc_PC
clr_A
ld_A
clr_B
ld_B
clr_C
ld_C
ld_Z
ALU_op
en
wen
A_Mux
B_Mux
REG_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
74000000
00000000
XXX
100
XX
00
10
00
XX
001
010
100
T[2]
T[1]
T[0]

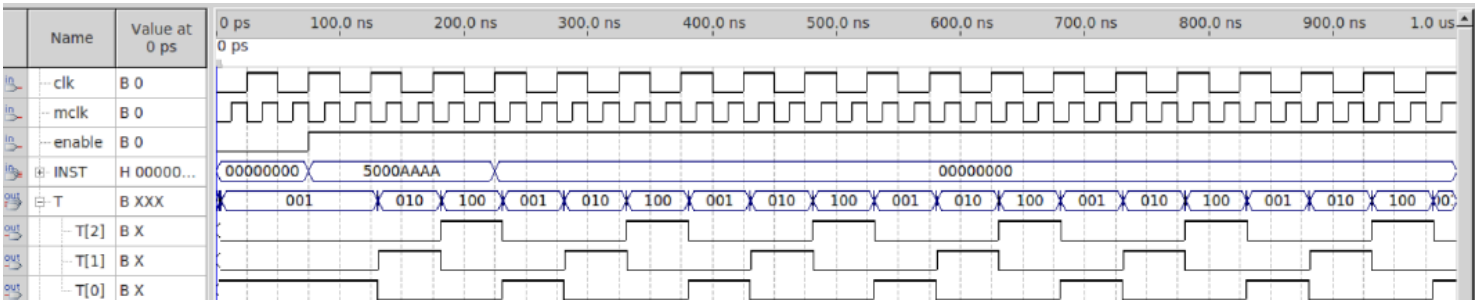
ROR:


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
7E000000
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
110
101
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
10
00
IM_M... B X
IM_M... B XX
XX
10
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0

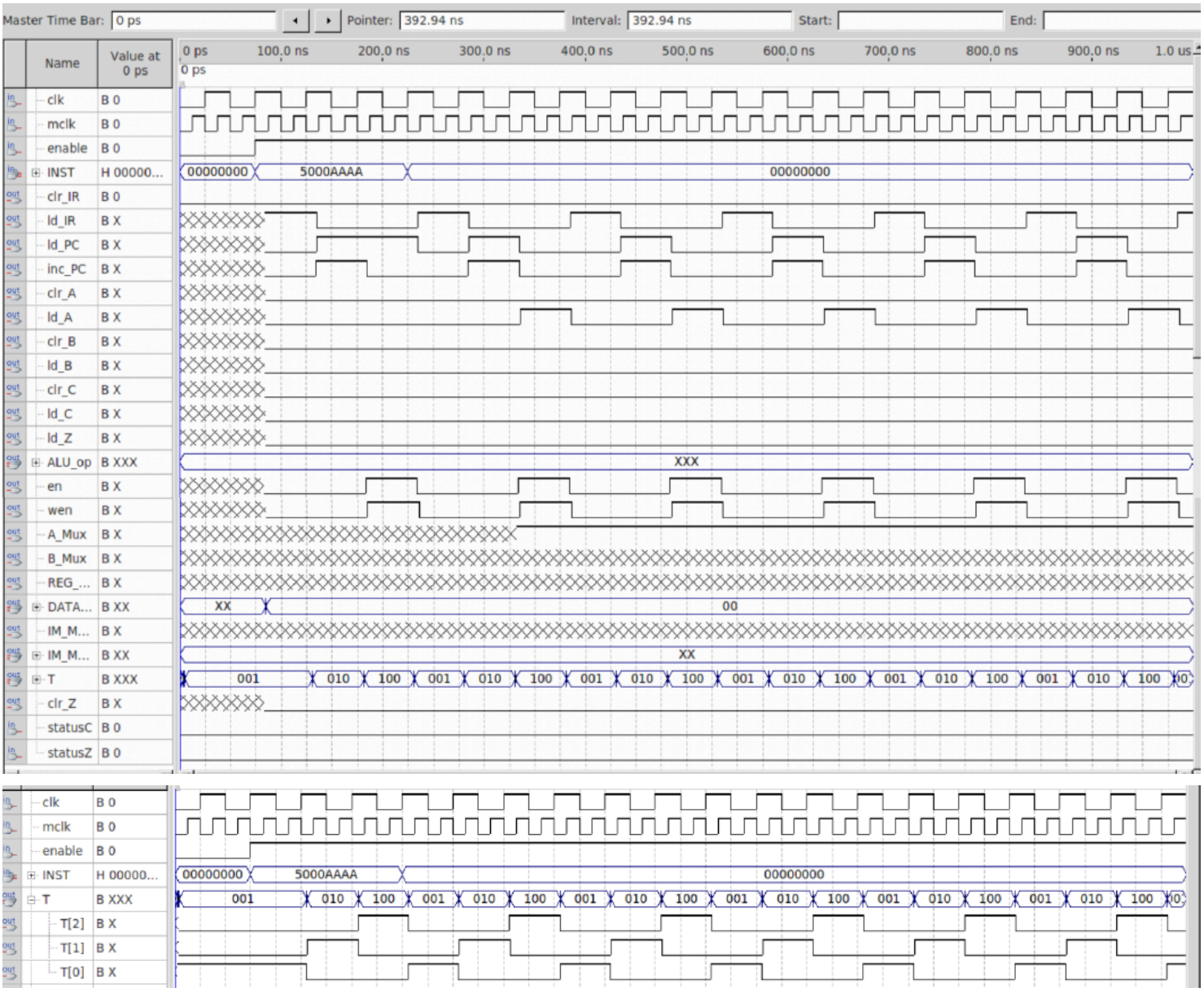

Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
7E000000
00000000
T B XXX
001
010
100
T[2] B X
T[1] B X
T[0] B X

BEQ:


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
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_...
DATA...
IM_M...
IM_M...
T
clr_Z
statusC
statusZ
00000000
5000AAAA
00000000
XXX
XX
00
XX
001
010
100


Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk
mclk
enable
INST
T
T[2]
T[1]
T[0]
00000000
5000AAAA
00000000
001
010
100

BNE:


Master Time Bar: 0 ps
Pointer: 392.94 ns
Interval: 392.94 ns
Start:
End:
Name
Value at 0 ps
0 ps
100.0 ns
200.0 ns
300.0 ns
400.0 ns
500.0 ns
600.0 ns
700.0 ns
800.0 ns
900.0 ns
1.0 us
clk B 0
mclk B 0
enable B 0
INST H 00000...
00000000
5000AAAA
00000000
clr_IR B 0
ld_IR B X
ld_PC B X
inc_PC B X
clr_A B X
ld_A B X
clr_B B X
ld_B B X
clr_C B X
ld_C B X
ld_Z B X
ALU_op B XXX
XXX
en B X
wen B X
A_Mux B X
B_Mux B X
REG_... B X
DATA... B XX
XX
00
IM_M... B X
IM_M... B XX
XX
T B XXX
001
010
100
clr_Z B X
statusC B 0
statusZ B 0
T[2] B X
T[1] B X
T[0] B X