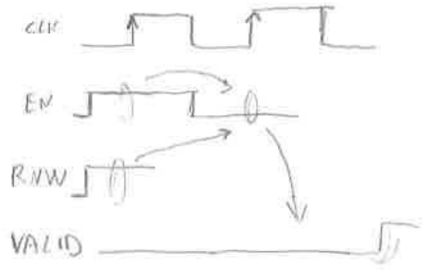
EXAM 2015

(1.1)



Using the first assumption:
AG(EN and RNW => AX(!EN and A(!EN U VALID)))

Using the second assumption:
AG(EN and RNW => AX(!EN and A(!EN W VALID)))

(1.2)

BECAUSE WHEN A REGISTER CHANGE ITS VALUE IT USES SOME POWER TO LOAD THE CAPACITORS INSIDE IT

(1.3)

IN ORDER TO SYNTHETIZE A VHDL CIRCUIT WITH THAT SPECIFICATIONS YOU NEED AT LEAST ONE PROCESS. A PROCESS IS ABLE TO MODEL A COMBINATIONAL CIRCUIT, A SERIES OF FLIP-FLOP OR A COMBINATIONAL CIRCUIT FOLLOWED BY A SERIES O REGISTER.

THE SPECIFICATION SAYS THAT THE OUTPUT IS TAKEN DIRECTLY FROM THE REGISTER SO A SECOND PROCESS FOR MODIFY THE STATE IN ORDER TO COMPUTE THE OUTPUT IS NOT NEEDED.

ANOTHER PROCESS (OR A SIMPLE CONCURRENT STATEMENT) IS NEEDED ONLY IF THE STATE OF THE REGISTER IS USED TO COMPUTE THE NEXT STATE OR IF THE CIRCUIT BEHIND THE OUTPUT REGISTERS IS NOT COMPLETELY COMBINATIONAL (THIS IS NOT SPECIFIED).

(1.4)

SLIDE 93 - VHDL CHAPTER

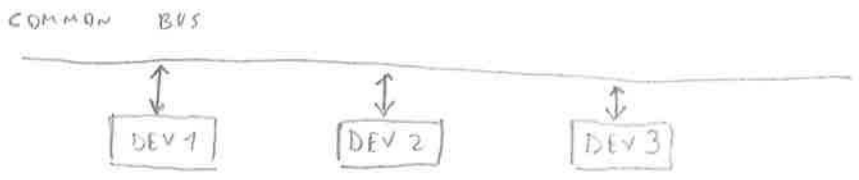
- VHDL RESOLVED TYPES ARE PARTICULAR SUB TYPE WITH A RESOLUTION FUNCTION ASSOCIATED. WHENEVER A SIGNAL IS DRIVEN BY TWO OR MORE SIGNAL THE RESOLUTION FUNCTION IS USED TO DETERMINATE THE VALUE TO ASSIGN.

  - EXAMPLE FROM LIBRARY IEEE.STD_LOGIC 1164

```
SUBTYPE STD_LOGIC IS RESOLVED STD_ULOGIC;
                        ↑         ↑
                        |     UNRESOLVED TYPE USED TO
                        |     CREATE THE RESOLVED ONE
               NAME OF THE
               FUNCTION USED TO
               MAKE THE RESOLUTION

FUNCTION RESOLVED (...
  ...
```

- THESE TYPE ARE USED WHEN A COMMON BUS IS USED BY MORE DATA SOURCES FOR EXAMPLE WITH A TIME DIVISION POLICY.



WHEN ONE DEVICE USES THE BUS ALL THE OTHER SET THEIR OUTPUT TO HIGH IMPEDENCE "Z".

- IN ALL THE OTHER APPLICATION IS BETTER TO USE THE UNRESOLVED TYPES BECAUSE THEY ARE FASTER DURING THE SIMULATION AND THEY HELP YOU TO FIND BUGS IN YOUR DESIGN.

1.5

ERRORS

- "S" IS THE OUTPUT OF THE ADDER SO IT SHOULD NOT BE ADDED TO THE SENSITIVITY LIST.
  THE UNDESIDERABLE EFFECT OF THIS ERROR IS THAT WHEN "A" OR "B" CHANGES THE PROCESS IS EXECUTED IN ORDER TO COMPUTE THE NEW VALUE OF "S" BUT AFTER THAT THE PROCESS IS EXECUTED AGAIN WITHOUT ANY CHANGES ON THE OUTPUT.

- "ADD_NOT_SUB" IS AN INPUT TO THE SYSTEM SO IT SHOULD BE ADDED TO THE SENSITIVITY LIST OTHERWISE THE OUTPUT IS NOT RECOMPUTED WHEN IT CHANGES.
  A F-F is created to store the value of S when the ADD-NOT-SUB signal changes

- THE SUM CAN GENERATE OVERFLOW AND THIS THING IS NOT WELL MANAGED.

- ONLY 2 CASES OF THE SIGNAL "ADD_NOT_SUB" ARE DEFINED (2 CASES OUT OF 9) SO THE SYNTHETIZER WILL ADD A LATCH TO STORE "S" IN ORDER TO PROVIDE IT AS OUTPUT IN CASE OF NOT MANAGED VALUE OF "ADD_NOT_SUB".

FIXED CODE

```
SIGNAL ADD_NOT_SUB : STD_ULOGIC;
SIGNAL A, B        : SIGNED (31 DOWTO 0);
SIGNAL S           : SIGNED (32 DOWTO 0);
...
PROCESS (A, B)
VARIABLE VAR_A, VAR_B : SIGNED (32 DOWTO 0);
BEGIN
   VAR_A := A(31) & A;  -- SIGN EXTENSION ON 33 BIT
   VAR_B := B(31) & B;
   IF ADD_NOT_SUB = '1' THEN
        S <= VAR_A + VAR_B;
   ELSIF ADD_NOT_SUB = '0' THEN
        S <= VAR_A - VAR_B;
   ELSE
        S <= (OTHERS => '0');
END PROCESS;
```

②

(2.1)

- ASSUMPTION : NUMBERS IN MEMORY ARE INTERPRETATED AS SIGNED 2'S COMPLEMENT, ADDRESS AS UNSIGNED

- INSTRUCTION 1 : BLOCK THE EXECUTION ON THIS FIRST INSTRUCTION UNTIL THE VALUE AT THE ADDRESS 0XFFFFFFFC IS LESS THAN ZERO.

- INSTRUCTION 2 : IF THE VALUE AT ADDRESS 0X21FA7AA8 IS LESS THAN ZERO JUMPS TO THE INSTRUCTION AT ADDRESS 0XAB2EE5D0

- INSTRUCTION 3 : I DO NOT RECOGNIZE ANY PARTICULAR PATTERN, ONLY THE NORMAL INSTRUCTION

$\text{MEM}[\overset{B}{\text{0XE8022DCF}}] = \text{MEM}[\overset{B}{\text{0XE8022DCF}}] - \text{MEM}[\overset{A}{\text{0XEFC51ADD}}]$

$\text{IF } (\text{MEM}[\underset{B}{\text{0XE8022DCF}}] <= 0) \text{ GO TO } \underset{C}{\text{0XD970380B}}$

- INSTRUCTION 4 : SAME AS BEFORE BUT WITH DIFFERENT ADDRESSES

A = 0X7B795D18 B = 0XC9DDE8E6 C = 0X54D72538

2.2

ADDRESS	VALUE
0X 00000000	Z
0X 00000004	Z
0X 00000008	0X0000000C
0X0000000C	TEMP
0X00000010	TEMP
0X00000014	0X00000018
0X00000018	X
0X0000001C	TEMP
0X00000020	0X00000024
0X00000024	Y
0X00000028	TEMP
0X0000002C	0X00000030
0X00000030	TEMP
0X00000034	Z
0X00000038	0X0000003C

Z:= 0;

TEMP := 0;

TEMP := TEMP - X

TEMP := TEMP - Y

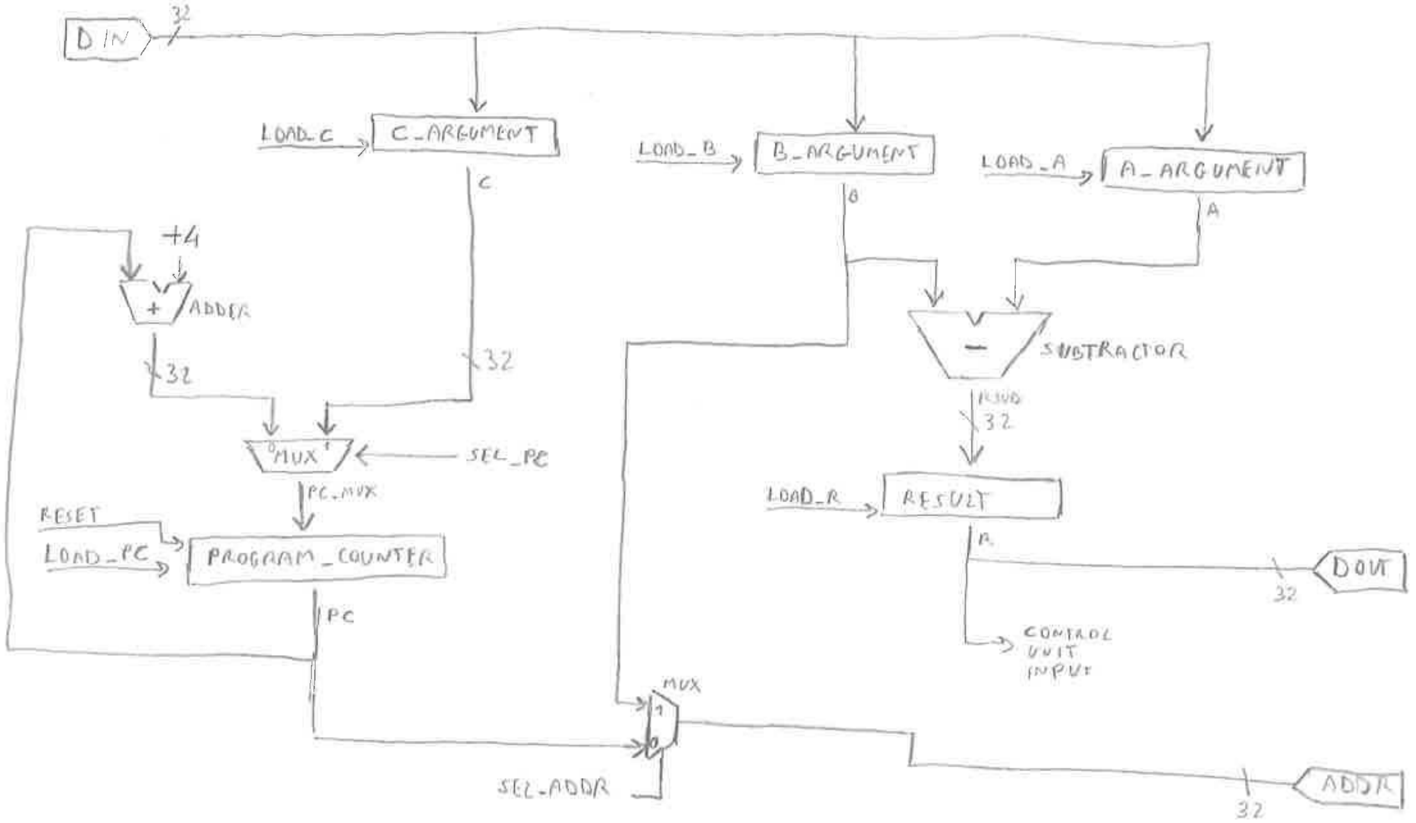
Z : = Z - TEMP

ADDRESS SPACE OF DATA

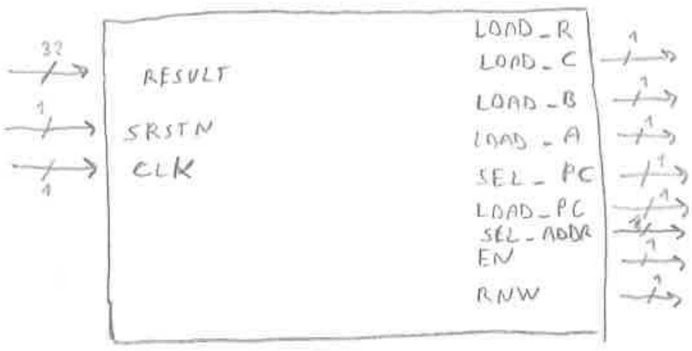
ADDRESS	VALUE
X	INTEGER_1
...	
Y	INTEGER_2
...	
Z	NOT INIZIALIZED
TEMP	NOT INIZIALIZED

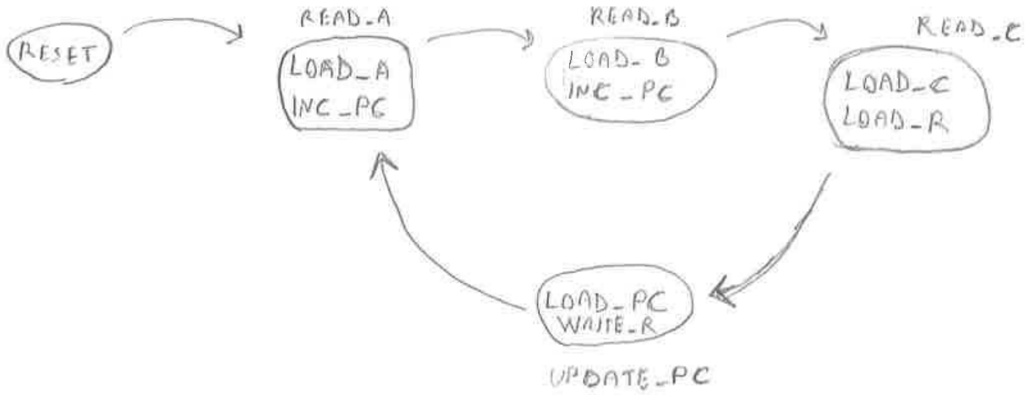
2.3

DATA PATH DESIGN BLOCK



CONTROL UNIT





2.4

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY SUBLEQ_CPU IS
    PORT(
        SRSTN, CLK          : IN  STD_ULOGIC;
        EN, RNW             : OUT STD_ULOGIC;
        ADDR, DOUT          : OUT STD_ULOGIC_VECTOR (31 DOWNTO 0);
        DIN                 : IN  STD_ULOGIC_VECTOR (31 DOWNTO 0)
    )
END SUBLEQ CPU;

ARCHITECTURE ARC OF SUBLEQ_CPU IS

    SIGNAL LOAD_R, LOAD_C, LOAD_B, LOAD_A, SEL_PC, LOAD_PC, SEL_ADDR   : STD_ULOGIC;
    SIGNAL A, B, C, R, R_SUB                       : SIGNED (31 DOWNTO 0);
    SIGNAL PC, PC_MUX                              : UNSIGNED (31 DOWNTO 0);
    TYPE STATE_TYPE IS (RESET, READ_A, READ_B, READ_C, UPDATE_PC);
    SIGNAL STATE          : STATE_TYPE;

  BEGIN

     -- SUBTRACTOR
     R_SUB <= B - A;
     -- MULTIPLEXER OF PROGRAM COUTER
     PC_MUX <= PC + 4      WHEN  SEL_PC = '0'  ELSE
               C           WHEN  SEL_PC = '1'  ELSE
               (OTHERS => '0');
     -- MULTIPLEXER OF ADDRESS
     ADDR <= PC WHEN SEL_ADDR = '0' ELSE   B WHEN SEL_ADDR = '1'  ELSE  (OTHERS => '0');

     REGISTERS : PROCESS (CLK)
        BEGIN
          IF RISING_EDGE (CLK) THEN
             IF SRSTN = '0' THEN
                PC <= (OTHERS => '0');
             ELSIF LOAD_PC = '1' THEN
                PC <= PC_MUX;
             END IF;
             IF LOAD_C = '1' THEN
                C <= DIN;
             END IF;
             IF LOAD_B = '1' THEN
                B <= DIN;
             END IF;
```

```
            IF   LOAD_A  ='1'   THEN
                 A <=  DIN;
            END   IF;

            IF   LOAD_R  ='1'  THEN
                 R <= DIN
            END  IF;

        END IF;
END  PROCESS  REGISTERS;

STATE_UPDATE : PROCESS (CLK)


       BEGIN
       IF  RISING_EDGE (CLK) THEN
           IF SRSTN  =  '0'   THEN
              STATE  <=  RESET;
           ELSE
              CASE ( STATE )   IS

                   WHEN     RESET   =>
                        STATE  <=  READ_A;
                   WHEN  READ_A  =>
                        STATE  <=  READ_B;

                   WHEN  READ_B  =>
                        STATE  <=  READ_C;
                   WHEN  READ_C  =>
                        STATE  <=  UPDATE_PC;
                   WHEN   OTHERS  =>
                        STATE  <=  RESET;

              END CASE;

           END  IF;
       END IF,

 END  PROCESS   STATE_UPDATE;
```

```
OUTPUT_EVALUATION : PROCESS (STATE)

    BEGIN
            -- DEFAULT VALUES
            SEL_ADDR <= '0';
            LOAD_R   <= '0';
            LOAD_C   <= '0';
            LOAD_B   <= '0';
            LOAD_A   <= '0';
            SEL_PC   <= '0';
            LOAD_PC  <= '0';
            EN       <= '0';
            RNW      <= '1';

            CASE (STATE) IS
              WHEN  RESET  =>
                    EN <= '1';
                    LOAD_A  <= '1';
                    LOAD_PC <= '1';

              WHEN  READ_A =>
                    EN <= '1'
                    LOAD_B <= '1';
                    LOAD_PC <= '1';
              WHEN  READ_B =>
                    EN <= '1'
                    LOAD_C <= '1';
                    LOAD_R <= '1';
              WHEN  READ_C =>
                    SEL_ADDR <= '1';
                    RNW <= '0';
                    EN <= '1';
                    IF R <= TO_SIGNED (0 , 32) THEN
                        SEL_PC <= '1';
                    END IF;
                    LOAD_PC <= '1';
              WHEN  UPDATE_PC  =>
                    EN <= '1';
                    LOAD_A <= '1';
                    LOAD_PC <= '1';
          END CASE;
     END PROCESS; OUTPUT_EVALUATION;
END ARCHITECTURE ARC;
```

2.5

IN ORDER TO INCREASE THE PERFORMANCE OF THIS IMPLEMENTASION IS POSSIBLE TO INCREASE THE PARALLELISM OF THE RAM IN ORDER TO READ THE THREE ARGUMENT AT THE SAME TIME.