Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή: Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Συστήματα Μικροϋπολογιστών (6ο εξάμηνο)
1η Ομάδα Ασκήσεων

Δημήτριος Καλαθάς - el18016
Δημήτριος Καλέμης - el18152

Ασκήσεις Προσομοίωσης

Άσκηση 1

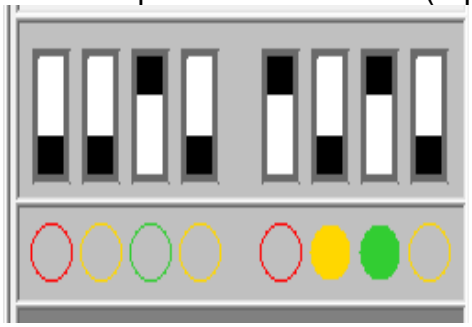
Αρχικό πρόγραμμα σε assembly	Πρόγραμμα για συνεχή λειτουργία
MVI C,08H LDA 2000H RAL JC 080DH DCR C JNZ 0805H MOV A,C CMA STA 3000H RST 1	START: MVI C,08H LDA 2000H LABEL2: RAL JC LABEL1 DCR C JNZ LABEL2 LABEL1: MOV A,C CMA STA 3000H JMP START END

Παρατηρήσεις

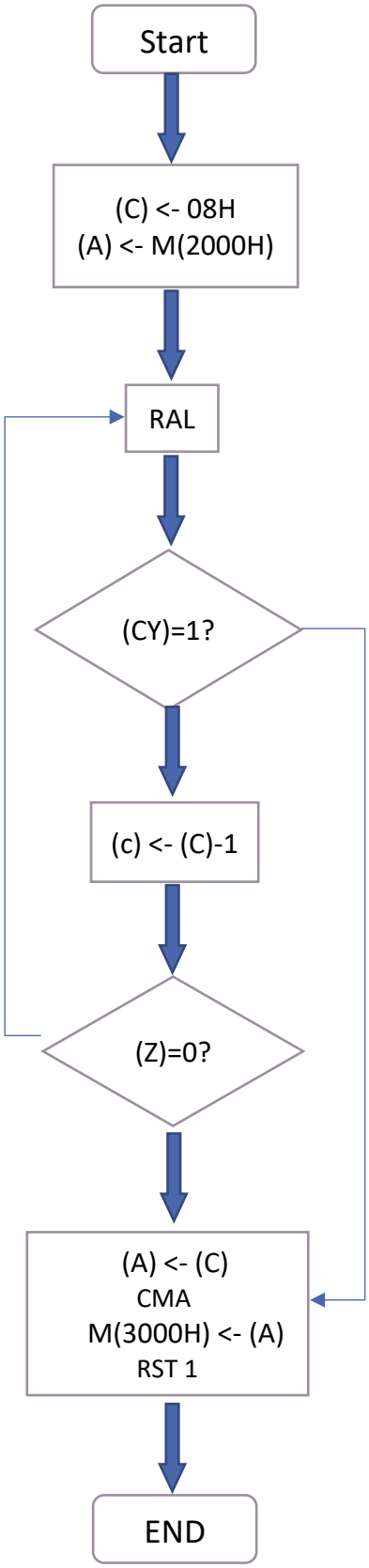
Το παραπάνω πρόγραμμα ανιχνεύει τον MSB των διακοπτών που είναι «πάνω» και εμφανίζει στα λαμπάκια τη θέση του (από αριστερά) σε δυαδική μορφή.

Παράδειγμα

Εδώ, ο πρώτος διακόπτης από δεξιά (MSB)που είναι πάνω είναι ο 5ος μετρώντας από αριστερά, οπότε εμφανίζεται στα λαμπάκια το διάδικό 5 (δηλαδή το 00000110).



Διάγραμμα Ροής



Άσκηση 2

Πρόγραμμα σε assembly

```
        IN 10H
        LXI B,1388H
        MVI E,01H
CHECK:
        CALL DELB           ;wait for ~1/2 sec
        LDA 2000H
        MOV D,A
        RRC                 ;Check if LSB = OFF
        JC CHECK            ;loop until LSB = ON
        MOV A,D
        RLC                 ;check if MSB = OFF
        JC LEFT             ;if MSB = OFF go LEFT
                            ;else go RIGHT
RIGHT:
        MOV A,E
        CMA
        STA 3000H
        CMA
        RRC                 ;Next led - Right
        MOV E,A
        JMP CHECK
LEFT:
        MOV A,E
        CMA
        STA 3000H
        CMA
        RLC                 ;Next led - Left
        MOV E,A
        JMP CHECK

        END
```

Άσκηση 3

Πρόγραμμα σε assembly

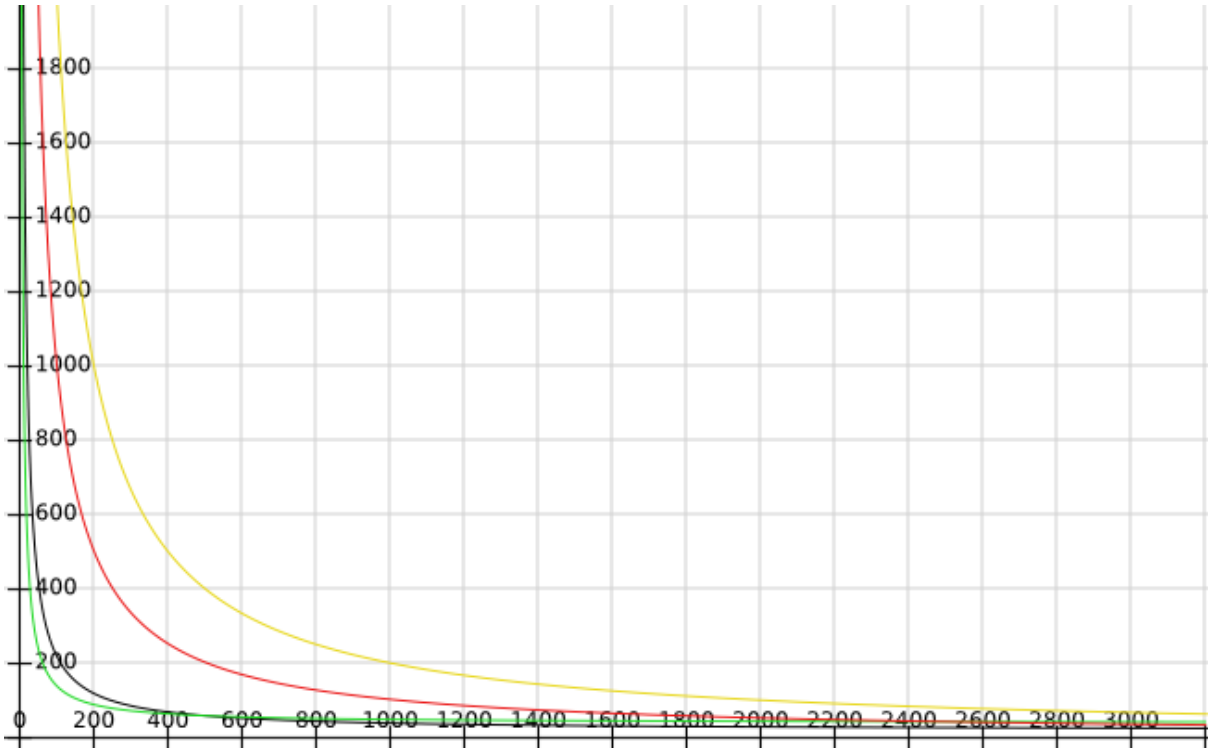
```
        LXI B,2710H

START:
        LDA 2000H
        CPI 64H             ;Compare with 100
        JNC MEG_99          ;Jump if A≥100
        MVI D,00H           ;Set a counter equal to 0

KATW_APO_100:
        SUI 0AH             ;Subtract 10
        INR D
        JNC KATW_APO_100    ;Jump if A still ≥10
        DCR D
        ADI 0AH
        MOV E,A             ;E gets Monades
        MOV A,D             ;A gets Dekades in 4 LSB
        RLC
        RLC
        RLC
        RLC                 ;A gets Dekades in 4 MSB
        ADD E               ;A gets also Monades in 4 LSB
        CMA
        STA 3000H           ;Store the result
        JMP START

MEG_99:
        CPI C8H             ;Compare with 200
        JNC MEG_199         ;Jump if A≥200
        MVI A,F0H           ;4 LSB leds open
        STA 3000H
        CALL DELB
        MVI A,FFH           ;4 LSB leds closed
        STA 3000H
        CALL DELB
        JMP START

MEG_199:
        MVI A,0FH           ;4 MSB leds open
        STA 3000H
        CALL DELB
        MVI A,FFH           ;4 MSB leds closed
        STA 3000H
        CALL DELB
        JMP START
        END
```

Άσκηση 4

Οι συναρτήσεις κόστους ανά τεμάχιο για κάθε τεχνολογία φαίνονται στον παρακάτω πίνακα

1η	2η	3η	4η
$\frac{20000+20x}{x}$	$\frac{10000+40x}{x}$	$\frac{100000+4x}{x}$	$\frac{200000+2x}{x}$

ενώ οι αντίστοιχες γραφικές παραστάσεις φαίνονται στο επόμενο διάγραμμα οπού στο οριζόντιο άξονα είναι τα τεμάχια και στον κάθετο άξονα το κόστος τους.



Εξισώνοντας τις εκφράσεις των καμπυλών ανά 2, βρίσκουμε τα σημεία τομής των καμπυλών μεταξύ τους

1η-2η	2η-3η	1η-3η	2η-4η	1η-4η	3η-4η
$x = 500$	$x = 2500$	$x = 5000$	$x = 5000$	$x = 10000$	$x = 50000$

Μελετώντας τον προηγούμενο πίνακα και το διάγραμμα των γραφικών παραστάσεων, εξάγουμε τα διαστήματα τιμών του αριθμού τεμαχίων που ελαχιστοποιούν το κόστος κατασκευής για κάθε τεχνολογία

$$0 < x < 500 : \textbf{2η}$$
$$500 < x < 5000 : \textbf{1η}$$
$$5000 < x < 50000 : \textbf{3η}$$
$$x > 50000 : \textbf{4η}$$

Παρατηρούμε ότι οι τεχνολογίες με υψηλό κόστος σχεδίασης γίνονται συμφέρουσες μόνο σε υψηλούς αριθμούς τεμαχίων.

Αν z το κόστος ανά IC για την τεχνολογία των FPGAs, τότε το συνολικό κόστος κατασκευής για τη 2η τεχνολογία θα είναι μικρότερο αυτού της 1ης όταν

$$\frac{10000 + (z+10)x}{x} < \frac{20000 + 20x}{x} \Rightarrow z < \frac{10000}{x} + 10$$

Για $z \leq 10$ το συνολικό κόστος κατασκευής της 2ης τεχνολογίας είναι πάντα μικρότερο από αυτό της 1ης, άρα για να αποκλειστεί η 1η τεχνολογία πρέπει το κόστος ανά IC της 2ης να είναι το πολύ **10€** ανά τεμάχιο.

Άσκηση 5

Α) Περιγραφή Verilog σε επίπεδο πυλών των F1, F2, F3, F4

F1=A(BC+D)+B'C'D	F2(A,B,C,D)=Σ(0,2,3,5,7,9,10,11,13,14)
module Circuit_F1 (A, B, C, D, F1); output F1; input A, B, C, D; wire Bnot, Anot, w1, w2, w3, w4; not G1 (Bnot, B); not G2 (Anot, A); and G3 (w1, B, C); or G4 (w2, w1, D); and G5 (w3, A, w2); and G6 (w4, D, Bnot, Anot); or G7 (F1, w3, w4); endmodule	primitive Circuit_F2 (A, B, C, D, F2); output F2; input A, B, C, D; table 0 0 0 0 : 1; 0 0 0 1 : 0; 0 0 1 0 : 1; 0 0 1 1 : 1; 0 1 0 0 : 0; 0 1 0 1 : 1; 0 1 1 0 : 0; 0 1 1 1 : 1; 1 0 0 0 : 0; 1 0 0 1 : 1; 1 0 1 0 : 1; 1 0 1 1 : 1; 1 1 0 0 : 0: 1 1 0 1 : 1; 1 1 1 0 : 1; 1 1 1 1 : 0; endtable endprimitive

F3=ABC+(A+BC)D+(B+C)DE	F4=A(B+CD+E)+BCDE
module Circuit_F3 (A, B, C, D, E, F3); output F3; input A, B, C, D, E; wire w1, w2, w3, w4, w5, w6, w7; and G1 (w1, A, B, C); and G2 (w2, B, C); or G3 (w3, w2, A); and G4 (w4, w3, D); or G5 (w5, C, B); and G6 (w6, E, D); and G7 (w7, w5, w6); or G8 (F3, w1, w4, w7); endmodule	module Circuit_F4(A, B, C, D, E, F4); output F4; input A, B, C, D, E; wire w1, w2, w3, w4; and G1 (w1, C, D); or G2 (w2, w1, B, E); and G3 (w3, w2, A); and G4 (w4, B, C, D, E); or (F4, w3, w4); endmodule

B) Οι ίδιες συναρτήσεις, με μοντελοποίηση ροής δεδομένων:

```
module Circuit_F1 (A, B, C, D, F1);
        output F1;
        input A, B, C, D;

        assign F1= (A&((B&C) | D)) | (~B&~C&D);
endmodule
```

```
module Circuit_F2 (A, B, C, D, F2);
        output F2;
        input A, B, C, D;

        assign F2=  (~A&~B&~C&~D)  |  (~A&~B&C&~D)  |  (~A&~B&C&D) |
        (~A&B&~C&D)  |  (~A&B&C&D)  |  (A&~B&~C&D)  |
        (A&~B&C&~D)  |  (A&~B&C&D)  |  (A&B&~C&D)  |  (A&B&C&~D);
endmodule
```

```
module Circuit_F3 (A, B, C, D, E, F3);
        output F3;
        input A, B, C, D, E;

        assign F3= (A&B&C) | (D&((B&C) | A)) | (D&E&(B | C));
endmodule
```

```
module Circuit_F4 (A, B, C, D, E, F4);
        output F4;
        input A, B, C, D, E;

        assign F4= (A&(B | (C&D) | E)) | (B&C&D&E);
endmodule
```

Άσκηση 6

(i)

a)

```
module Circuit_A (A, B, C, D, F);
        input A, B, C, D;
        output F;
        wire w, x, y, z, a, d;
        and (x, D, A, b);
        and (y, c ,A);
        and (w, z ,D);
        or (z, y, C);
        or (F, x, w);
        not (c, C);
        not (b, B);
endmodule
```

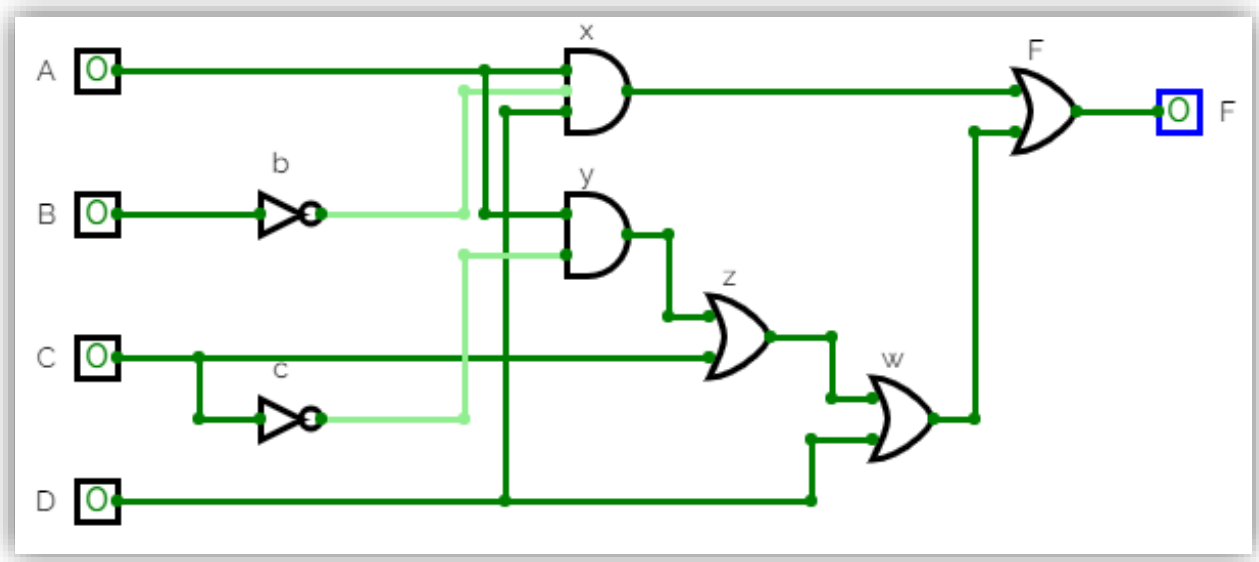


FIGURE 1: ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΓΙΑ ΤΟ Α

---

b)

```
module Circuit_B (F1, F2, F3, X, Y, W, Z);
        output F1, F2, F3;
        input X, Y, W, Z;
        or (F1, F2, F3);
        nor (F2, w1, w2, w3);
        or (F3, w4, w5);
        and (w1, w6, Z);
        and (w2, w6, w7, W);
        or (w3, w7, W, Z);
        not (w6, Y);
        not (w7, X);
        xnor (w4, Y, Z);
        xor (w5, X, W);
endmodule
```

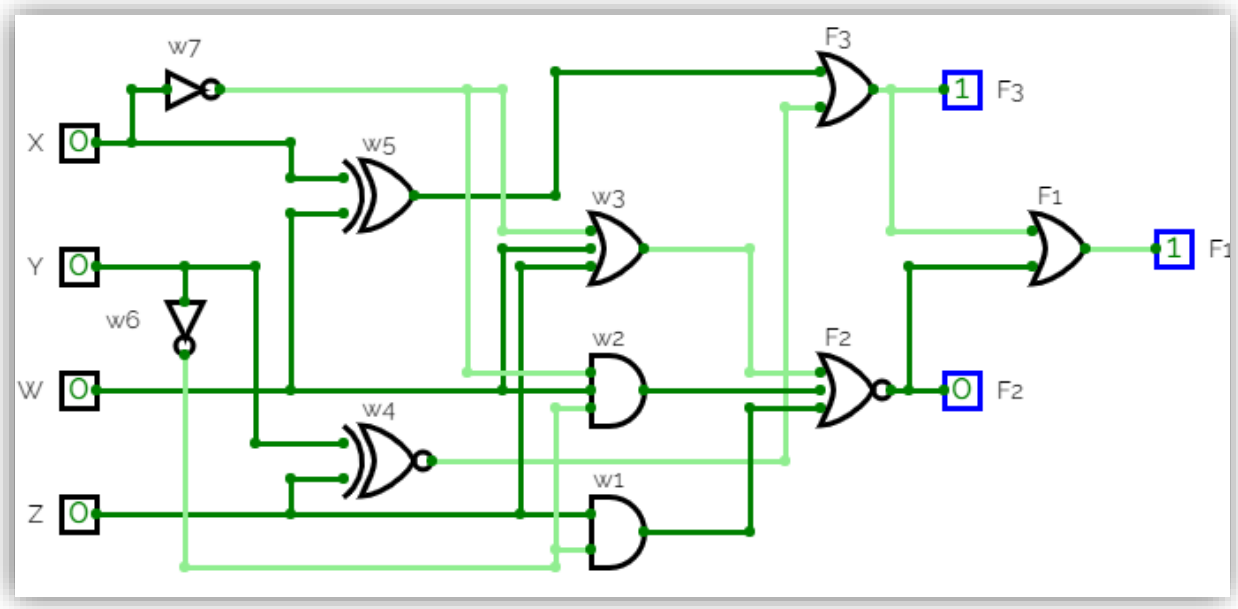


FIGURE 2: ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΓΙΑ ΤΟ Β

---

c)

```
module Circuit_C (x1, x2, x3, A, B);
        output x1, x2, x3;
        input A, B;
        assign x1 = A && B;
        and (x2, A, B);
        assign x3 = A || B;
endmodule
```

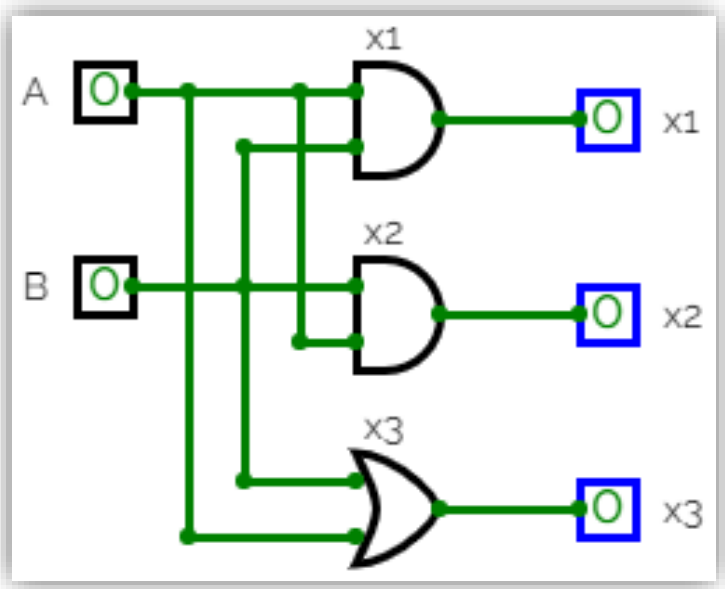
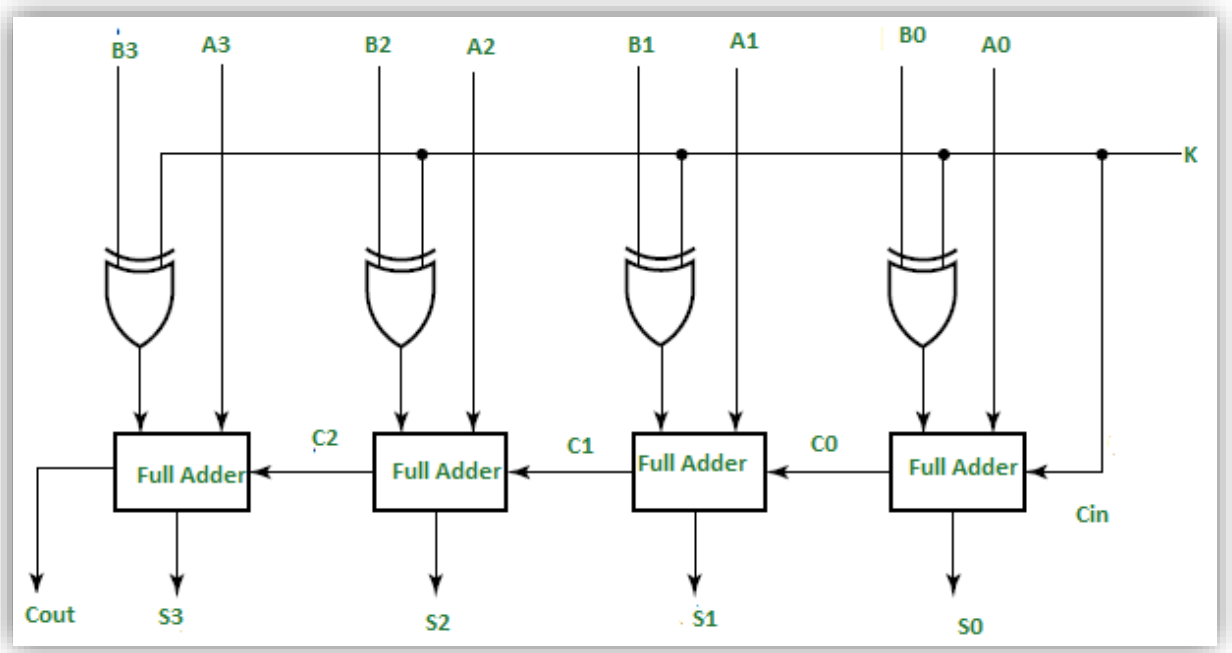


FIGURE 3: ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΓΙΑ ΤΟ C

(ii) Παρακάτω παρουσιάζεται ένας αθροιστής-αφαιρέτης τεσσάρων bit για μη προσημασμένους δυαδικούς αριθμούς:



Περιγραφή HDL σε επίπεδο πύλης:

```
module half_adder (output S, C, input x, y);
                        xor (S, x, y);
                        and (C, x, y);
endmodule

module full_adder (output S, C, input x, y, z);
                        wire S1, C1, C2;
                        half_adder HA1 (S1, C1, x, y),
                                   HA2 (S, C2, S1, z);
                        or G1 (C, C2, C1);
endmodule

module 4bit_add_sub (output [3:0]S, output Cout, input [3:0]A, B, input Cin);
                        wire [3:0]w;
                        wire C0, C1, C2;

                        xor (w[0], B[0], Cin);
                        xor (w[1], B[1], Cin);
                        xor (w[2], B[2], Cin);
                        xor (w[3], B[3], Cin);
                        full_adder(S[0], C0, A[0], w[0], Cin);
                        full_adder(S[1], C1, A[1], w[1], C0);
                        full_adder(S[2], C2, A[2], w[2], C1);
                        full_adder(S[3], Cout, A[3], w[3], C2);

endmodule
```

(iii)

```
module 4bit_add_sub (output [3:0]S, output Cout, input [3:0]A, B, input Cin);

                    assign {Cout, S} = Cin? (A+(~B)+1) : (A+B);

endmodule
```

Άσκηση 7

(i) Παρακάτω φαίνεται η απλοποίηση της εξόδου y:

AB/x	0	1
00	1	0
01	1	0
11	0	1
10	1	0

y = A'x' + ABx + AB'x'

```
module Mealy (y, x, clock, reset);
        output [1:0]y;
        input x, clock, reset;
        reg [1:0] state;
        parameter a=2'b00, b=2'b01, c=2'b10, d=2'b11;
        always @ (posedge clock, negedge reset)
                if (reset == 0) state <= a;
                else case (state)
                        a: if(~x) state <= d; else state <= a;
                        b: if(~x) state <= c; else state <= a;
                        c: if(~x) state <= d; else state <= b;
                        d: if(~x) state <= c; else state <= d;
                endcase
        assign y = (~state[1] & ~x) | (state[1] & state[0] & x) | (state[1] & ~state[0] & ~x);
endmodule
```

(ii) Παρακάτω φαίνεται η απλοποίηση της εξόδου y:

AB/x	0	1
00	0	0
01	1	0
11	1	0
10	1	0

y = Bx' + Ax' = x'(A + B)

```
module Mealy (y, x, clock, reset);
        output [1:0]y;
        input x, clock, reset;
        reg [1:0] state;
        parameter a=2'b00, b=2'b01, c=2'b10, d=2'b11;
        always @ (posedge clock, negedge reset)
                if (reset == 0) state <= a;
                else case (state)
                        a: if(~x) state <= d; else state <= a;
                        b: if(~x) state <= c; else state <= a;
                        c: if(~x) state <= b; else state <= d;
                        d: if(~x) state <= c; else state <= d;
                endcase
        assign y = ~x(state[1] | state[0]);
endmodule
```