Parte I 3

I 2019

P1

a)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
110	100	001	1	0000	0	0	0	1	0	1

op code	DR	SA	SB
1 1 0 0001	110	100	001

1 1 0 001 Comp2 = -15

001 111 → 15

BRN 4, -15

If R4 = 0 then PC ← PC - 15

else PC ← PC + 1

b)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
101	000	111	1	1100	0	1	0	0	0	0

op code	DR	SA	SB
1 0 0 1100	101	000	111

LDI 5, 7

R5 ← 7

c)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
000	111	010	0	0000	0	0	1	0	0	0

opcode	DR	SA	SB
0 1 0 0 0 0 0	000	111	010

ST 7, 2

M[R7] ← R2

d)

DA	AA	BA	MB	FS	MD	RW	MW	PL	JB	BC
110	000	001	0	1101	0	1	0	0	0	1

opcode	DR	SA	SB
0 0 0 1 1 0 1	110	000	001

SHR 6, 1

R6 ← sr(R1)

P2

Un contador de 0 a N necesita N+1 estados. Dado que la máquina es de Moore y que tenemos que señalizar "overflow" del contador hay que considerar estados adicionales.

Si el avance es de 1 o 3 dígitos, el overflow se puede producir con 3 casos

a) $N \rightarrow 0$ (avance 1)
$N-2 \rightarrow 0$ (avance 3)

b) $N-1 \rightarrow 1$

c) $N \rightarrow 2$

Luego hay que tener copias de los estados 0, 1 y 2 para incluir señales de overflow (3 estados)

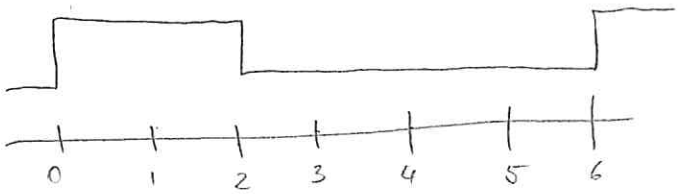
Luego, el total de estados es $(N+1)+3 = \boxed{N+4}$

P3

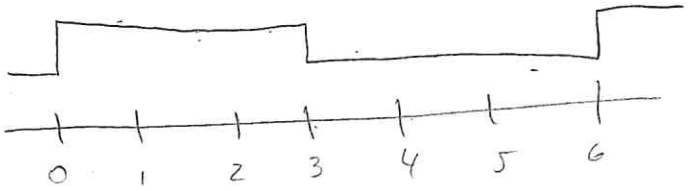
Si uno puede elegir arbitrariamente la frecuencia del clock, conviene el elegir una frecuencia $f/6$ donde el periodo del PWM es $T = \frac{1}{f}$

Con esto se tienen los ciclos de trabajo:

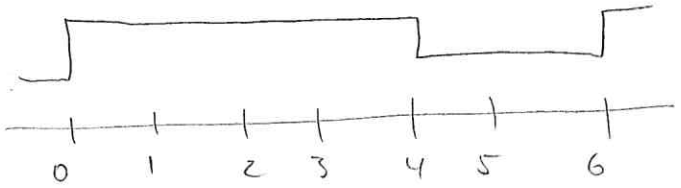
33,3%



50 %



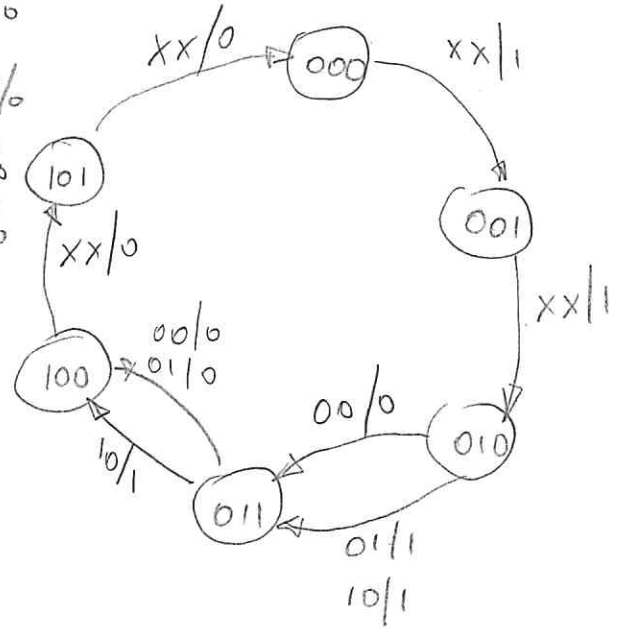
66,6%



Esto genera una máquina de estados

Entradas $P_1 P_0$ Salida Y

00 → 33,3%
01 → 50 %
10 → 66,6%



$Q_2 Q_1 Q_0$	$P_1 P_0$	Y
0 0 0	x x	1
0 0 1	x x	1
0 1 0	0 0	0
0 1 0	0 1	1
0 1 0	1 0	1
0 1 1	0 x	0
0 1 1	1 0	1
1 0 0	x x	0
1 0 1	x x	0

P3

$Q_2 = 0$

Q_1Q_0 \ P_1P_0	00	01	11	10
00	1	1	X	1
01	1	1	X	1
11	0	0	X	1
10	0	1	X	1

$Q_2 = 1$

Q_1Q_0 \ P_1P_0	00	01	11	10
00	0	0	X	0
01	0	0	X	0
11	X	X	X	X
10	X	X	X	X

$$y = \bar{Q}_2 (\bar{Q}_1 + \bar{Q}_0 + P_1)(\bar{Q}_1 + P_1 + P_0)$$

P3

SystemVerilog

```
module PWM(input logic clk,
           input logic reset,
           input logic [1:0] P,
           output logic y);

typedef enum logic {S0, S1, S2, S3, S4, S5} statetype;
statetype state, nextstate;

always-ff @(posedge clk, posedge reset)
  if (reset)  state <= S0
  else        state <= nextstate;

always-comb
  case (state)
    S0: nextstate = S1;
    S1: nextstate = S2;
    S2: nextstate = S3;
    S3: nextstate = S4;
    S4: nextstate = S5;
    S5: nextstate = S0;
    default: nextstate = S0;
  endcase
assign y = ~state[2] & (~state[1] | ~state[0] | P[1]) &
           (~state[1] | P[1] | P[0]);

endmodule
```

P4

```
0  LDI  0,7
1  ADI  0,0,7    %R0 ← 14
2  LD   1,0      %R1 ← N
3  ADI  0,0,5    %R0 ← 19      20 direcciones en RAM
4  LD   2,0      %R2 ← M[D]
5  LDI  3,7      %R3 ← línea 7
6  DEC  1,1
7  INC  0,0      %R0 ← R0 + 1
8  LD   4,0      %R4 ← M[D+1]
9  ADD  2,2,4    %R2 ← R2 + R4
10 DEC  1,1      %R1 ← R1 - 1
11 BRZ  1,2      %R1 ≟ 0
12 JMP  3        %Salto a línea 7
13 LDI  0,4
14 ST   0,2      %M[R0] ← R2
```