



Trabalho Prático 1

Computação Física

(C.F.)

Curso de Licenciatura Informática e Multimédia (LEIM)

Ano Letivo 2017/2018

Turma LEIM 24D

Docentes:

Eng. Jorge Pais

Eng. Carlos Carvalho

Grupo: 1

Membros:

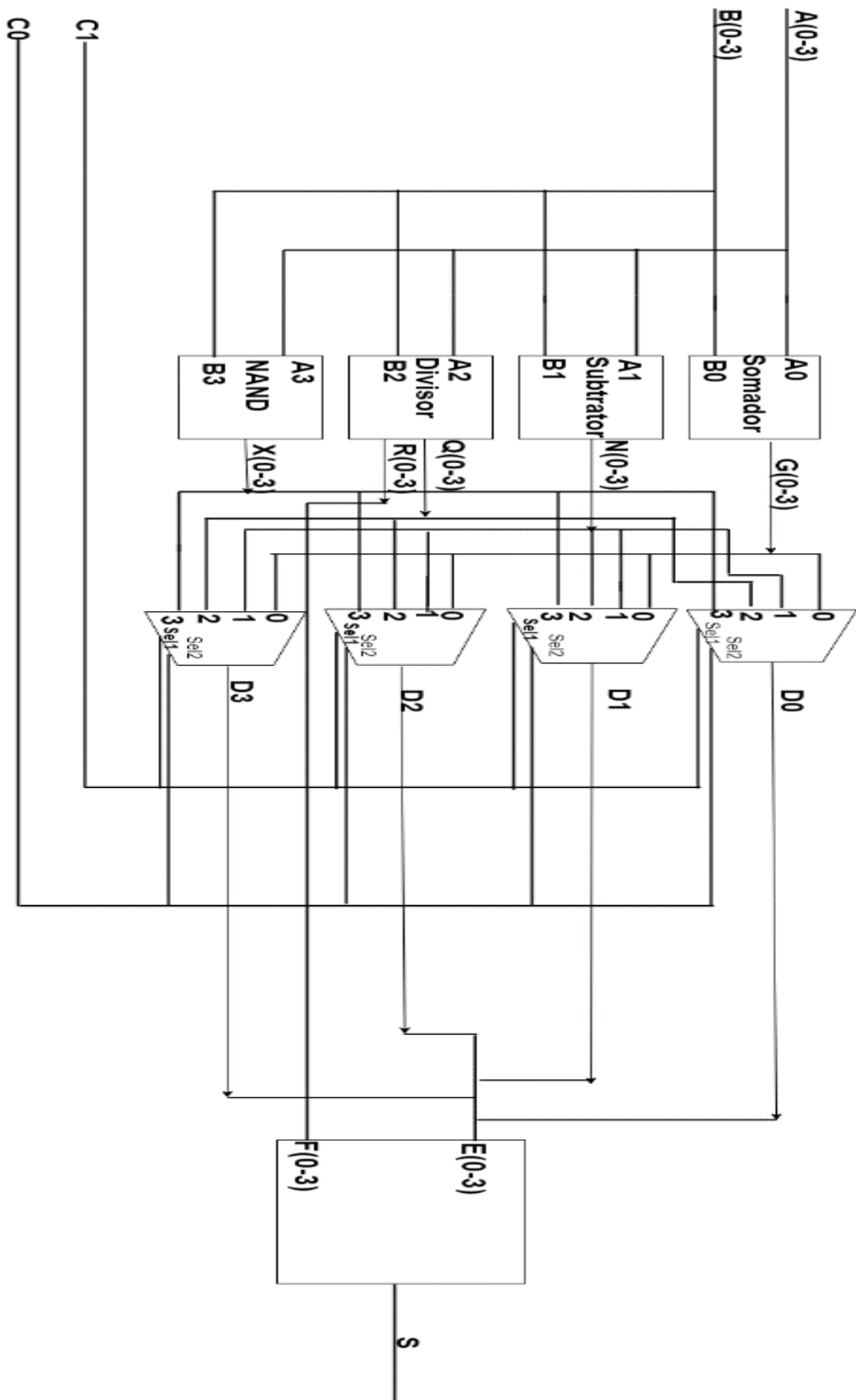
Luis Fonseca (A45125)

Philipp Al-Badavi(A45138)

Introdução

Para este trabalho foi pedido a realização e a representação de circuitos combinatórios e sequencias com as operações/operadores lógicas(os) apresentados no enunciado que são: soma, subtração, divisão e NAND. Neste relatório consta os processos que foram usados, assim como a explicação do procedimento adotado e uma respetiva conclusão.

ALU(arithmetic logic unit)



Somador

Com as entradas fornecidas, coube nos fazer uma tabela de verdade, a partir dai fazer um mapa de karnaugh, retirar as expressões para a implementação no arduino e desenhar o respectivo circuito. Encontra se os processos usados.

Tabela de Verdade:

A	B	S	CyOut
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Mapa de Karnaugh:

S	A
0	1
1	0
B	

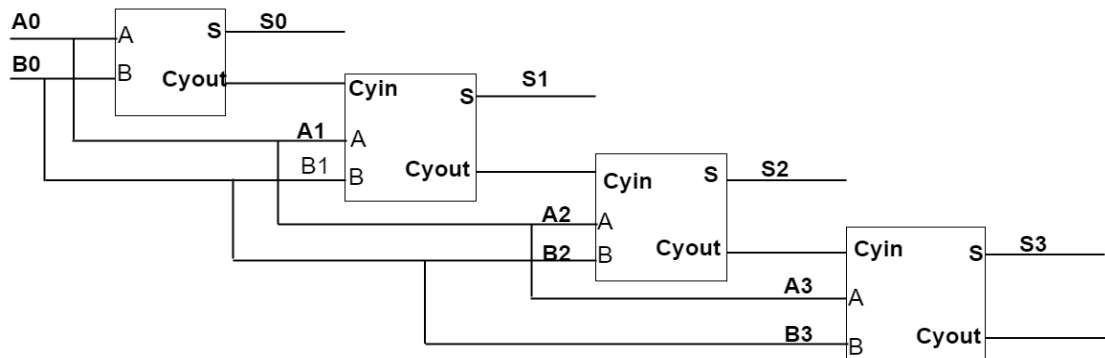
CyOut	A
0	0
0	1
B	

Expressões:

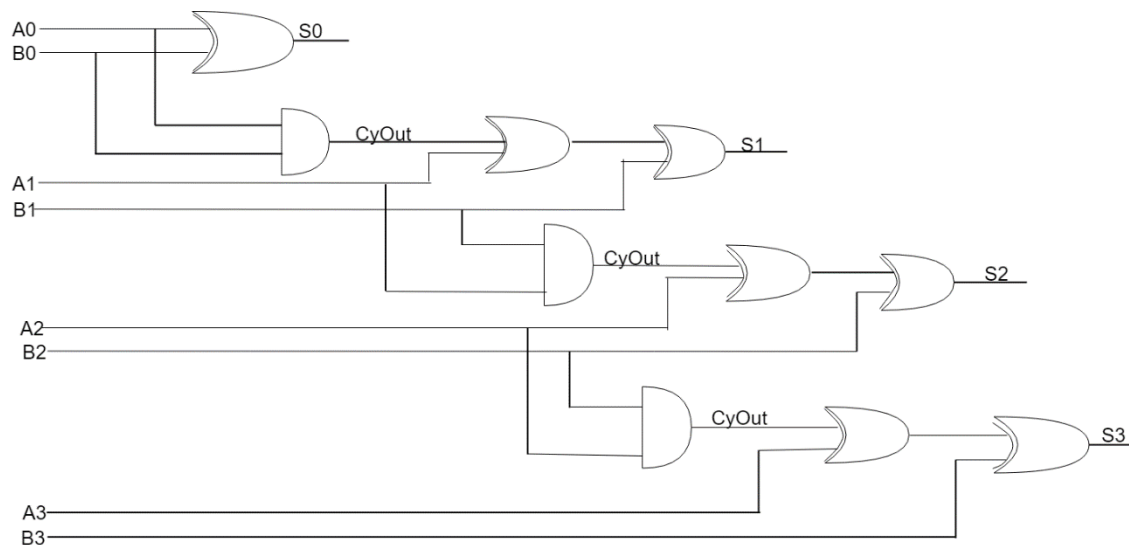
$$S = A \oplus B + A.B$$

$$CyOut = A.B$$

Modulo:



Circuito:



Subtrator

Com as entradas fornecidas, coube nos fazer uma tabela de verdade, a partir dai fazer um mapa de karnaugh, retirar as expressões para a implementação no arduino e desenhar o respetivo circuito. Encontra se os processos usados.

Tabela de Verdade:

A	B	S	Cyout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Mapas de Karnaugh:

Cyout

	A
B	0
	1
	0

S

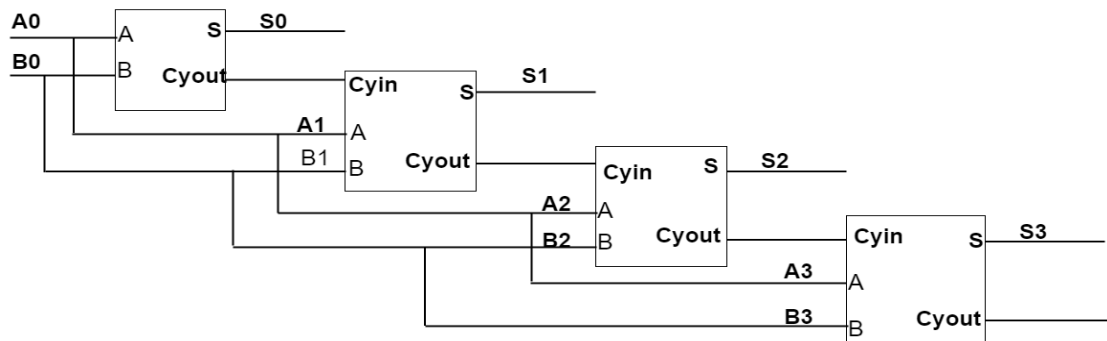
	A
B	1
	0
	1

Expressões:

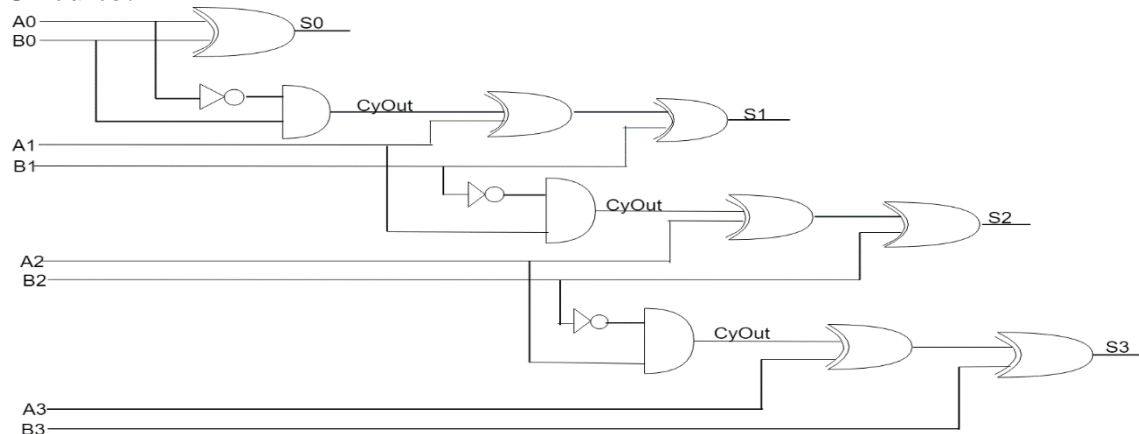
$$S = A \oplus B$$

$$CyOut = A \cdot B$$

Modulo:



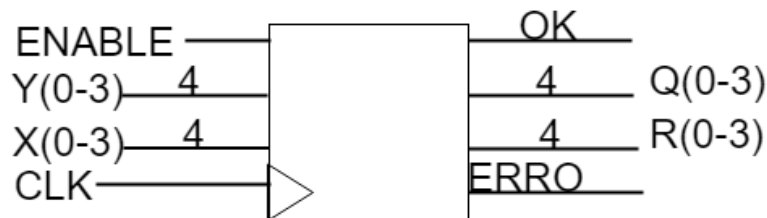
Circuito:



Divisor

Para esta operação logica o processo usado foi diferente dos que foi usado nas outras operações. Neste foram definidos vários passos para chegar ao objetivo pretendido, mas usando tabelas de verdade e mapas de Karnaugh.

1. Definir as entradas e as saídas



2. Algoritmo da divisão

```
1 void setup() {
2     Serial.begin(9600);
3 }
4
5 void loop() {}
6
7 void divisor(int Enable, int Y, int X, int *Pq, int *Pr, int *pOK, int *pErro){
8     int Q, R, OK, ERRO;
9     Pq = &Q; pOK = &OK;
10    Pr = &R; pErro = &ERRO;
11    if(!Enable){
12        Q = 0;
13        R = X;
14    }else{
15        if(Y == 0){
16            ERRO = 0; OK = 1;
17            while(R >= Y){
18                Q++;
19                R = R-Y;
20            }
21            OK = 1;
22        }
23    }
24 }
```

3. Tipos de Hardware

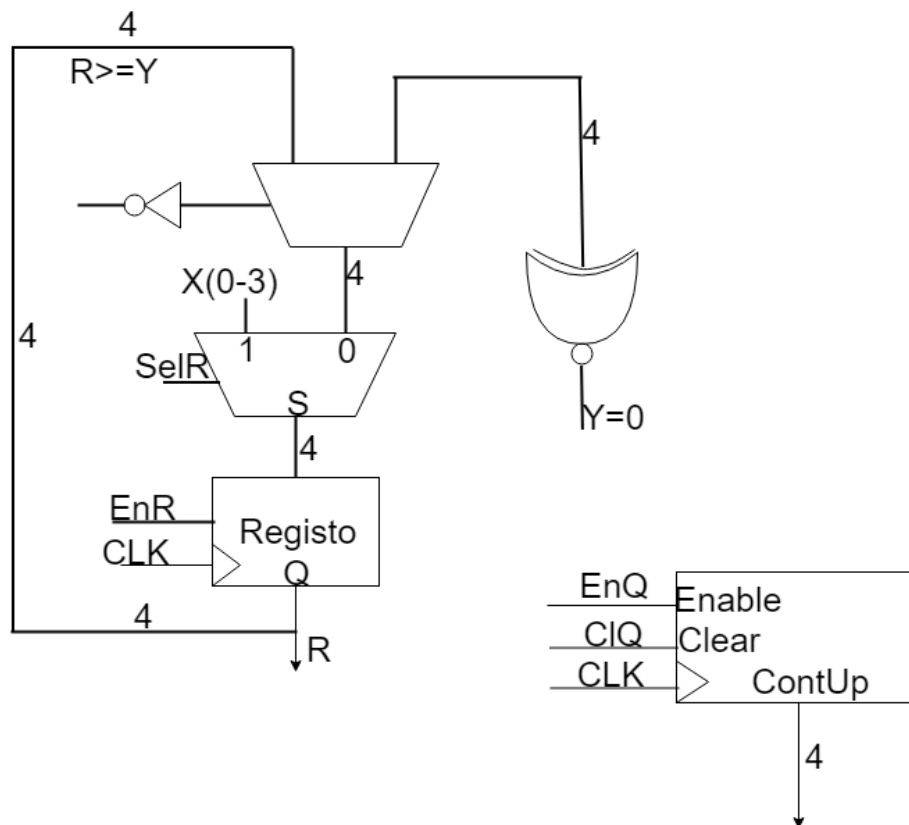
Q – ContadorUp (4bits)

R – Registo (4bits)

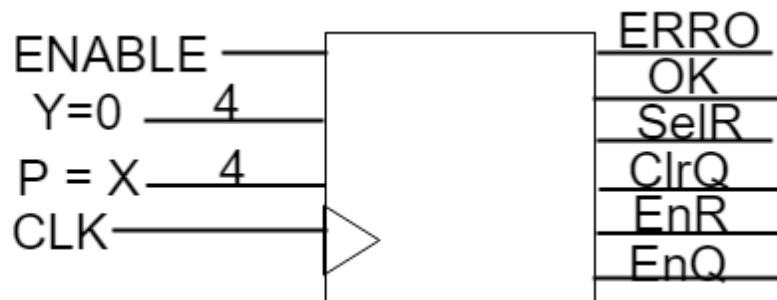
Erro,OK,Enable – Sinais de 1bit

X,Y – Entradas a 4 bits

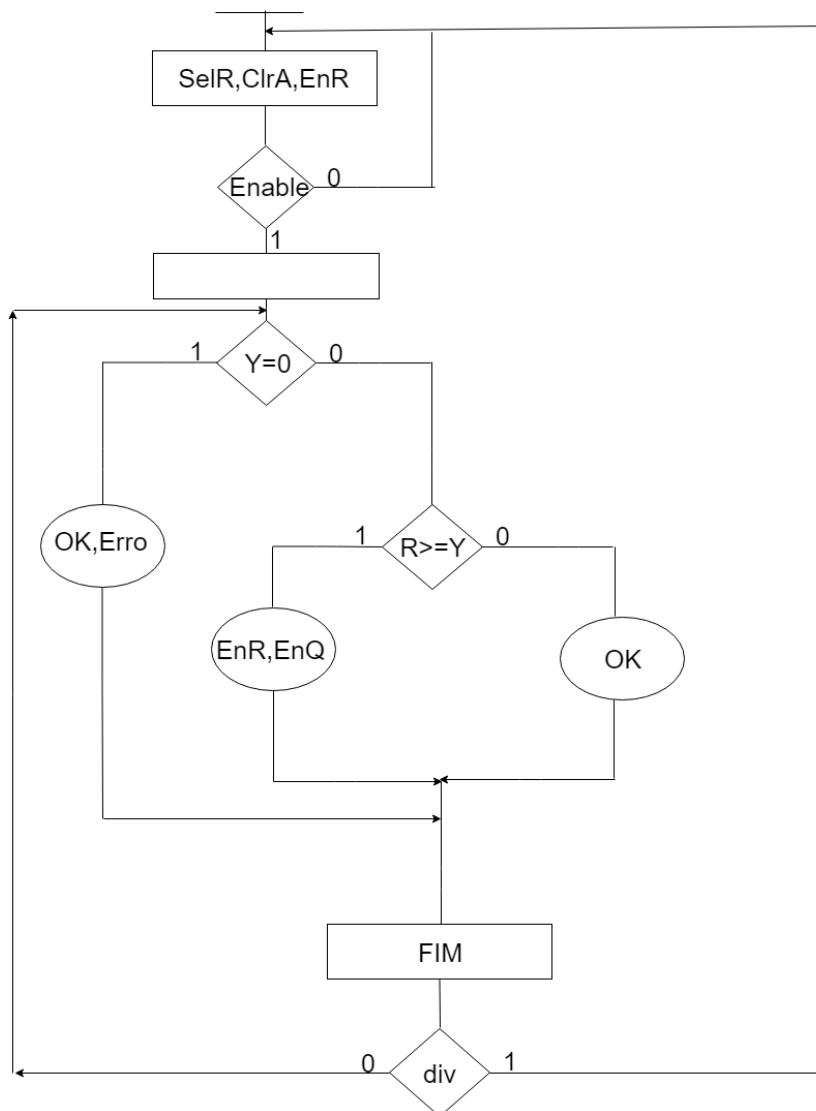
4. Diagrama de blocos do modulo funcional



5. Especificar as entradas e saídas do modulo de controlo



6. ASM – Modulo de Controlo

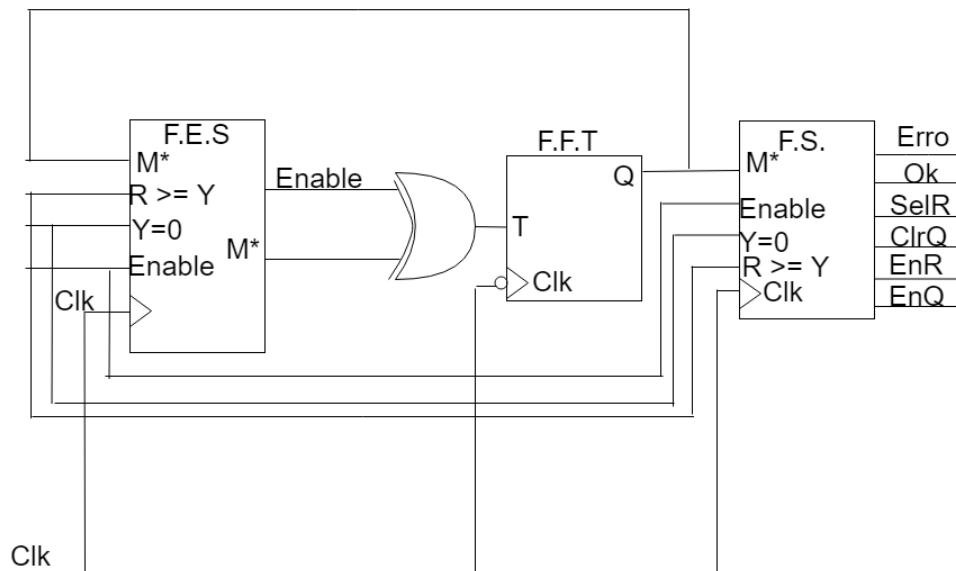


7. Escolher o tipo de célula de Memória

R: Flip-Flop-Edge Triggered T

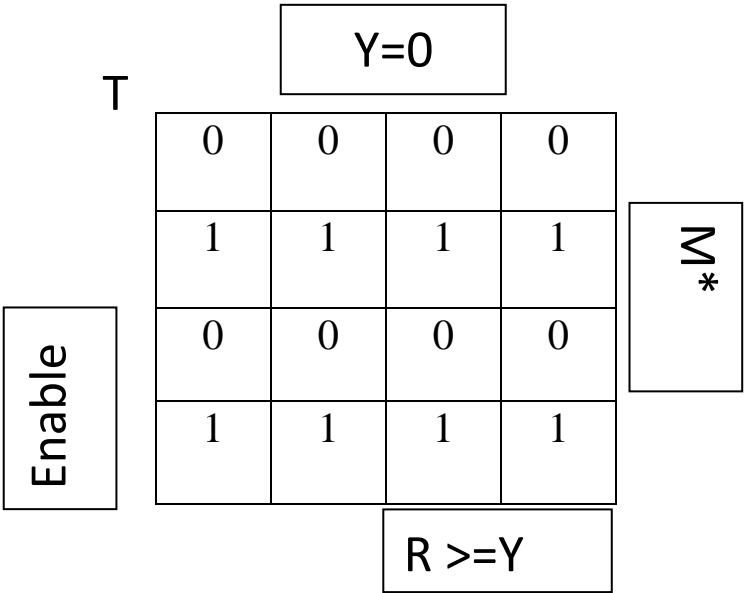
T	Q
0	Q^*
1	$Q^*/$

8. Esquema Mealy-Moore



9.Tabela de Verdade e Mapa de Karnaugh

Enable	Y = 0	R >=Y	M*	M	Q
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	1	0



$T = \text{Enable} \oplus M^*$

NAND

Com as entradas fornecidas, coube nos fazer uma tabela de verdade, a partir daí fazer um mapa de karnaugh, retirar as expressões para a implementação no Arduino e desenhar o respectivo circuito. Encontra-se os processos usados. Esta expressão consiste na negação da operação lógica AND e como é possível observar na tabela de verdade, NAND só toma o valor 1 quando AND for igual a 0.

Tabela de Verdade:

A	B	A.B	(A.B)/
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

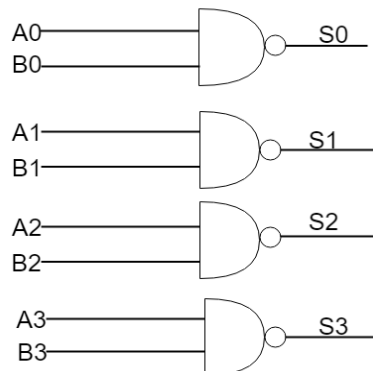
Mapas de Karnaugh:

S	A	
	1	0
B	0	0

Expressões:

$$S = (A.B)/$$

Circuito:



No Arduíno...

Com as expressões retiradas pelos processos mencionados, coube-nos correr no Arduíno para verificar os resultados. Através das varias funções conseguimos obter os resultados que esperávamos. Também foi feito um multiplexe de 4x1(4 entradas, 2 entradas de seleção e 1 saída) caso o utilizador pretenda efetuar outras operações. O código e o respetivo output encontram se em baixo.

- Código Arduíno:

```
24     S= (somadorlbit(X3, Y3, Cyout, pCyout) << 3) | S; //S= S3 | S2 | S1 | S0
25     return S;
26 }
27 //-----SUBTRATOR-----
28
29 int subtratorlbit(int A, int B, int CyIn, int *CyOut)
30 {
31     int S;
32     S = ~A & 0x01;
33     *CyOut = (~A & B) | (CyIn & B) | (~A & CyIn);
34     return A^B^CyIn;
35 }
36 int subtrator4bits(int X3, int X2, int X1, int X0, int Y3, int Y2, int Y1, int Y0)
37 {
38     int CyOut;
39     int *pCyOut;
40     int S;
41     pCyOut= &CyOut;
42     S = (subtratorlbit(X0,Y0,0,pCyOut));
43     S = (subtratorlbit(X1, Y1, CyOut, pCyOut) << 1) | S;
44     S = (subtratorlbit(X2, Y2, CyOut, pCyOut) << 2) | S;
45     S = (subtratorlbit(X3, Y3, CyOut, pCyOut) << 3) | S;
46     //S = CyOut << 4 | S;
47
48 1
49 void setup() {
50     Serial.begin(9600);
51     pinMode(13,OUTPUT);
52 }
53
54 //-----SOMADOR-----
55 int somadorlbit(int A,int B, int CyIn, int *CyOut)
56 {
57     int S;
58     S = A^B^CyIn;
59     *CyOut = A & B | A & CyIn | B & CyIn;
60     return S;
61 }
62 int somador4bits(int X3, int X2, int X1, int X0, int Y3, int Y2, int Y1, int Y0)
63 {
64     int Cyout;
65     int *pCyout;
66     int S;
67     pCyout= &Cyout;
68     S= (somadorlbit(X0, Y0, 0, pCyout)); //S= S0
69     S= (somadorlbit(X1, Y1, Cyout, pCyout) << 1) | S; //S= S1 | S0
70     S= (somadorlbit(X2, Y2, Cyout, pCyout) << 2) | S; //S= S2 | S1 | S0
```

```

47     return S;
48 }
49
50 //-----NAND-----
51
52 int NAND1bit(int A, int B, int CyIn, int *CyOut)
53 {
54     int S;
55     S = ~(A & B) & 0x01;
56     return S;
57 }
58 int NAND4bits(int X3, int X2, int X1, int X0, int Y3, int Y2, int Y1, int Y0){
59     int CyOut;
60     int *pCyOut;
61     int S;
62     pCyOut= &CyOut;
63     S = (NAND1bit(X0,Y0,0,pCyOut));
64     S = (NAND1bit(X1, Y1, CyOut, pCyOut) << 1) | S;
65     S = (NAND1bit(X2, Y2, CyOut, pCyOut) << 2) | S;
66     S = (NAND1bit(X3, Y3, CyOut, pCyOut) << 3) | S;
67     return S;
68 }
95     return S;
96 }
97 //-----MUX-----
98
99 byte MUX_4x1 (boolean C0, boolean C1, int X3, int X2, int X1, int X0, int Y3, int Y2, int Y1, int Y0){
100     int A = somador4bits(X0, X1, X2, X3, Y0, Y1, Y2, Y3);
101     int B = subtrator4bits(X0, X1, X2, X3, Y0, Y1, Y2, Y3);
102     //int C = divisor4bits(X0, X1, X2, X3, Y0, Y1, Y2, Y3);
103     int D = NAND4bits(X0, X1, X2, X3, Y0, Y1, Y2, Y3);
104
105     if (C1 == 0 && C0 == 0){
106         return A;
107     }
108     else if (C1 == 0 && C0 == 1){
109         return B;
110     }
111     else if (C1 == 1 && C0 == 0){
112         // return C;
113     }
114     else if (C1 == 1 && C0 == 1){
115         return D;
116     }
117 }

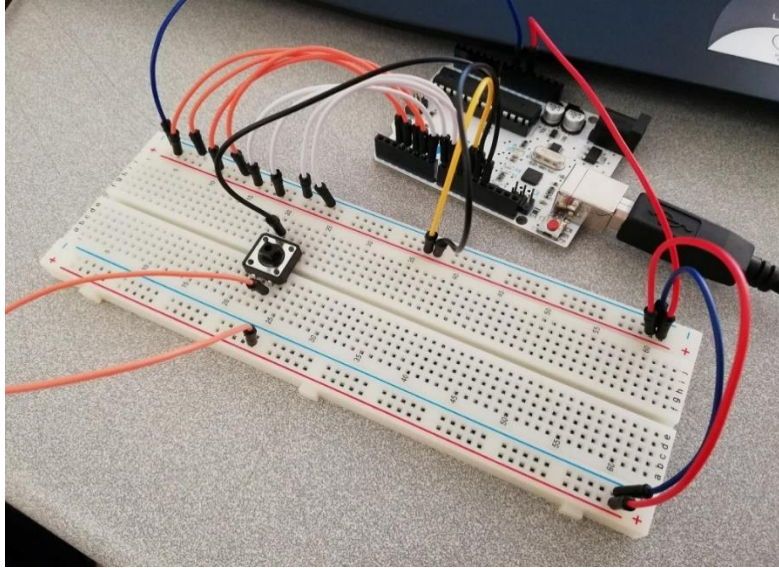
```

```

118
119 void loop() {
120
121     boolean X3=digitalRead(3);//laranja
122     boolean X2=digitalRead(4);//laranja
123     boolean X1=digitalRead(5);//laranja
124     boolean X0=digitalRead(6);//laranja
125     boolean Y3=digitalRead(7)//branco
126     boolean Y2=digitalRead(8)//branco
127     boolean Y1=digitalRead(9)//branco
128     boolean Y0=digitalRead(10)//branco
129     boolean C0=digitalRead(11);//amarelo
130     boolean C1=digitalRead(12)//preto
131     boolean BOTAO=digitalRead(13);
132     if(BOTAO == HIGH){
133         byte f = MUX_4x1(C0, C1, X3, X2, X1, X0, Y3, Y2, Y1, Y0);
134
135         Serial.print("A=");Serial.print(X0);Serial.print(X1);Serial.print(X2);Serial.print(X3);Serial.print("    ");Serial.println("Funções: Somador-> C1=0 e
136         Serial.print("B=");Serial.print(Y0);Serial.print(Y1);Serial.print(Y2);Serial.print(Y3);Serial.print("    ");Serial.println("Divisor-> C1=1 e
137         Serial.print("C1=");Serial.print(C1);Serial.print(" C0=");Serial.println(C0);
138         Serial.println(C1==0 && C0==0 ? "Somador" : C1==0 && C0==1 ? "Subtrator" : C1==1 && C0==0 ? "Divisor" : C1==1 && C0==1 ? "NAND" : "");
139         Serial.println("RESULTADOS:");
140         Serial.print("S(10)="); Serial.println(f);
141         Serial.print("S(16)="); Serial.println(f,16);
142         Serial.print("S(2)="); Serial.println(f,2);
143         Serial.println("-----");
144     }
145 }

```

- Modulo do Utilizador:



- Outputs:

COM11 (Arduino/Genuino Uno)

Send

```

A=0101      Funções: Somador-> C1=0 e C0=0; Subtrator-> C1=0 e C0=1
B=0011      Divisor-> C1=1 e C0=0; NAND-> C1=1 e C0=1
C1=0 C0=0
Somador
RESULTADOS:
S(10)=8
S(16)=8
S(2)=1000

```

Autoscroll No line ending 9600 baud Clear output

COM11 (Arduino/Genuino Uno)

Send

```

A=0101      Funções: Somador-> C1=0 e C0=0; Subtrator-> C1=0 e C0=1
B=0011      Divisor-> C1=1 e C0=0; NAND-> C1=1 e C0=1
C1=0 C0=1
Subtrator
RESULTADOS:
S(10)=2
S(16)=2
S(2)=10

```

Autoscroll No line ending 9600 baud Clear output

COM11 (Arduino/Genuino Uno)

Send

```

A=0101      Funções: Somador-> C1=0 e C0=0; Subtrator-> C1=0 e C0=1
B=0011      Divisor-> C1=1 e C0=0; NAND-> C1=1 e C0=1
C1=1 C0=1
NAND
RESULTADOS:
S(10)=14
S(16)=E
S(2)=1110

```

Autoscroll No line ending 9600 baud Clear output

Conclusão

Com este trabalho foi permitido uma profunda abordagem em relação a este tópico, como se atua e extrair as expressões através de vários processos.