Universidade de Aveiro
Arquitetura de Computadores Avançada

Assignment 1 – Hadamard codes

Gonçalo Aguiar, 98266
Henrique Ramos, 98612

Serial Input - Encoder

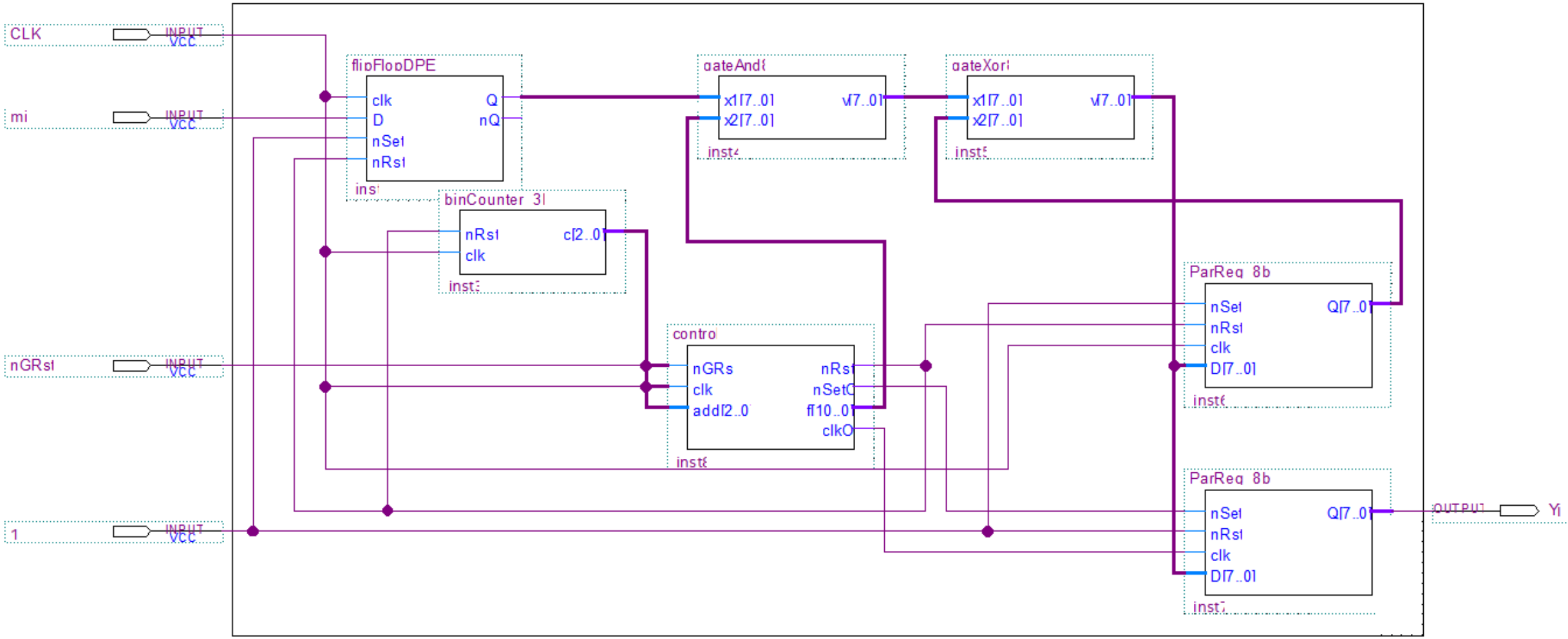
Para codificar o input baseámos o nosso código VHDL no seguinte algoritmo visto que os k's são sempre constantes para cada linha do processo de codificação :

```
for(int i = 0;i<4;i++)
    for(int j = 0;j<8;j++)
        y[i] = y[i] xor (k[j][i] and m[i]);
```

De modo a implementar este algoritmo em VHDL criámos o circuito do slide seguinte composto por:

- **Unidade de Controlo**: Coordena o circuito e guarda os k's para cada iteração.
- **BinCounter_3bit**: Permite iterar pelas instruções da Unidade de Controlo.
- **Gates And e Xor:** Fazem as operações do algoritmo.
- **Register_8bits:** Guarda o estado de cada y(y0-y7).

Serial Input - Encoder



Implementation Cost = 20 FlipFlops D-type + 9 and + 10 xor + 4 nand + 1 nor

Parallel Input - Decoder

Expressões para calcular os valores dos m(deduzidas a partir do Mapa de Karnaugh):

- **m#(1)** = c3c2(c1+c0) + c1c0(c3+c2)
- **m#(0)** = ~c3~c2(~c1+~c0) + ~c1~c0(~c3+~c2)
- **m#(E)** = ~(m#(1) + m#(0))
- **valid** = ~(m0(E) + m1(E) + m2(E))
- **m0** = m0(1)valid
- **m1** = m1(1)valid
- **m2** = m2(1)valid

= {0,1,2}

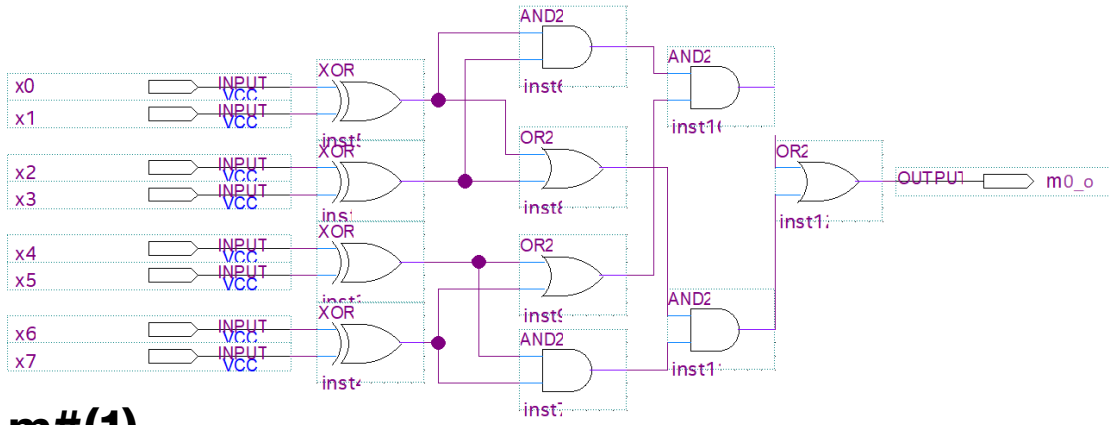
~ = not

Para calcular o **m3** utilizámos um circuito composto por:

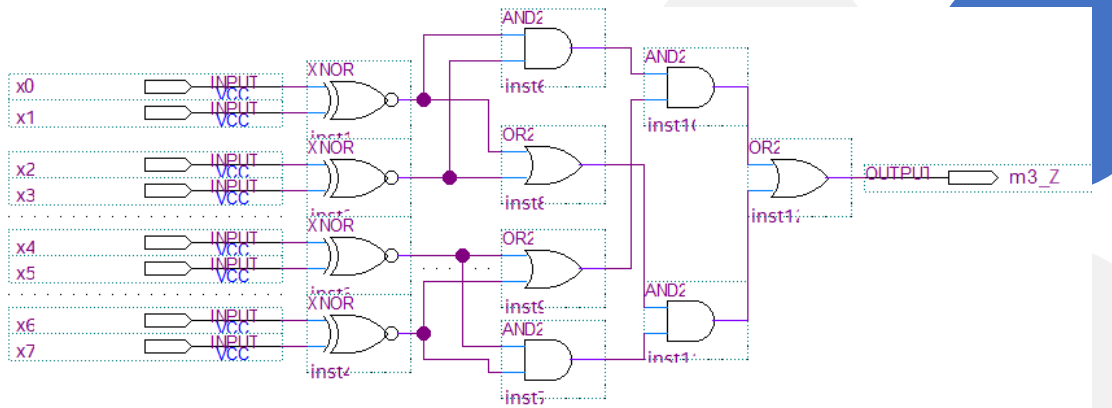
- **Partial Encoder**: Codifica em paralelo a supor que m3= 0
- **1º Xor** : Compara os valores que saem do Partial Encoder com os valores reais.
- **PopCounter e 2º Xor:** Conta o número de valores diferentes e confirma o valor de m3 , alterando-o se estiver errado.

$c_{\#3}c_{\#2}$ \ $c_{\#1}c_{\#0}$	00	01	11	10
00	0	0	E	0
01	0	E	1	E
11	E	1	1	1
10	0	E	1	E

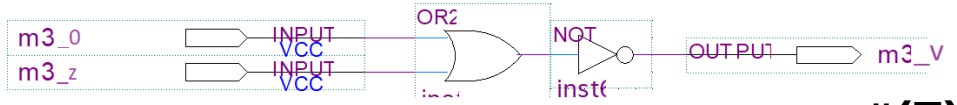
Parallel Input - Decoder



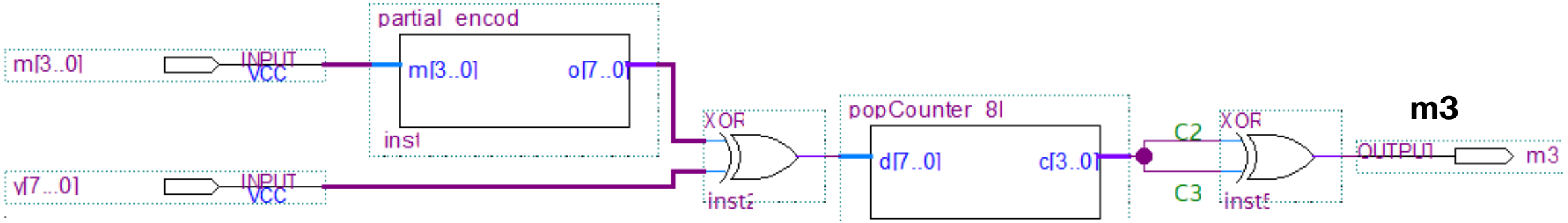
m#(1)



m#(0)



m#(E)



m3

Implementation Cost = 43 xor + 40 and + 24 or

Worst Case Propagation Delay = 10 xor + 5 and + 5 or