Hamming Codes

Digital Circuit Implementation

Turma 1 Grupo 6

Tiago Marques , 98459

Bruno Lemos , 98221

Assignment 1 – Arquiteturas de Alto Desempenho

Professor António Rui Borges

Novembro 2022

universidade de aveiro

Encoder – Parallel Implementation

Through the expressions given by the professor, we tried to find common operations that could be simplified in order to reduce the number of components. After an analysis we found 2 possible implementations.

$$
\begin{aligned}
x_0 &= m_4 \\
x_1 &= m_1 \oplus m_4 \\
x_2 &= m_2 \oplus m_4 \\
\mathbf{Y} &= m_1 \oplus m_2 \\
x_3 &= \mathbf{Y} \oplus m_4 \\
x_4 &= m_3 \oplus m_4 \\
x_5 &= m_1 \oplus x_4 \\
x_6 &= m_2 \oplus x_4 \\
x_7 &= \mathbf{Y} \oplus x_4
\end{aligned}
$$

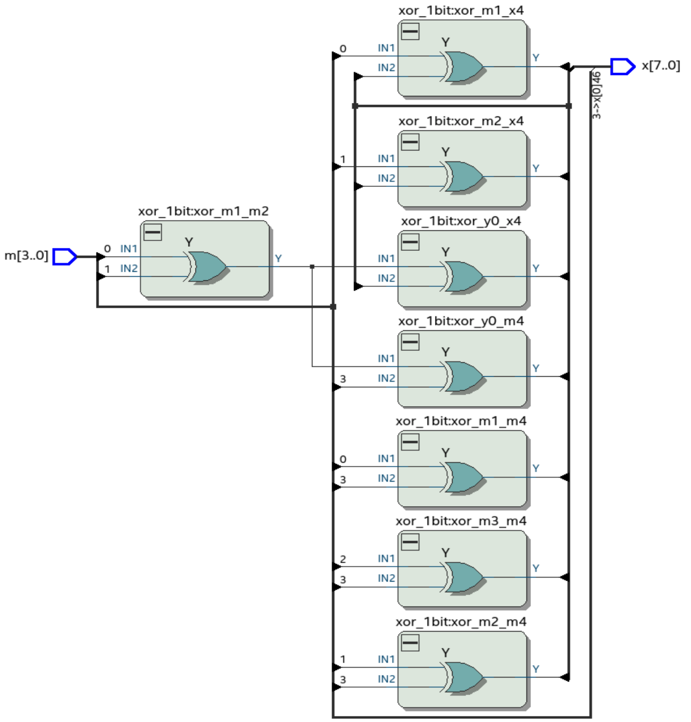
8 XORS

2 XOR delay

$$
\begin{aligned}
x_0 &= m_4 \\
x_1 &= m_1 \oplus m_4 \\
x_2 &= m_2 \oplus m_4 \\
x_3 &= m_1 \oplus x_2 \\
x_4 &= m_3 \oplus m_4 \\
x_5 &= m_1 \oplus x_4 \\
x_6 &= m_2 \oplus x_4 \\
x_7 &= m_1 \oplus x_6
\end{aligned}
$$

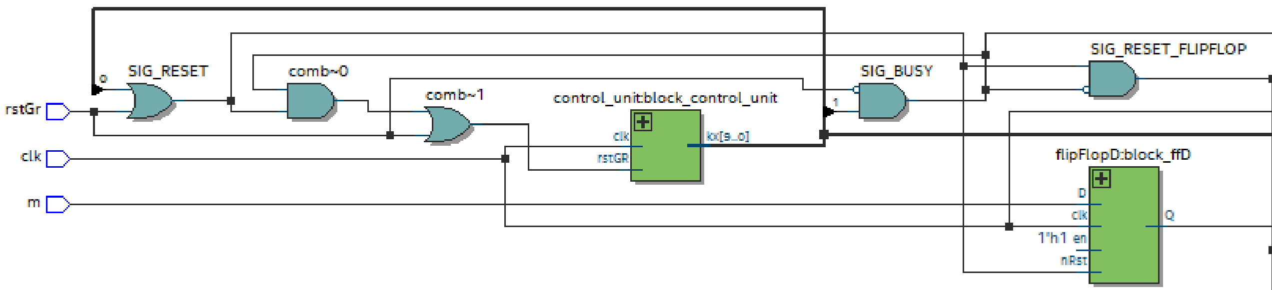
7 XORS

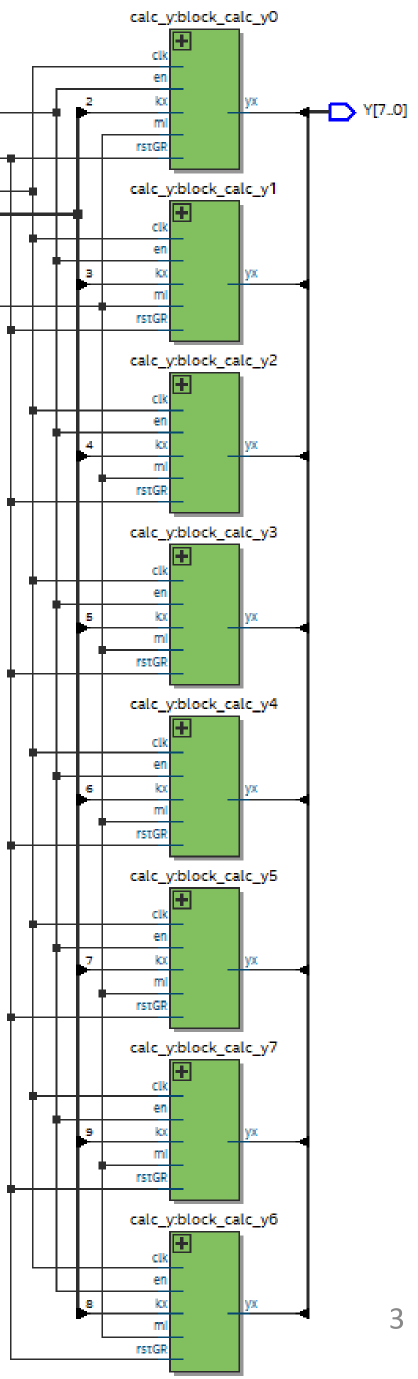
3 XOR delay



We chose the implementation with 8 XOR because despite having 1 more XOR it is faster, and we think that the benefit is worth the cost.

Encoder – Serial Implementation





Consists of 2 main blocks:

- **Control unit :** contains a counter and a ROM, sends control signals.
- **Calc y :** Calculate Y based in one expression and show it in final

Through the expression below we can extract the Y value assigned through the respective K values:

$$Y(x) = kX0 \cdot M0 \oplus kX1 \cdot M1 \oplus kX2 \cdot M2 \oplus kX3 \cdot M3$$

Implementation Cost:

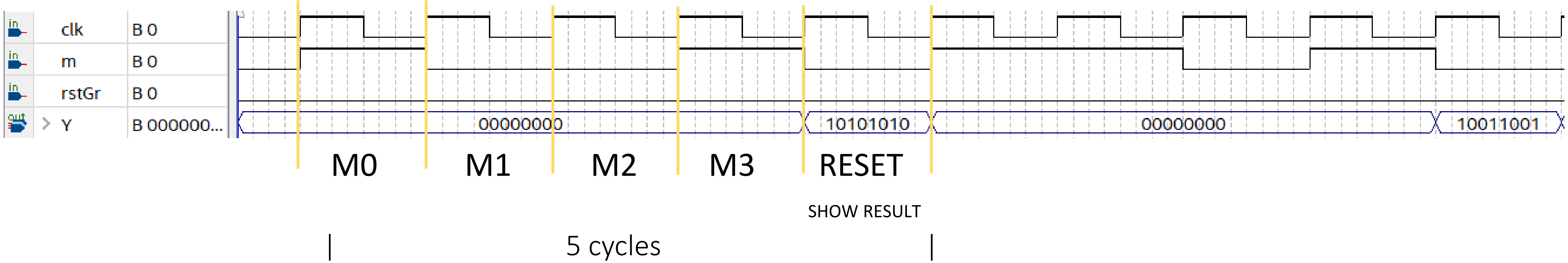
1 ROM + 1 COUNTER + 8 CalcY + 2 NOT + 3 AND + 2OR + 1 FLIPFLOP

For each : CalcY : 2 ANDS + 1 XOR + 1 OR + 2 NOT + 1 FLIPFLOP

Execution Time:

5 clocks [reset/show M0 M1 M2 M3]

Encoder – Serial Implementation



STATE \ SIGNAL	K [0..7]	busy	reset
RESET	Don’t care	0	1
M0	01010101	1	0
M1	00110011	1	0
M2	00001111	1	0
M3	11111111	1	1

Busy = 1 - Calculate Y
Busy = 1 and Reset = 1 - Reset Counter and Calculate Y
Busy = 0 and Reset = 1 - Show the result and Reset FlipFlop in block CalcY

Decoder

Consists of 3 blocks:

- **Y_to_C:** Made with 12 XORS that calculate codewords(C#1, C#2, C#3, C#4).
- **validator:** Calculates the value of m# and checks if there is an error or not. We use this block 3 times, for m1, m2 and m3.
- **calc_m4:** Calculates the value of m4.

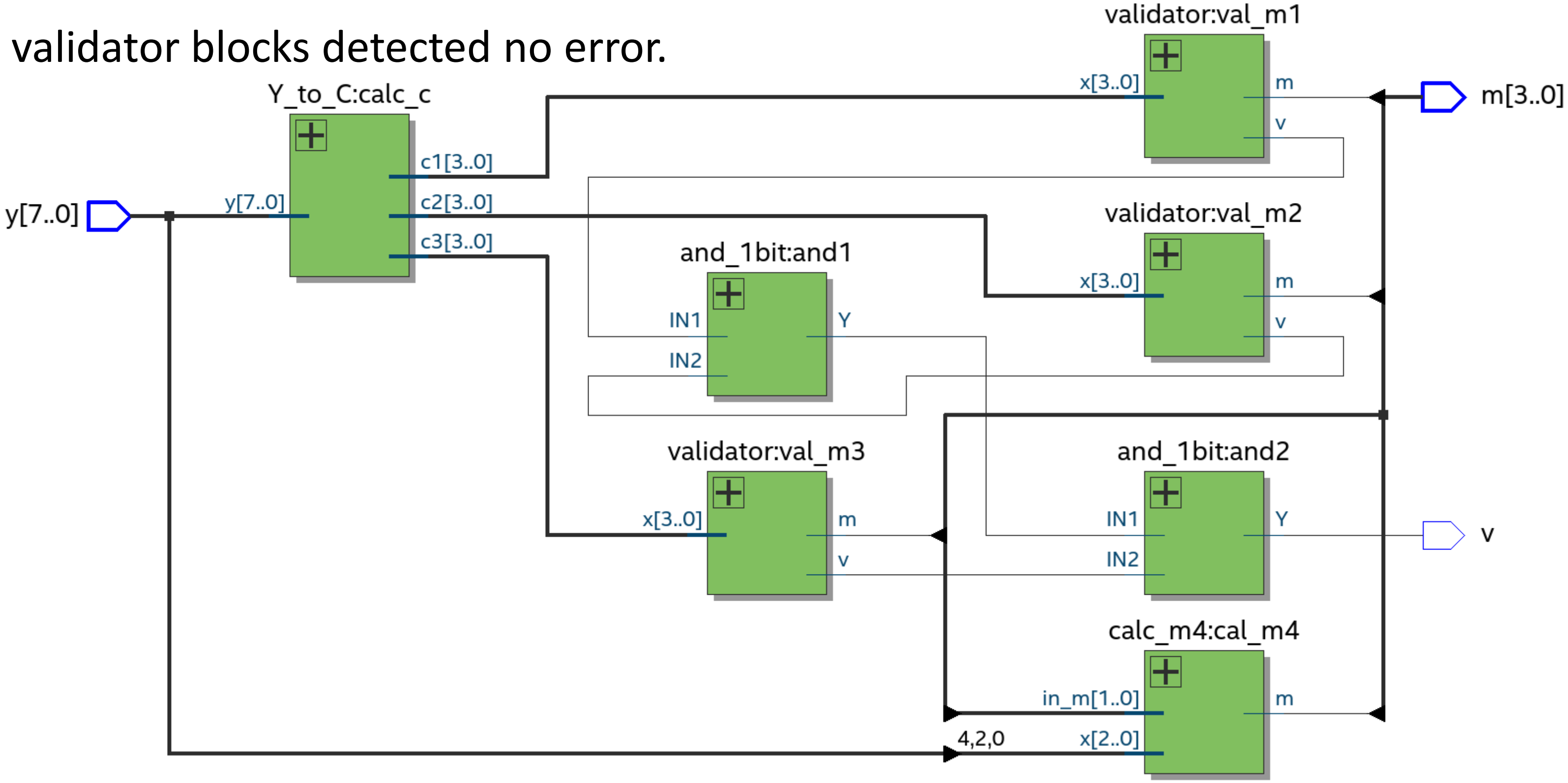
Note: 2 ANDS are used to check if the 3 validator blocks detected no error.

Implementation Cost:

17 XOR + 10 AND + 14 OR + 6 NOT

Propagation delay:

3 XOR + 3AND + 3 OR



Decoder: validator

A truth table was built, with the help of the Karnaugh map related to the application of the property of local decodability. With an analysis of the truth table, we can see that m# has the value of eq1 and that v (validity bit) has the value of eq0 XOR eq1.

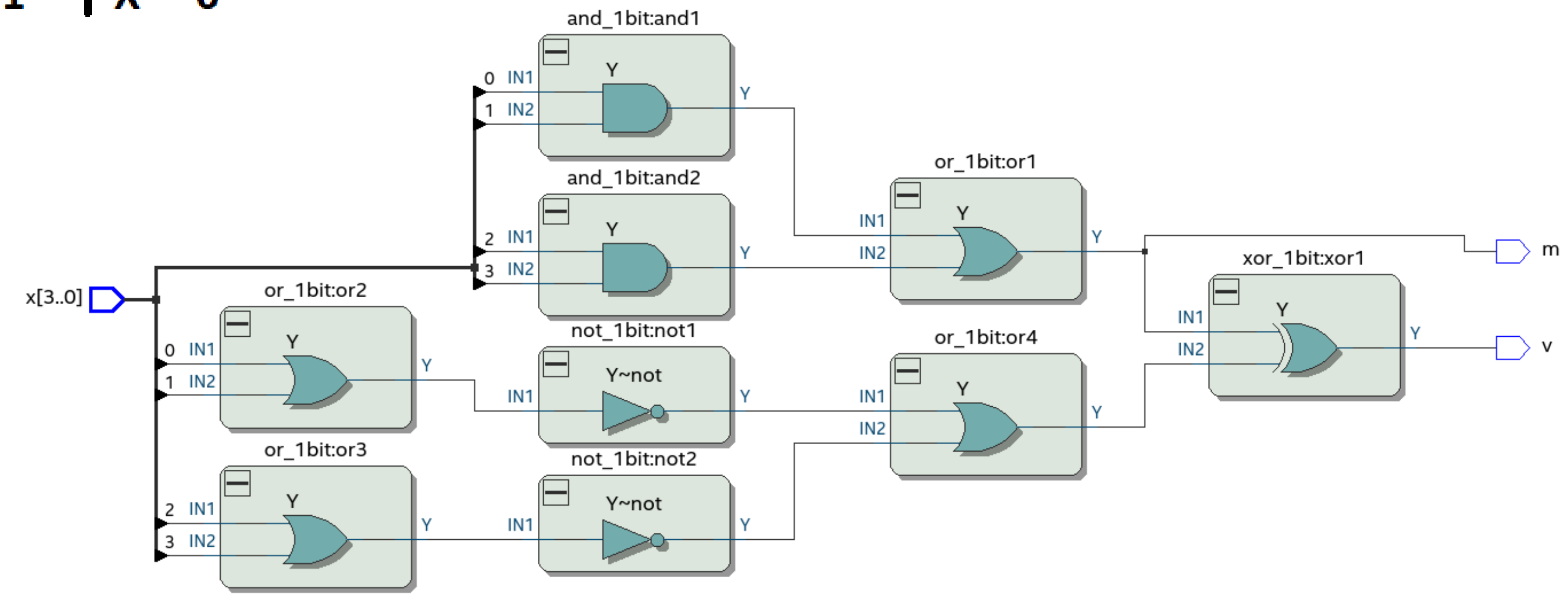
	00	01	11	10
00	0	0	E	0
01	0	E	1	E
11	E	1	1	1
10	0	E	1	E

eq0 eq1

eq0	eq1	m	v
0	0	X	0
0	1	1	1
1	0	0	1
1	1	X	0

$$eq1 = (c0.c1) + (c2.c3)$$

$$eq0 = (\overline{c0}.\overline{c1}) + (\overline{c2}.\overline{c3}) \Leftrightarrow \overline{(c0+c1)} + \overline{(c2+c3)}$$



Decoder: calc_m4

Calculating the real value of m4 is tricky because it enters in the computation of all the codeword bits. With that in mind, we can use the calculated values of m# (# ϵ [1,3]) and the received code values (y1 … y7) to reverse the processe and calculate m4. We use the following 3 equations:

$m_4 = x_0$ ⇨ eq0

$m_4 = m_2 \oplus x_2$ ⇨ eq2

$m_4 = m_3 \oplus x_4$ ⇨ eq4

eq4 \ eq2, eq0	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$(eq0.eq2) + (eq0.eq4) + (eq2.eq4) \equiv (eq0.eq2) + ((eq0 + eq2).eq4)$$

We know, in fact, that m2 and m3 have their correct values, otherwise their corresponding validity bit is 0 and the decoded message is considered invalid. What we don't know is whether any of the x0, x2 or x4 have been distorted or not and that's why we use 3 equations.

If 2 out of the 3 x# we use in these 3 equations have suffered distortion, then during the calculation of the values of m# at least one of their validity bit is 0, which invalidates the decoded message.

If 1 out of the 3 x# we use in these 3 equations have suffered distortion, we get the value of 1 of these equations wrong, but the values of the other 2 equations are right. After computing the results of these 3 equations, the value of m4 is given by the value which took place the most.