universidade de aveiro
Departamento de Eletrónica, Telecomunicações e Informática

Computação Reconfigurável

Trabalho Prático

2020/2021

P2

Alunos:

Rodrigo Lopes da Silva Santos, 89180
Alexey Kononov Vladimirovich, 89227

CRC16 Encoder

Descrição:

Neste projeto, foi desenvolvido um co-processador especializado para calcular o CRC (Cyclic Redundancy Check) de 16 bits dada uma palavra de 32 bits.
A palavra é gerada aleatoriamente e enviada para o coprocessador que aplica o método de cálculo de CRC baseado nas propriedades do resto da divisão. Na qual são identificados os bits da palavra original que têm interferência no resultado de cada bit do CRC.

Explicação:

Após uma pesquisa, sobre diversos polinómios geradores de CRC, foi escolhido o polinómio x^16 + x^15 + x^2 + 1. Após calculados os bits com influência no resultado final de cada bit do CRC chegamos a esta tabela de influência.

Word Bit / CRC bit	r^15	r^14	r^13	r^12	r^11	r^10	r^9	r^8	r^7	r^6	r^5	r^4	r^3	r^2	r^1	r^0
a0	1													1		1
a1	1												1	1	1	1
a2	1											1	1		1	1
a3	1										1	1			1	1
a4	1									1	1				1	1
a5	1								1	1					1	1
a6	1							1	1						1	1
a7	1						1	1							1	1
a8	1					1	1								1	1
a9	1				1	1									1	1
a10	1			1	1										1	1
a11	1		1	1											1	1
a12	1	1	1												1	1
a13		1													1	1
a14	1													1	1	
a15	1												1			1
a16	1											1		1	1	1
a17	1										1		1		1	1
a18	1									1		1			1	1
a19	1								1		1				1	1

a20	1							1		1					1	1
a21	1						1		1						1	1
a22	1					1		1							1	1
a23	1				1		1								1	1
a24	1			1		1									1	1
a25	1		1		1										1	1
a26	1	1		1											1	1
a27			1												1	1
a28		1												1	1	
a29	1													1		
a30	1											1	1	1		1
a31	1										1	1	1	1	1	1

A partir da tabela obtida para implementar em VHDL, apenas temos de fazer xor entre todos os bits sinalizados a 1 da palavra inicial para coluna a coluna para obter o CRC resultante.

```
crc(0) <= dataIn(31) xor dataIn(30) xor dataIn(27) xor dataIn(26) xor dataIn(25) xor
dataIn(24) xor dataIn(23) xor dataIn(22) xor dataIn(21) xor dataIn(20) xor dataIn(19) xor
dataIn(18) xor dataIn(17) xor dataIn(16) xor dataIn(15) xor dataIn(13) xor dataIn(12) xor
dataIn(11) xor dataIn(10) xor dataIn(9) xor dataIn(8) xor dataIn(7) xor dataIn(6) xor dataIn(5)
xor dataIn(4) xor dataIn(3) xor dataIn(2) xor dataIn(1) xor dataIn(0);

crc(1) <= dataIn(31) xor dataIn(28) xor dataIn(27) xor or dataIn(26) xor dataIn(25) xor
dataIn(24) xor dataIn(23) xor dataIn(22) xor dataIn(21) xor dataIn(20) xor dataIn(19) xor
dataIn(18) xor dataIn(17) xor dataIn(16) xor dataIn(14) xor dataIn(13) xor dataIn(12) xor
dataIn(11) xor dataIn(10) xor dataIn(9) xor dataIn(8) xor dataIn(7) xor dataIn(6) xor dataIn(5)
xor dataIn(4) xor dataIn(3) xor dataIn(2) xor dataIn(1);

crc(2) <= dataIn(31) xor dataIn(30) xordataIn(29) xor dataIn(28) xor dataIn(16) xor
dataIn(14) xor dataIn(1) xor dataIn(0);

crc(3) <= dataIn(31) xor dataIn(30) xor dataIn(29) xor dataIn(17) xor dataIn(15) xor dataIn(2)
xor dataIn(1);

crc(4) <= dataIn(31) xor dataIn(30) xor dataIn(18) xor dataIn(16) xor dataIn(3) xor dataIn(2) ;

crc(5) <= dataIn(31) xor dataIn(19) xor dataIn(17) xor dataIn(4) xor dataIn(3) ;

crc(6) <= dataIn(20) xor dataIn(18) xor dataIn(5) xor dataIn(4) ;

crc(7) <= dataIn(21) xor or dataIn(19) xor dataIn(6) xor or dataIn(5) ;

crc(8) <= dataIn(22) xor dataIn(20) xor dataIn(7) xor dataIn(6) ;
crc(9) <= dataIn(23) xor dataIn(21) xor dataIn(8) xor dataIn(7) ;
```

```
crc(10)<= dataIn(24) xor dataIn(22) xor dataIn(9) xor dataIn(8) ;

crc(11) <= dataIn(25) xor dataIn(23) xor dataIn(10) xor dataIn(9) ;

crc(12) <= dataIn(26) xor dataIn(24) xor dataIn(11) xor dataIn(10) ;

crc(13) <= dataIn(27) xor dataIn(25) xor dataIn(12) xor dataIn(11) ;

crc(14) <= dataIn(28) xor dataIn(26) xor dataIn(13) xor dataIn(12) ;

crc(15) <= dataIn(31) xor or dataIn(30) xor dataIn(29) xor dataIn(26) xor dataIn(25) xor
dataIn(24) xor dataIn(23) xor dataIn(22) xor dataIn(21) xor dataIn(20) xor dataIn(19) xor
dataIn(18) xor dataIn(17) xor dataIn(16) xor dataIn(15) xor dataIn(14) xor dataIn(12) xor
dataIn(11) xor dataIn(10) xor dataIn(9) xor dataIn(8) xor or dataIn(7) xor dataIn(6) xor
dataIn(5) xor or dataIn(4) xor dataIn(3) xor dataIn(2) xor dataIn(1) xor dataIn(0);
```

Esta implementação poderia ser melhorada caso fossem efetuados xors que se repetem múltiplas vezes uma única vez, armazenando o resultado dessa operação para uso nas outras operações idênticas.

Diagrama de Blocos do coprocessador:

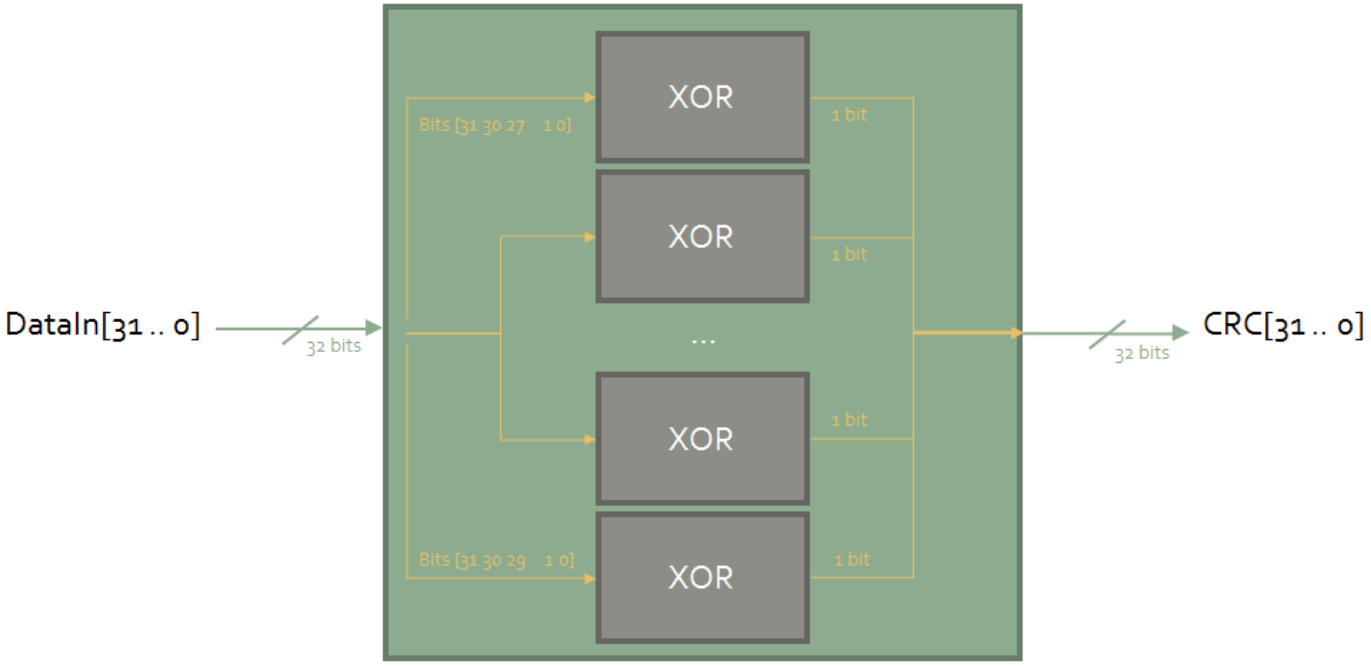

Figura 1: Diagrama de Blocos

Nota: Saída de 32 bits, no entanto o CRC encontra-se nos 16 bits menos significativos da saída, sendo os restantes 16 bits 0's concatenados.

Resultados:

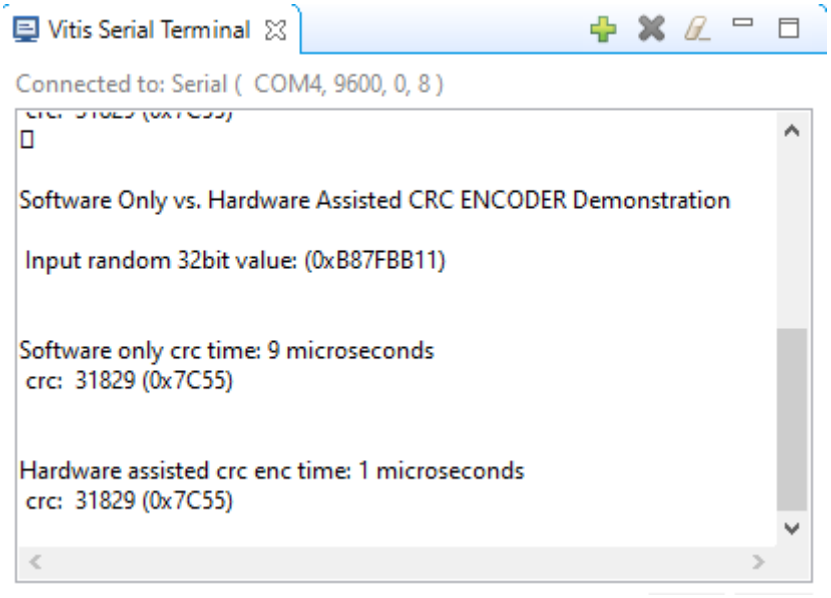

Figura 2: ResultadosTemporais

Como é possível verificar pela imagem anterior, o tempo de cálculo do CRC16 para uma palavra de 32 bits foi cerca de 9 vezes mais rápido utilizando Hardware especializado, do que utilizando apenas Software.

Contribuições e Auto-avaliação:

Para o desenvolvimento do projeto ambos os membros contribuirão de igual forma, numa divisão de 50% para cada um, no trabalho desenvolvido.

Auto-avaliação do grupo em relação ao projeto: 15 valores.