Universidade Federal de Santa Catarina
EEL7123/EEL510269
Semestre: 2019/2 – Lab1a

Conversor Binario-RNS

1 Introdução e objetivos

O objectivo deste laboratório consiste em projetar em FPGA uma unidade conversora de binário a numeração residual (RNS) vistas nas aulas teóricas. Estas unidades serão reutilizadas nas seguintes aulas experimentais para o desenvolvimento de unidades RNS completas com funcionalidade aritmética soma e multiplicação. A Figura 1 descreve os três níveis de operação das unidades RNS usando o conjunto de módulos $\{m_1, m_2, m_3\} = \{2^{2n}, 2^n - 1, 2^n + 1\}$: i) Conversores binário a RNS (Binary-to-RNS converters) que veremos em este aula 1a, ii) unidades aritméticas RNS (RNS arithmethic units) que serão vistas nas aulas 2a, 2b e 3a e iii) conversor RNS a binário (RNS-to-Binary converters) que será visto na aula 1b.

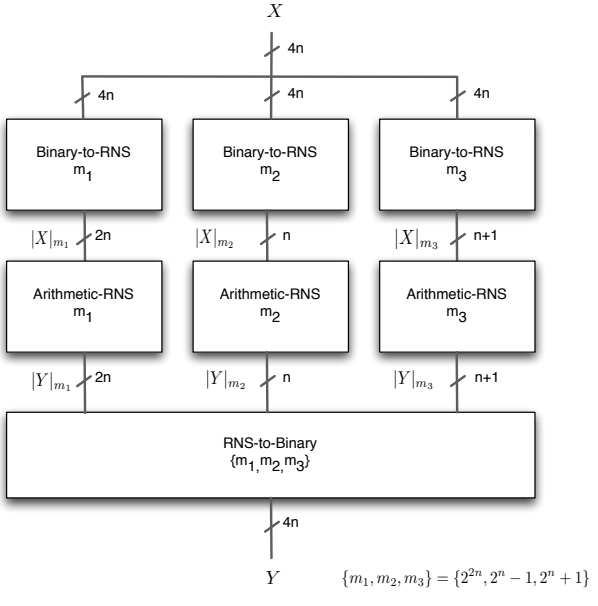

Figura 1: Unidade RNS completa usando conjunto de módulos $\{m_1, m_2, m_3\} = \{2^n, 2^n - 1, 2^n + 1\}$.

2 Conversor Binário-RNS

Um numero inteiro $X = \{x_{(4n-1)}, \ldots, x_1, x_0\}$ pode ser expressado em notação binaria como:

$$X = \sum_{i=1}^{4n-1} 2^i x_i = 2^{3n} N_3 + 2^{2n} N_2 + 2^n N_1 + N_0, \tag{1}$$

onde os arrays $N_3 = \{x_{(4n-1)}, \ldots, x_{(3n+1)}, x_{3n}\}$, $N_2 = \{x_{(3n-1)}, \ldots, x_{(2n+1)}, x_{2n}\}$, $N_1 = \{x_{(2n-1)}, \ldots, x_{(n+1)}, x_n\}$ e $N_0 = \{x_{(n-1)}, \ldots, x_1, x_0\}$. Usando notação binaria e conjunto de módulos $\{m_1, m_2, m_3\} = \{2^{2n}, 2^n - 1, 2^n + 1\}$, a faixa dinâmica do valor X é $[0, M-1]$, onde $M = m_1 m_2 m_3$. Três conversores são necessários de modo a obter a representação do RNS, um para cada elemento de base.

- **Canal** $m_1 = 2^{2n}$**:** O canal mais simples é o conversor usando o modulo m_1. O valor $|X|_{m_1}$ pode ser obtido pelo resto da divisão do X por 2^{2n} , o que pode por conseguida por médio de truncar o valor de X, uma vez que:

$$|X|_{m_1} = \overbrace{|2^{3n}|_{m_1}}^{=0} N_3 + \overbrace{|2^{2n}|_{m_1}}^{=0} N_2 + 2^n N_1 + N_0 = \{x_{(2n-1)}, \ldots, x_1, x_0\}. \tag{2}$$

- **Canal** $m_2 = 2^n - 1$**:** Devido a que $|2^n|_{2^n-1} = 1$, podemos expressar a Eq. 1 como:

$$|X|_{m_2} = |N_3 + N_2 + N_1 + N_0|_{2^n-1} = |N_3 + |N_2 + N_1 + N_0|_{2^n-1}|_{2^n-1}. \tag{3}$$

- **Canal** $m_3 = 2^n + 1$**:** Devido a que $|2^n|_{2^n+1} = -1$, podemos expressar a Eq. 1 como:

$$|X|_{m_3} = |N_3 - N_2 + N_1 - N_0|_{2^n+1} = |-N_3 + |N_2 - N_1 + N_0|_{2^n+1}|_{2^n+1}. \tag{4}$$

3 Familiarização com ferramenta Quartus e placa DE2

O Quartus II é um software utilizado para o projeto de circuitos e sistemas lógicos com foco nos dispositivos lógicos programáveis (FPGAs – field-programmable gate arrays - e CPLDs – complex programmable logic devices) produzidos pela Altera Corporation. A DE2 é uma placa educacional e de desenvolvimento que possui, além de diversos periféricos (chaves, botões, leds, displays de cristal líquido e 7 segmentos, rede, wireless, VGA, serial, dentro outros), um FPGA da Altera como componente principal. Assim, iremos utilizar, ao longo desse semestre, o Quartus II para a realização dos projetos de circuitos lógicos e a DE2 para o seu teste e avaliação.

Ligando a placa DE2:

- O primeiro passo para ligar a placa DE2 é retirá-la de sua caixa e colocá-la sobre a bancada. Atenção: o manuseio da DE2 deve ser feito com muito cuidado. Segure a placa apenas pelas suas laterais ou pela proteção acrílica superior transparente. Evite colocar os dedos nos componentes uma vez que a energia estática do seu corpo pode danificá-los.
- Com a placa colocada sobre a bancada, pegue a fonte de energia, também disponível na caixa da DE2, e ligue essa fonte no conector indicado como 9V DC Power Supply Connector na Figura 2. Com a fonte ligada na placa, ligue também a sua outra extremidade na tomada.
- Agora, pegue o cabo USB disponível na caixa da placa, ligue uma extremidade desse cabo na USB Blaster Port, indicada na Figura 2, e a outra extremidade em uma das portas USB do computador. Atenção: não ligue o cabo USB na entrada USB Device da placa pois nesse caso não será possível estabelecer comunicação com o computador.
- Realizados os passos anteriores, aperte o botão liga/desliga (ver Power ON/OFF Switch na Figura 2) e sua placa deverá funcionar corretamente.

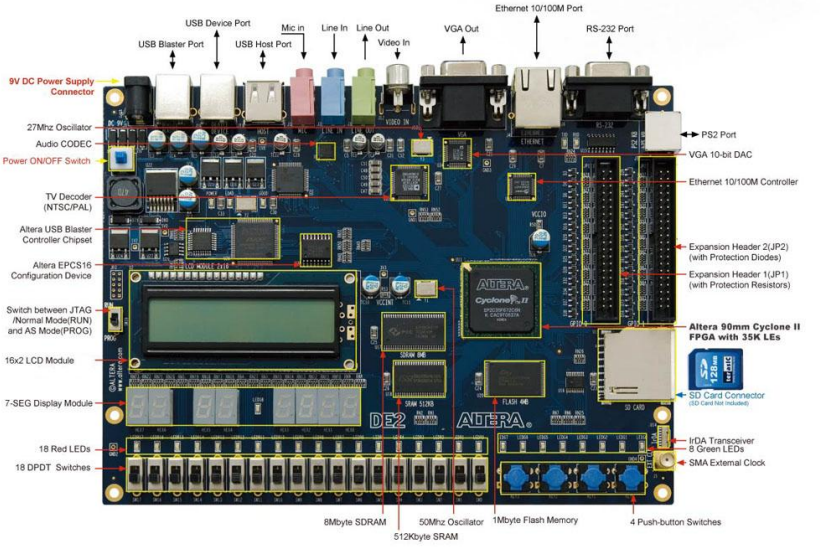

Figura 2: Visão geral da placa DE2 da Altera.

4 Implementação em VHDL do conversor Binario-RNS

- Com a placa em funcionamento, baixe o arquivo "$Lab01a.zip$" disponível no site da disciplina e descompacte esse arquivo na pasta "$/Desktop/EEL510269/lab01$". Atenção: o caminho do diretório para o qual o arquivo será descompactado não deve conter espaços.
- Agora, execute o software Quartus II 13.0sp1 Web Edition (a versão 12.1sp2 também pode ser utilizada). Com o software em funcionamento, acesse o menu File e a opção Open Project (File → Open Project) e abra o projeto disponível na pasta destino da descompactação. Atenção: não use a opção "File → Open"para abrir o projeto, mas sim a "File → Open Project".
- Uma vez aberto o projeto, clique na entidade "$Traditionalsystem\_bintoRNS$" disponível na aba Hierarchy do Project Navigator do Quartus II.
- Com o projeto e a entidade principal abertos, você deverá ver uma janela com a descrição em VHDL do conversor Binário-RNS.

Nas linhas $1 \rightarrow 30$ estão definidas as livrarias, as entradas e saídas da estrutura estão definidas nas linhas $33 \rightarrow 37$ e a arquitetura a partir da linha 41.

4.1 Tarefa a ser realizada na sala de aula

Agora o aluno deve preencher partes do código VHDL dos três canais modulares para assim obter um conversor Binario-RNS usando o conjunto de modulos $\{2^{2n}, 2^n - 1, 2^n + 1\}$ e $n = 4$. Nota: O aluno tem de ter em consideração que as entradas do circuito $X = \{x_{(4n-1)}, \ldots, x_1, x_0\}$ estão associadas aos Switches 15 a 0, *SW: in STD_ LOGIC_ VECTOR(4*n-1 downto 0)*, e as saidas estão associadas aos LEDs vermelhos 16 a 0, *LEDR: out STD_ LOGIC_ VECTOR(4*n downto 0)* como é mostrado na seguinte Figura.

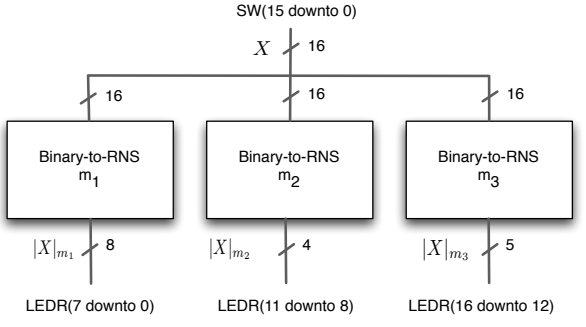

Figura 3: Bloco binario-RNS com associação de pinos entrada-saida.

Para a implementação do canal $m_1 = \{2^{2n}\}$ defina o array necessário a ser incluído na linha de código 83. Dica: use a Eq. 2.

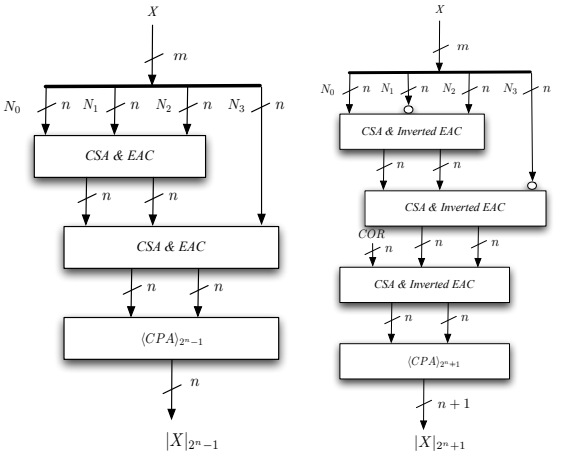

Figura 4: Diagrama de blocos para conversor binario-RNS modulo a) $m_2 = \{2^n - 1\}$ e b) $m_3 = \{2^n + 1\}$.

Para a implementação do canal $m_2 = \{2^n - 1\}$ usaremos a Eq. 3. O diagrama de blocos para este canal é mostrado em Fig. 4a. Cada Carry Save Adder (CSA) soma três termos fornecendo os arrays Acarreio $C = c_n, ..., c_2, c_1$ e Soma $S = s_{n-1}, ..., s_1, s_0$. Tendo em consideração que $|2^n c_n|_{2^n-1} = c_n$, podemos recolocar o bit c_n na posição 2^0 (i.e: operação End-Around-Carry).

O CSA+EAC é implementado no VDHL $CSA\_2n\_mp\_1.vhd$ previamente fornecido usando o bit de controle $modo = 0$. Finalmente a soma na ultima etapa é feita por um Carry Propagate Adder (CPA) também com EAC usando $adder\_2n\_mp\_1.vhd$ com o bit de controle $modo = 0$. **Preencha o resto do VHDL linhas a partir da** 83 **usando os sinais auxiliares dados nas linhas 65 a 73.**

Para a implementação do canal $m_3 = \{2^n + 1\}$ usaremos a Eq. 4. O diagrama de blocos para este canal é mostrado em Fig. 4b. Cada Carry Save Adder (CSA) soma de novo três termos fornecendo os arrays Acarreio $C = c_n, ..., c_2, c_1$ e Soma $S = s_{n-1}, ..., s_1, s_0$. Tendo em consideração que $|2^n c_n|_{2^n+1} = ||2^n|_{2^n+1} c_n|_{2^n+1} = |-c_n|_{2^n+1} = |COR_{level-j} + \bar{c_n}|_{2^n+1}$, podemos recolocar o bit c_n na posição 2^0 de forma complementada (i.e: operação Inverted End-Around-Carry) adicionando um factor corretor $COR_{level-j}$, onde j define qual nível de CSA com inverted EAC está associado dito factor corretor. O factor corretor $COR_{level-j}$ pode ser calculado a partir da seguinte equação $|COR_{level-j} + \bar{c_n}|_{2^n+1} = 0$ quando $c_n = 0$ (i.e. $\bar{c_n} = 1$). Desta forma $COR_{level-j} = 2^n$ por nivel CSA-inverted EAC.

Os termos negativos $-N_0$ e $-N_2$ seguem a mesma regra $|-N_i|_{2^n+1} = |COR_{Ni} +$

Tabela 1: Tabela de resultados de simulação

	$X = 0$	$X = 511$	$X = 1020$	$X = 3490$
Canal m_1				
Canal m_2				
Canal m_3				

Tabela 2: Tabela de resultados na placa DE2

	$X = 0$	$X = 511$	$X = 1020$	$X = 3490$
Canal m_1				
Canal m_2				
Canal m_3				

$\bar{N}_i|_{2^n+1}$, onde i define qual array de N_i está associado dito factor corrector. O factor corretor COR_{Ni} pode ser calculado a partir da seguinte equação $|COR_{Ni}+\bar{N}_i|_{2^n+1} = 0$ quando $N_i = 0$ (i.e. $\bar{N}_i = 2^n - 1$). Desta forma $COR_{Ni} = 2$ para cada $-N_i$.

O factor corrector final consiste na suma modular de todas as correcções parciais $COR = |\sum_{j=1,2,3,4} COR_{level-j} + \sum_{i=0,2} COR_{Ni}|_{m_3}$.

O CSA+Inverted EAC é implementado no VDHL no arquivo CSA_2n_mp_1.vhd previamente fornecido usando o bit de control $modo = 1$. Finalmente a soma na ultima etapa é feita por um Carry Propagate Adder (CPA) também com EAC usando $adder\_2n\_mp\_1.vhd$ com o bit de controle $modo = 1$ (esta ultima etapa também contribui com um $COR_{level-j}$ como está indicado na equação do COR). **Preencha o resto do VHDL linhas a partir da** 87 **usando os sinais auxiliares dados nas linhas 65 a 73.**

Uma vez preenchido o VHDL compile o projeto até não ter erros na descrição. Uma vez compilado abra modelsim e simule o circuito. Dica: use o script .do para forçar as entradas. **Preencha a tabela 1 com os dados da simulação.**

Uma vez terminada a simulação implemente na placa DE2 o circuito e preencha a tabela 2 com os resultados obtidos nos LEDs vermelhos. Importante: antes de desligar o computador, guarde a pasta lab1a em um pendrive ou envie por e-mail já que este circuito sera usado nos seguintes laboratórios.

4.2 Questões finais

- **Pergunta 1:** A operação End-Around-Carry é uma operação que impõe aumento de hardware quando é usado num circuito?.

- **Pergunta 2:** Se o factor corretor total pode ser expressado como a soma modular $2^n + 1$ de todos os termos, qual sería o valor do factor corretor a adicionar na terceira etapa de CSA+Inverted EAC?.