Universidade Federal de Santa Catarina
EEL7123/EEL510457
Semestre: 2020/2 – Lab1 RNS

Conversor Binario-RNS

1 Introdução e objetivos

O objetivo 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 nesta aula $LAB1\_RNS$, ii) unidades aritméticas RNS (RNS aritmethic units) que serão vistas na aula $LAB3\_RNS$ e iii) conversor RNS a binário (RNS-to-Binary converters) que será visto na aula $LAB2\_RNS$.

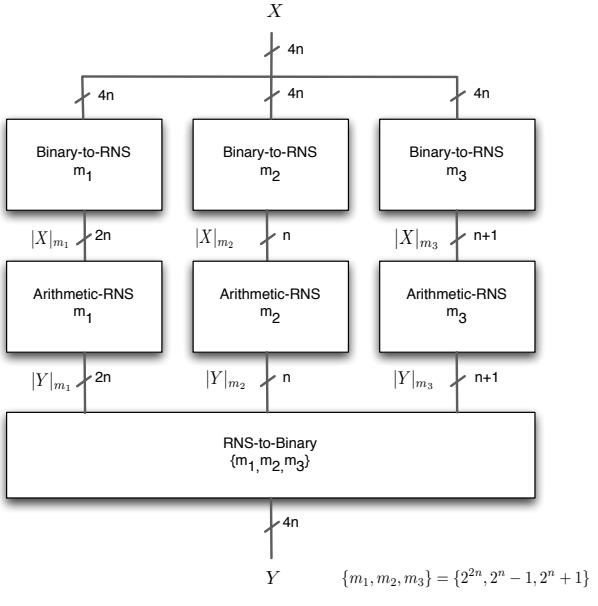

Figura 1: Unidade RNS completa usando conjunto de módulos $\{m_1, m_2, m_3\} = \{2^n, 2^n - 1, 2^n + 1\}$.

2 Conversor Binário-RNS

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

$$X = \sum_{i=0}^{4n-1} 2^i x_i = 2^{3n} N_3 + 2^{2n} N_2 + 2^n N_1 + N_0, \tag{1}$$

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 para obter a representação do RNS, um para cada elemento de base.

- **Caminho $m_1 = 2^{2n}$:** 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 ser obtida por meio do truncamento do valor de X, uma vez que:

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

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

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

- **Caminho $m_3 = 2^n + 1$:** 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 Implementação em VHDL do conversor Binario-RNS

- Baixe o arquivo "*Lab*01.*zip*" disponível no site da disciplina e descompacte esse arquivo no seu computador. Atenção: o caminho do diretório para o qual o arquivo será descompactado não deve conter espaços.
- Agora, execute o software Quartus II (já seja no computador pessoal ou via x2goclient como foi explicado na aula de "Introdução de Laboratório"). 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.

3.1 Tarefa a ser realizada

Agora o aluno deve preencher partes do código VHDL dos três caminhos 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 levar 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.

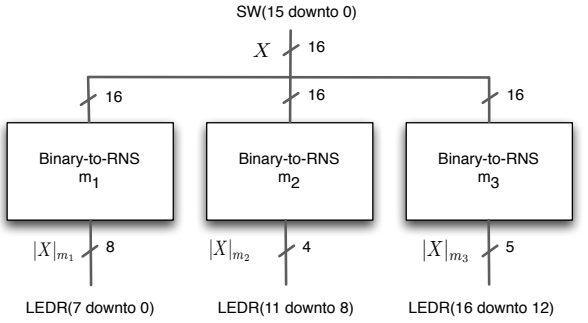

Figura 2: Bloco binario-RNS com associação de pinos entrada-saida.

Para a implementação do caminho $m_1 = \{2^{2n}\}$ defina o array necessário a ser incluído. Dica: use a Eq. 2.

Para a implementação do caminho $m_2 = \{2^n - 1\}$ usaremos a Eq. 3 a qual pode ser implementada a partir de uma arvore de CSA com End-Around-Carry (EAC) que somam os 4 vectores N_3, N_2, N_1 e N_0 (Figura 4a). **O aluno deve implementar os CSA com End-Around-Carry (EAC) preenchendo o arquivo** $CSA\_EAC.vhd$ **e a arvore no arquivo** $Traditionalsystem\_bintoRNS.vhd$. Finalmente a soma na ultima etapa é feita por um somador modulo $\{2^n-1\}$ ($CPA\_mod15.vhd$) o qual é fornecido no Moodle (não usaremos um CPA com End-Around-Carry (EAC) para evitar problemas de estabilidade no simulador e emulador).

Para a implementação do caminho $m_3 = \{2^n + 1\}$ usaremos a Eq. 4 a qual pode ser implementada a partir de uma arvore de CSA com Inverted-End-Around-Carry (IEAC) que somam os 5 vectores N_3, $\bar{N}_2$, N_1, $\bar{N}_0$ e um factor corrector COR (Figura 4b). **O aluno deve implementar os CSA com End-Around-Carry (EAC) preenchendo o arquivo** $CSA\_EAC.vhd$ **e a arvore no arquivo** $Traditionalsystem\_bintoRNS.vhd$. Finalmente a soma na ultima etapa é feita por um somador modulo $\{2^n + 1\}$ ($CPA\_mod17.vhd$) o qual é fornecido no Moodle (não

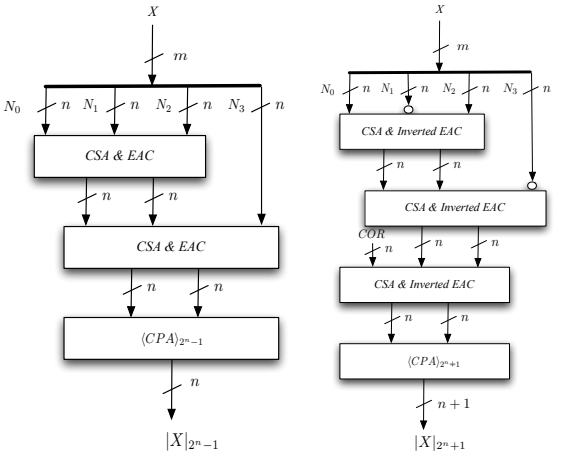

Figura 3: Diagrama de blocos para conversor binario-RNS modulo a) $m_2 = \{2^n - 1\}$ e b) $m_3 = \{2^n + 1\}$.

usaremos um CPA com Inverted-End-Around-Carry (IEAC) para evitar problemas de estabilidade no simulador e emulador). Para o calculo do factor corrector o aluno pode fazer uma simulação com $X = 0$, $COR = 0$ e, por meio do valor de saída Y, obter o valor do factor corrector usando a equação $|Y + COR|_{2^n+1} = 0$ e substituir o valor do COR correto no VHDL.

3.2 Simulação usando Modelsim

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.**

Tabela 1: Tabela de resultados de simulação

	$X = 0$	$X = 511$	$X = 1020$	$X = 65279$
Canal m_1				
Canal m_2				
Canal m_3				

Visualize o diagrama de blocos gerado Tools-> Netlist Viewers-> RTL Viewer. Faça print do diagrama de blocos.

3.3 Emulação na placa DE2

Com a emulação, o fluxo de projeto → simulação → emulação será completado. Os passos a seguir para a emulação são mostrados a seguir:

- **Passo 1:** Renomear o arquivo "$Traditionalsystem\_bintoRNS.vhd$"como "$usertop.vhd$".
- **Passo 2:** A entity deve ser a seguinte:

```
entity usertop is
generic(n :  natural := 4);
port(
CLOCK_50:  in std_logic;
CLK_500Hz:  in std_logic;
RKEY: in std_logic_vector(3 downto 0);
KEY: in std_logic_vector(3 downto 0);
RSW: in std_logic_vector(17 downto 0);
SW: in std_logic_vector(17 downto 0);
LEDR: out std_logic_vector(17 downto 0);
HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7:  out std_logic_vector(6 downto
0));
end usertop;
```

- **Passo 3.a (via emulador via x2goclient/VPN)**: Com as devidas adaptações feitas nos arquivos abra o terminal de comandos (ver tutorial de acesso remoto ao Quartus no Moodle) e digite:
  `cd /path`
  onde "path" é o caminho até os arquivos VHDL que deseja utilizar na emulação.
  Em seguida, compile com o comando (os .vhd nessa ordem):
  `fpgacompile fulladder.vhd CSA_EAC.vhd CSA_IEAC.vhd Mux2_1.vhd CPA_mod15.vhd CPA_mod17.vhd usertop.vhd`
  Para emular o circuito, na mesma pasta, digite
  `./fpgatest`
  Assim que fizer isso, uma janela com a emulação do seu circuito abrirá. Quando desejar terminar a emulação, feche a janela e, em seguida, use Ctrl+C para liberar o terminal.
  Ao final da utilização, não se esqueça de efetuar o logout no X2Go Client.

- **Passo 3.b (via emulador on-line):** Fazer a emulação na placa DE2 usando o emulador on-line (sem necessidade de ter o VPN da UFSC ligado). Com as devidas adaptações feitas nos arquivos abra o emulador `http://150.162.54.54:5000/` e faça UPLOAD dos arquivos VHDL que deseja utilizar na emulação. Defina usertop.vhd como topo da hierarquia usando o botão SET TOP LEVEL e compile. Em seguida, vai para o emulador, inicie a emulação e verifique o funcionamento.
- **Passo 4**: Faça print ou fotografia dos resultados de saída para as quatro combinações de entrada $X = 0$, $X = 511$, $X = 1020$, $X = 3490$. Explique os resultados obtidos no $Lab1\_RNS.doc$.

3.4 Entrega de material

Os alunos deverão entregar na tarefa disponível no Moodle com:

- Os arquivos VHDLs da tarefa.
- O arquivo *Lab*1_*RNS.doc* incluindo o print de tela da simulação das 4 combinações de entrada, a tabela 1 preenchida indicando o factor corrector (COR), o diagrama de blocos, os prints de tela da emulação e as dúvidas que tiveram (caso existam). O aluno deve explicar os resultados obtidos na simulação e emulação. O aluno pode incluir na tarefa de forma opcional um vídeo mostrando o funcionamento no emulador para combinações pedidas.