Universidade Federal de Santa Catarina
EEL7123/EEL510457
Semestre: 2020/2 – Lab2 RNS

Conversor RNS-binario

1 Introdução e objetivos

O objectivo deste laboratório consiste em projetar em FPGA uma unidade conversora de numeração residual (RNS) a binário 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) já vistas no laboratório Lab1_RNS, ii) unidades aritméticas RNS (RNS aritmethic units) que serão vistas no Lab3_RNS e iii) conversores RNS a binário (RNS-to-Binary converters) que será visto nessa aula Lab2_RNS.

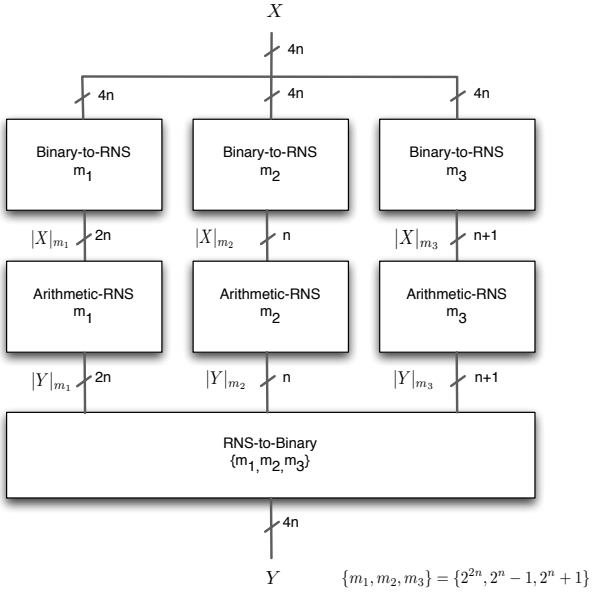

Figura 1: Unidade RNS completa usando conjunto de módulos $\{m_1, m_2, m_3\} = \{2^n, 2^n - 1, 2^n + 1\}$.

2 Conversor RNS-binario

A conversão de RNS a binario é obtida usando o novo Teorema Chinés do Resto (CRT), visto na aula teorica.

$$X = \left| \sum_{i=1}^{3} V_i R_i \right|_{\hat{m}_1} m_1 + R_1, \tag{1}$$

onde $V_1 = \frac{\left|\hat{m}_1^{-1}\right|_{m_1} \hat{m}_1 - 1}{m_1}$, $V_2 = \frac{\left|\hat{m}_2^{-1}\right|_{m_2} \hat{m}_2}{m_1}$, $V_3 = \frac{\left|\hat{m}_3^{-1}\right|_{m_3} \hat{m}_3}{m_1}$, $\hat{m}_i = \frac{M}{m_i}$, $\left|\hat{m}_i^{-1}\right|_{m_i}$ é a multiplicativa inversa de $\hat{m}_i$ e R_i as entradas residuais. Para o moduli set $\{m_1, m_2, m_3\}$, onde $m_1 = 2^8$, $m_2 = 2^4 - 1$, $m_3 = 2^4 + 1$ escolhido obtemos $\hat{m}_1 = 2^8 - 1$, $\hat{m}_2 = 2^8(2^4 + 1)$, $\hat{m}_3 = 2^8(2^4 - 1)$, $\left|\hat{m}_1^{-1}\right|_{m_1} = 2^8 - 1$ $\left|\hat{m}_2^{-1}\right|_{m_2} = 2^3$ e $\left|\hat{m}_3^{-1}\right|_{m_3} = 2^3$.

Eq.(1) pode se escrever como:

$$X = \left| \overbrace{|2^8 - 2|_{\hat{m}_1}}^{V_1=-1} R_1 + \overbrace{(2^{8-1} + 2^{4-1})}^{V_2} R_2 + \overbrace{(2^{8-1} - 2^{4-1})}^{V_3} R_3 \right|_{\hat{m}_1} m_1 + R_1 =$$

$$= \left| \overline{R_1} + 2^7 R_2 + 2^3 R_2 + 2^7 R_3 + 2^3 \overline{R_3} \right|_{2^8-1} 2^8 + R_1. \tag{2}$$

Podemos finalmente expressar os termos da Eq.2 como:

$$\begin{aligned} A =& \bar{r}_{1,7}\bar{r}_{1,6}\bar{r}_{1,5}\bar{r}_{1,4}\bar{r}_{1,3}\bar{r}_{1,2}\bar{r}_{1,1}\bar{r}_{1,0}. \\ B =& r_{2,0}r_{2,3}r_{2,2}r_{2,1}r_{2,0}r_{2,3}r_{2,2}r_{2,1} \\ C =& r_{3,0}000r_{3,4}r_{3,3}r_{3,2}r_{3,1} \\ D =& \bar{r}_{3,4}\bar{r}_{3,3}\bar{r}_{3,2}\bar{r}_{3,1}\bar{r}_{3,0}000 \end{aligned} \tag{3}$$

Devido a que alguns termos são números negativos, vamos precisar do factor corrector, o qual é calculado forçando as entradas R_1, R_2, R_3 a zero na operação modular e corrigir a saída (a qual deve ser zero) colocando um fator corretor:

$$\left| \overbrace{\overline{R_1}}^{=255} + \overbrace{2^7 R_2}^{=0} + \overbrace{2^3 R_2}^{=0} + \overbrace{2^7 R_3}^{=0} + \overbrace{2^3 \overline{R_3}}^{=248} + COR \right|_{2^8-1} = 0 \rightarrow COR = 7. \tag{4}$$

O valor de $COR = 7_{10} = 111_2$ encaixa no vector D da Eq.3 mantendo unicamente 4 vectores A, B, C e D:

$$\begin{aligned} A =& \bar{r}_{1,7}\bar{r}_{1,6}\bar{r}_{1,5}\bar{r}_{1,4}\bar{r}_{1,3}\bar{r}_{1,2}\bar{r}_{1,1}\bar{r}_{1,0}. \\ B =& r_{2,0}r_{2,3}r_{2,2}r_{2,1}r_{2,0}r_{2,3}r_{2,2}r_{2,1} \\ C =& r_{3,0}000r_{3,4}r_{3,3}r_{3,2}r_{3,1} \\ D =& \bar{r}_{3,4}\bar{r}_{3,3}\bar{r}_{3,2}\bar{r}_{3,1}\bar{r}_{3,0}111 \end{aligned} \tag{5}$$

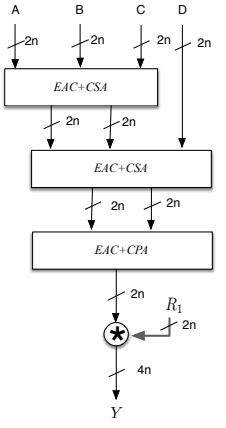

Figura 2: Diagrama de blocos para conversor RNS-binario para o modulo $\{m_1, m_2, m_3\} = \{2^{2n}, 2^n - 1, 2^n + 1\}$.

Pelo que a Eq.2 pode se reescrever como:

$$X = |A + B + C + D|_{2^8-1} m_1 + R_1, \tag{6}$$

A Fig. 2 mostra o diagrama de blocos para a implementação da Eq.6 .

3 Implementação em VHDL do conversor RNS-Binario

- Baixe o arquivo "$Lab2.zip$" disponível no site da disciplina e descompacte esse arquivo no computador. Atenção: o caminho do diretório para o qual o arquivo será descompactado não deve conter espaços.
- Agora, execute o software Quartus II. Com o software em funcionamento, acesse o menu File e a opção Open Project (File → Open Project) para abrir 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 $traditionalSystem\_RNStoBin$ 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 RNS-binario.

Tabela 1: Tabela de resultados de simulação

Canal m_1	**Canal** m_2	**Canal** m_3	X
0	0	0	
255	1	1	
252	0	0	
255	14	16	

3.1 Tarefa a ser realizada

Agora o aluno deve preencher partes do código VHDL para assim obter um conversor RNS-Binario usando o conjunto de módulos $\{2^{2n}, 2^n - 1, 2^n + 1\}$ e $n = 4$. Nota: O aluno tem de levar em consideração que as entradas do circuito $R_1 = \{r_{1,(2n-1)}, \ldots, r_{1,1}, r_{1,0}\}$, $R_2 = \{r_{2,(n-1)}, \ldots, r_{2,1}, r_{2,0}\}$, e $R_3 = \{r_{(3,n)}, \ldots, r_{3,1}, r_{3,0}\}$ estão associadas aos Switches 7 a 0, R_1, Switches 11 a 8, R_2, Switches 16 a 12, R_3, e as saídas estão associadas aos LEDs vermelhos 15 a 0 como é mostrado na Fig. 3.

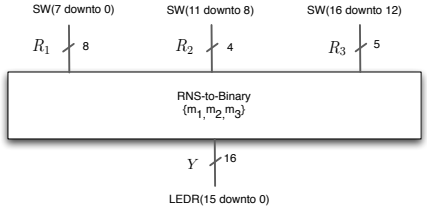

Figura 3: Bloco RNS-binario com associação de pinos entrada-saida.

Para a implementação da soma modular apresentada na Eq.6, usaremos uma arvore de CSA com End-Around-Carry que somam os 4 vectores A, B, C e D (Figura 2). O aluno deve implementar os CSA com End-Around-Carry (EAC) preenchendo o arquivo $CSA\_EAC.vhd$ e a arvore no arquivo $Traditionalsystem\_RNStoBin.vhd$. Finalmente a soma na ultima etapa é feita por um somador modulo $\{2^{2n} - 1\}$ ($CPA\_mod255.vhd$) o qual é fornecido no Moodle (não usaremos um CPA com End-Around-Carry (EAC) para evitar problemas de estabilidade no simulador e emulador).

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 um script .do para forçar as entradas. **Preencha a tabela 1 com os dados da simulação.**

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 se mostram a seguir:

- **Passo 1:** Renomear o arquivo "*Topo.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 (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 Mux2_1.vhd CPA_mod255.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 (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 apresentada na tabela 1. Explique os resultados obtidos no *Lab2_RNS.doc*.

3.4 Entrega de material

Os alunos deverão entregar na tarefa disponível no Moodle com:

- Os arquivos VHDLs da tarefa.
- O arquivo *Lab2_RNS.doc* incluindo o print de tela da simulação das 4 combinações de entrada, a tabela 1 preenchida, 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.