PROGRAMAÇÃO EM VHDL DE CIRCUITOS LÓGICOS PARA IMPLEMENTAÇÃO EM FPGA

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA

(PIBIC/CNPq/INPE)

Yegor Gomes de Mello (UFRN, Bolsista PIBIC/CNPq)
E-mail: yegor_melo@crn.inpe.br

Manoel Jozeane Mafra da Carvalho (INPE/CRN, Orientador)
E-mail: Manoel@crn.inpe.br
Dr. Ana Maria Guimarães Guerreiro (UFRN/DCA, Orientadora)
E-mail: anamaria@dca.ufrn.br

COLABORADOR

José Marcelo Lima Duarte (INPE/CRN, Mestrado)
E-mail: jmarcelo@crn.inpe.br

Julho de 2006

De novo, lhes falava Jesus, dizendo:
Eu sou a luz do mundo; quem me segue não andaras nas trevas;
Pelo contrário terá a luz da vida.
(JOÃO 8:12)

Aos meus pais,
EDILSON BRITO DE MELO e
GERALDA GOMES DE MELO.

AGRADECIMENTOS

Agradeço a todas pessoas que me ajudaram a vencer mais esta etapa do Curso de Graduação em Eng. Elétrica.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – Bolsa de Iniciação Científica – PIBIC.

Ao Instituto Nacional de Pesquisas Espaciais - INPE pela oportunidade de estudos e utilização de suas instalações.

Aos amigos Bolsistas da Estação Multi Missão, Eng. José Marcelo, aos colegas Graduandos Kurius Yuri e Francisco de Assis pela amizade e companheirismo demonstrados.

Ao orientador Tecnologista sênior do INPE-CRN Manoel Jozeane Mafra da Carvalho pelo conhecimento passado, e pela orientação e apoio na realização deste trabalho.

À orientadora Prof. Dr. Ana Maria Guerreiro pela orientação, quem tem dado bastante apoio na realização desse trabalho.

A meus pais por sempre acreditarem na importância do estudo.

PROGRAMAÇÃO EM VHDL DE CIRCUITOS LÓGICOS PARA IMPLEMENTAÇÃO EM FPGA

RESUMO

Esse Projeto foi iniciado em Fevereiro de 2006, com o objetivo de desenvolver códigos de descrição de Circuitos Lógicos bem como a implementação dos mesmos. A aplicação inicial desse projeto foi dar auxílio a um outro projeto de Demodulação de Sinais usando o Costas Loop, um tipo de PLL “Phase-Locked Loop” (ou Malha de Captura de Fase). O auxílio dado ao projeto é referente à Programação em VHDL "**V**HSIC **H**ardware **D**escription **L**anguage” (Circuito Integrado de Altíssima Velocidade em Linguagem de Descrição de Hardware), linguagem usada para facilitar o design de circuitos digitais em FPGAs “Field Programable Gate Array” (Matriz de Portas Lógicas Programáveis no “Campo”).

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

HDL	**-**	**Hardware Description Language**
VHDL	**-**	**VHSIC Hardware Description Language**
FPGA	**-**	**Field Programmable Gate Array**
EMMN	**-**	**Estação Multi Missão de Natal**
A/D	**-**	**Conversor Analógico Digital**
LUT	**-**	**Look Up Table**
ATAN	**-**	**Arco Tangente**
PLL	**-**	**Phase-Locked Loop**

CAPÍTULO 1

INTRODUÇÃO

Uma linguagem de descrição de circuitos (HDL) possibilita várias aplicações, como no teste de circuitos e na síntese do circuito descrito. O Projeto de Programação em VHDL de Circuitos Lógicos para Implementação em FPGA está utilizando esta linguagem, pois a mesma pode suportar projetos com múltiplos níveis de hierarquias o que é muito importante para a aplicação inicial deste projeto, ou seja, dar auxílio ao Projeto do Demodulador PM Digital para ser implementado em FPGA para Estação EMMN do CRN-INPE, o Demodulador utiliza o algoritimo Costas Loop para fazer o rastreamento da fase e frequência da portadora.

Para isso foi necessário adquirir o Kit Cyclone II EP2C35 FPGA, fabricado pela Altera, juntamente com o programa de simulação e implementação Quartus II, fornecido pelo fabricante, adquirido pelo INPE. Na figura a seguir é mostrada a plataforma adquirida.

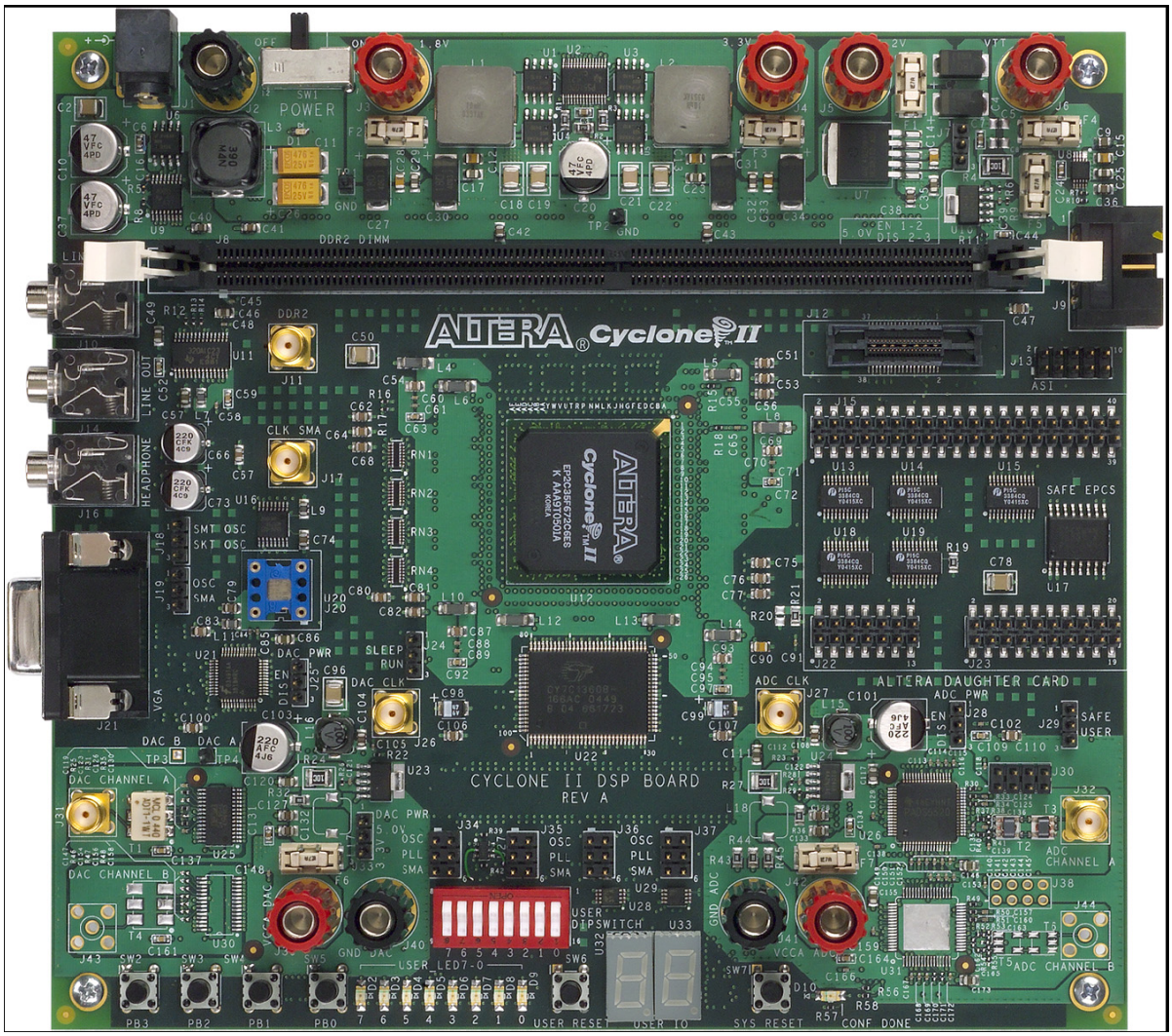

FIGURA 1.1 – Plataforma Cyclone II EP2C35 FPGA

FONTE: **Development Kit Version 1.0.0 Altera Corporation**
Cyclone II EP2C35 DSP Development Board Reference Manual

O projeto foi iniciado com o estudo de VHDL acompanhando resoluções de exercícios, simulações e implementações. Para realizar as simulações, como já foi dito, foi utilizado o Software Quartus II. O mesmo possui ferramentas totalmente integradas, dentre elas temos:

- Entrada do projeto por múltiplos métodos
- Síntese lógica
- Place & Route
- Simulação
- Timing & Power Analysis
- Programação do dispositivo

O Quartus II foi estudado em conjunto com a linguagem. Na seguir é mostrado o ambiente do Quartus II.

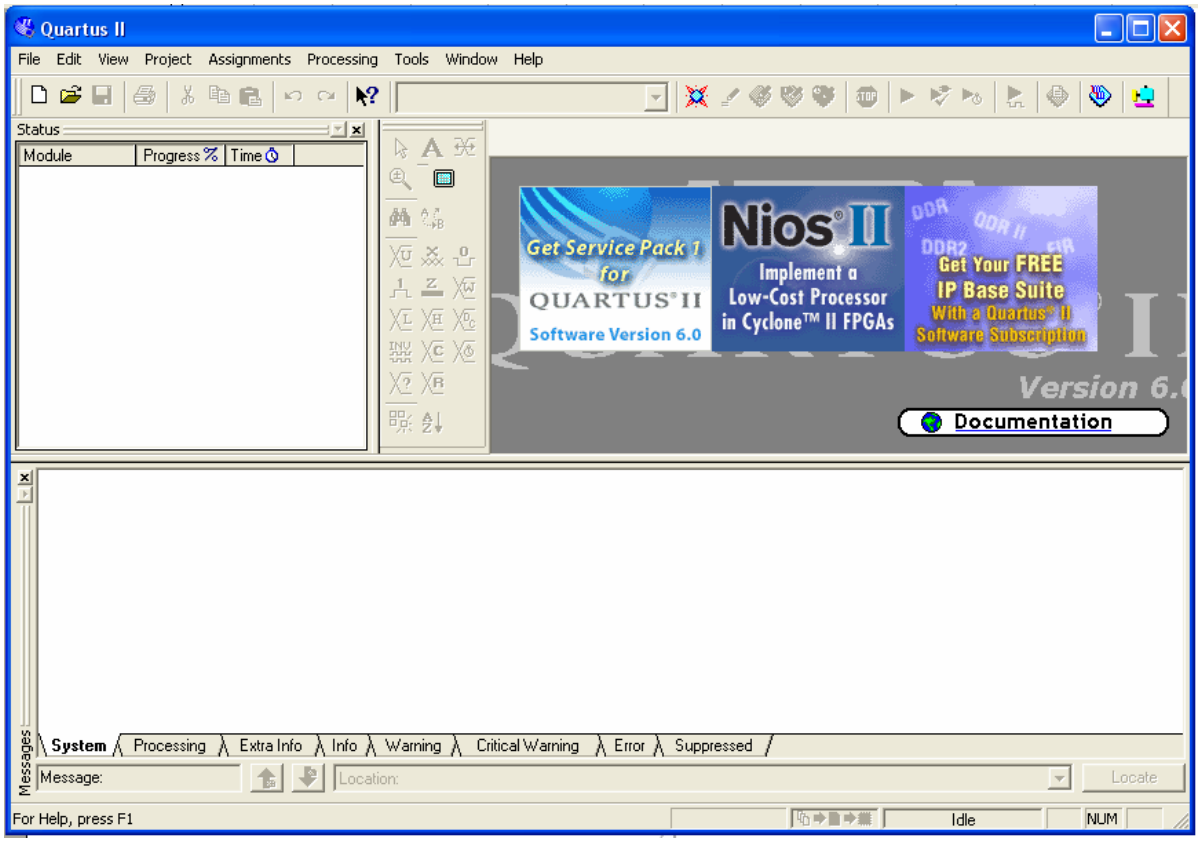

FIGURA 1.2 – Ambiente do Software utilizado para as simulações

FONTE: **Quartus II Altera Corporation**

O projeto foi dividido nas seguintes etapas:

1. Estudo da Linguagem
2. Estudo do Software
3. Desenvolvimento de projetos
4. Descrições e Simulações
5. Implementações

Inserido nas etapas acima existe o Projeto de Demodulação, que será descrito mais adiante.

CAPÍTULO 2

DESENVOLVIMENTO – PROJETO DE DEMODULAÇÃO

2.1 Costas Loop

O demodulador do tipo Costas Loop será implementado numa placa para desenvolvimento de processamento digital de sinais em FPGA. Para digitalização dos dados é utilizado um conversor A/D. Na figura a seguir teremos demonstra o Costas Loop.

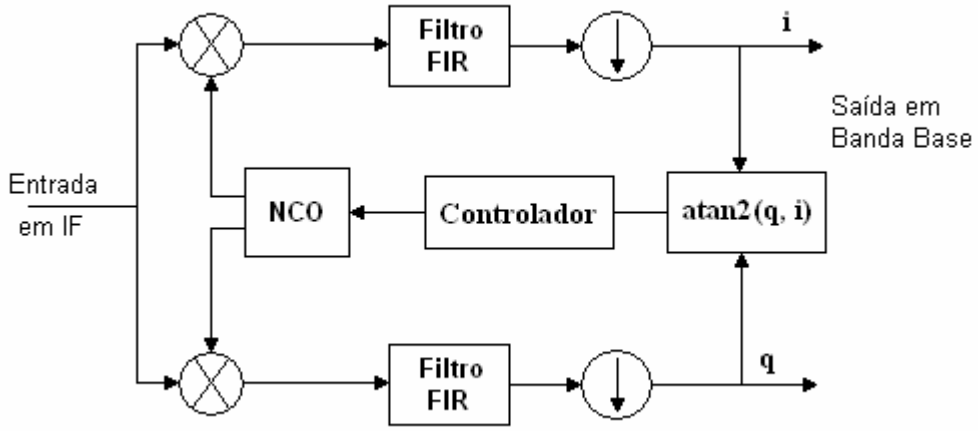

FIGURA 2.1 – Fluxo de Demonstração do Costas Loop

FONTE: **INPE – CRN**

Com relação ao Costas Loop o enfoque do meu trabalho é o cálculo do Arco Tangente, que será descrito a seguir.

2.1.1 Arco Tangente

A alternativa escolhida para o cálculo do Arco Tangente será o método da LUT (Look Up Table). Tal método consiste em determinar os valores do ângulo a partir de uma tabela previamente elaborada, essa tabela terá valores de ângulos apenas para o primeiro quadrante. O ATAN(X) é completamente linear para $0 < x < 1$ e não linear para $x > 1$, com isso será feito o armazenamento apenas para valores de $0 < x < 1$, onde o x correspondente será obtido a partir dos valores do eixo da ordenada (I) e o eixo das abscissas (Q) pela divisão (I) / (Q) ou (Q) / (I). Dependendo dos valores de Q ou de I será adicionada uma lógica para obtermos o valor referente ao primeiro quadrante com uma correção adequada para cada caso.

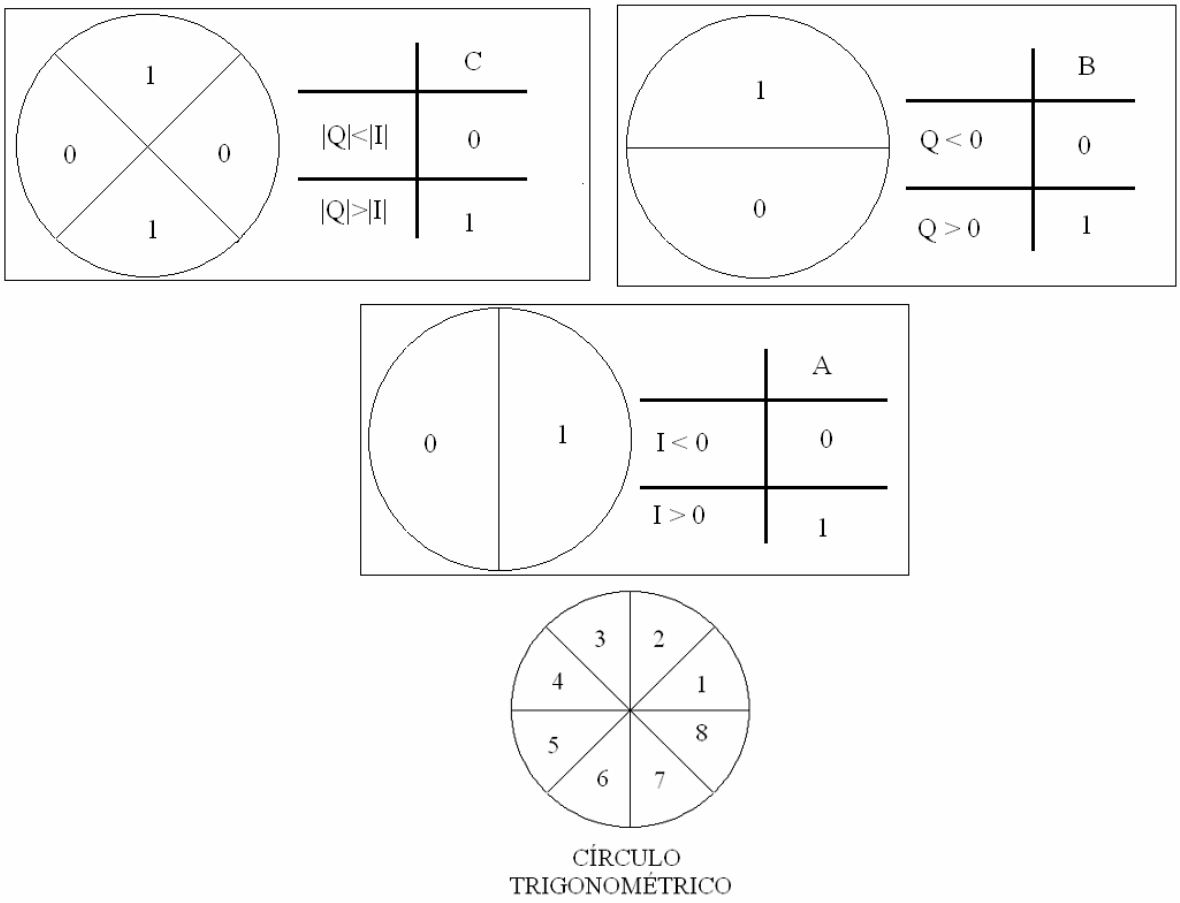

FIGURA 2.2 – Esquema da lógica utilizada com a correção.

FONTE: **INPE – CRN**

TABELA 2.1

C	B	A	QUADRANTE	CORREÇÃO
0	0	0	5	$y - \pi$
0	0	1	4	- y
0	1	0	8	$\pi - y$
0	1	1	1	y
1	0	0	6	$-\pi/2$ - y
1	0	1	7	$-\pi/2$ + y
1	1	0	3	$\pi/2$ + y
1	1	1	2	$\pi/2$ - y

Com os valores de x será feita uma comparação dos valores acessando a tabela e conseqüentemente, será determinado o ângulo desejado com a correção adequada. A figura a seguir mostra o algoritmo para o cálculo do arco tangente e a seguinte o respectivo diagrama de blocos.

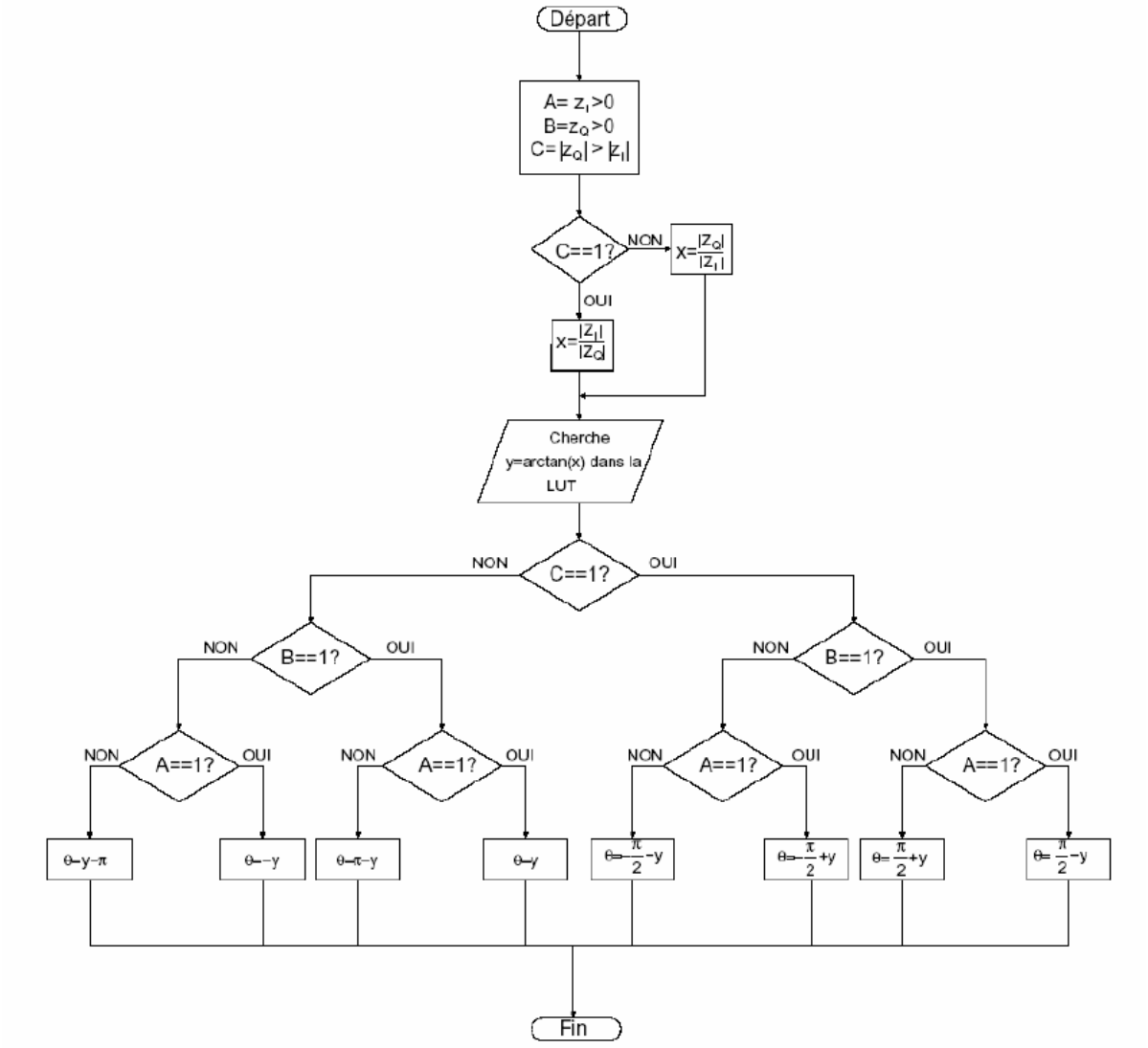

FIGURA 2.3 – Diagrama de flux do algoritmo para determinação do arco tangente

FONTE: **INPE – CRN**

.

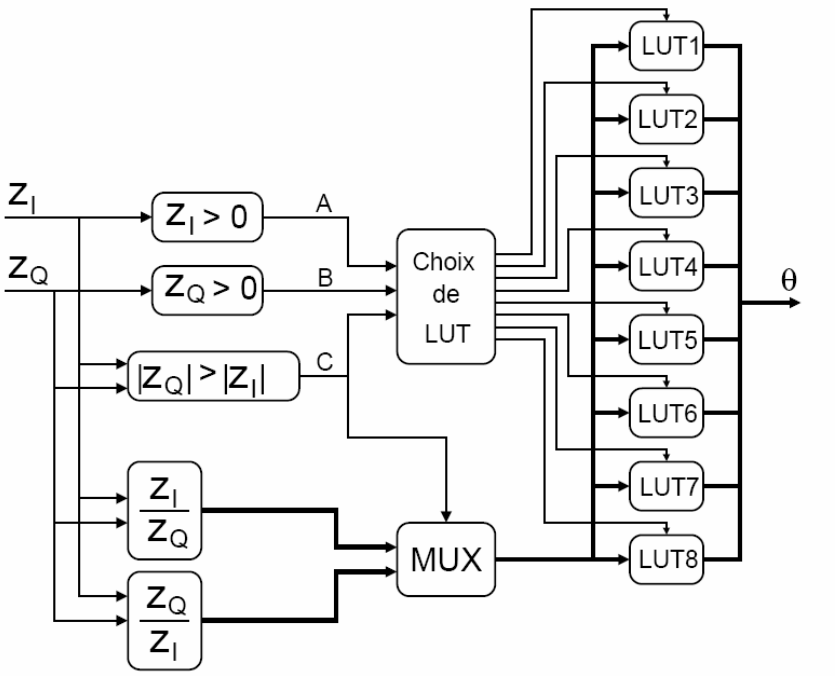

FIGURA 2.4 – Diagrama de bloco do algoritmo para determinação do arco tangente

FONTE: **INPE – CRN**

CAPÍTULO 3

CONCLUSÃO

O algoritmo do Arco Tangente ainda não foi concluído, estudos estão sendo feitos para, brevemente conseguirmos concluir essa etapa do projeto de Programação em VHDL de Circuitos Lógicos para Implementação em FPGA. Concluído essa fase teremos outros projetos para inserir essa Linguagem de Descrição de Hardware.

REFERÊNCIAS BIBLIOGRÁFICAS

Tocci, Ronald J., Widmer, Neal S., **Sistemas Digitais Princípios e Aplicações**, LTC, 2000

Roberto d' Amore, **Descrição e síntese de Circuitos Digitais**, LTC.

APÊNDICE A

ALGORITMOS DESENVOLVIDOS NO DECORRER DO ESTUDO DA LINGUAGEM

```
ENTITY std_a IS
 PORT (s1,s2,s3,s4,s5  : OUT BIT_VECTOR (3 DOWNTO 0));
END std_a;

ARCHITECTURE teste OF std_a IS
  CONSTANT c1  : BIT_VECTOR(3 DOWNTO 0) := "1011";  -- constante
BEGIN
  s1 <= c1;                                -- definindo atraves de constante
  s2 <= "1011";                            -- definindo valor bit a bit
  s3 <= B"1_0_11";                         -- binario default com separadores
  s4 <= X"B";                              -- hexadecimal
  s5 <= (3 =>'1', 2 =>'0', OTHERS =>'1'); -- uso da palavara reservada "others"
END teste;
```

```
ENTITY maq_est1 IS
  PORT (ck      :  IN     BIT;                    -- relogio borda subida
        rst     :  IN     BIT;                    -- rst=1, q=00
        q       :  BUFFER BIT_VECTOR (1 DOWNTO 0)); -- saida codigo Gray
END maq_est1;

ARCHITECTURE teste OF maq_est1 IS

BEGIN
  abc: PROCESS (ck, rst)
  BEGIN
    IF rst = '1' THEN                      -- estado inicial
      q <= "00";
    ELSIF (ck'EVENT and ck ='1') THEN      -- ciclo de estados
      CASE q IS
        WHEN "00" => q <= "01";
        WHEN "01" => q <= "11";
        WHEN "11" => q <= "10";
        WHEN "10" => q <= "00";
       END CASE;
    END IF;
  END PROCESS abc;
END teste;
```

```
 1 ENTITY maq_est2 IS
 2   PORT (ck      :  IN     BIT;                        -- relogio borda subida
 3         sobe    :  IN     BIT;                        -- sobe=1, q=00,01,11,10,00...
 4         rst     :  IN     BIT;                        -- rst=1, q=00
 5         q       :  OUT    BIT_VECTOR (1 DOWNTO 0)); -- saida codigo Gray
 6 END maq_est2;
 7
 8 ARCHITECTURE teste OF maq_est2 IS
 9   SIGNAL estado : INTEGER RANGE 3 DOWNTO 0;
10 BEGIN
11   abc: PROCESS (ck, rst)
12   BEGIN
13     IF rst = '1' THEN                        -- estado inicial
14       estado <= 0;
15     ELSIF (ck'EVENT and ck ='1') THEN      -- ciclo de estados
16       CASE estado IS
17         WHEN 0 =>                             -- s0
18           IF sobe = '1' THEN estado <= 1;   -- s1
19           ELSE               estado <= 3;   -- s3
20           END IF;
21         WHEN 1 =>                             -- s1
22           IF sobe = '1' THEN estado <= 2;   -- s2
23           ELSE               estado <= 0;   -- s0
24           END IF;
25         WHEN 2 =>                             -- s2
26           IF sobe = '1' THEN estado <= 3;   -- s3
27           ELSE               estado <= 1;   -- s1
28           END IF;
29         WHEN 3 =>                             -- s3
30           IF sobe = '1' THEN estado <= 0;   -- s0
31           ELSE               estado <= 2;   -- s2
32           END IF;
33       END CASE;
34     END IF;
35   END PROCESS abc;
36
37   WITH estado SELECT
38     q <= "00" WHEN 0,
39          "01" WHEN 1,
40          "11" WHEN 2,
41          "10" WHEN 3;
42 END teste;
```