UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
INE5406 - SISTEMAS DIGITAIS

PROJETO PRÁTICO DE SISTEMAS DIGITAIS:

SISTEMA PARA APROXIMAÇÃO DO VALOR DE PI ATRAVÉS DO MÉTODO DE MONTE CARLO

Área:
Sistemas Digitais

Lucas Zacchi de Medeiros

Florianópolis
Setembro, 2018

Lucas Zacchi de Medeiros

PROJETO PRÁTICO DE SISTEMAS DIGITAIS: SISTEMA PARA APROXIMAÇÃO DO VALOR DE PI ATRAVÉS DO MÉTODO DE MONTE CARLO

Trabalho da disciplina “INE540 - Sistemas Digitais” apresentado ao Curso de Ciências da Computação do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina

Professor: Rafael Luiz Cancian, Dr. Eng.

Florianópolis
06/Setembro/2018

Lista de Figuras

Sumário

1. Introdução

Este projeto prático de sistemas digitais visa desenvolver um sistema digital síncrono que realize o cálculo da aproximação do valor de π. Esse sistema digital será descrito em VHDL, sintetizado, simulado e prototipado em FPGA da Altera.

No sistema proposto, o usuário poderá especificar o número de iterações para o cálculo do valor de π, que será realizado utilizando o método de aproximação de Monte Carlo, que é feito gerando pontos aleatórios em uma área delimitada por um círculo inscrito em um quadrado. A razão entre os pontos dentro do círculo e os pontos fora dele geram uma aproximação do valor de π. Quando o cálculo termina, o resultado é apresentado.

2. Projeto do Sistema

O projeto do sistema digital inicia com a idenntificação das entradas e saídas e descrição e captura do comportamento do sistema, que é realizado nas seções seguintes

2.1 Identificação das Entradas e Saídas

O sistema digital proposto terá a interface apresentada na seguinte figura.

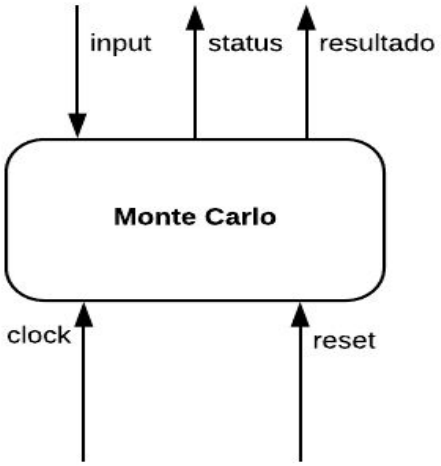

Figura 2.1: *Interface do sistema proposto. O sinal 'input' informa o número de iterações, o sinal 'status' informa em que etapa o sistema se encontra e o sinal 'resultado' apresenta o resultado obtido no final da computação. O sistema ainda contém o sinal 'clock' correspondente ao sinal de relógio de sincronismo, e o sinal de 'reset' assíncrono.*

2.2 Descrição e Captura do Comportamento

O funcionamento do sistema é descrito da seguinte maneira: Quando o sinal de 'reset' for ativado, o sistema fica no Estado Inicial, aguardando o recebimento de um comando de início. Após o sinal 'iniciar', o sistema passa a esperar os dados de entrada, que correspondem ao número de iterações realizadas no cálculo.

Quando a entrada é fornecida, o sistema realizará os cálculos, em uma série de estados. Inicialmente é definido um círculo de raio r=1 inscrito em um quadrado. A área deste círculo é dada por: $Ac = \pi r^2 = \pi 1^2 = \pi$, e a área do quadrado é $Aq = (2r)^2 = 4$.

A razão entre a área do círculo e a área do quadrado é dada por:

$$p = \frac{Ac}{Aq} = \frac{\pi}{4} = 0.7853981...$$

Uma vez calculada essa razão, basta multiplicar o resultado por 4 para obter o valor de π. Para isso, o sistema gera números pseudo-aleatórios que serão as coordenadas dos pontos *P(x,y)*. Após isso, o sistema verifica se esses pontos estão dentro do círculo, somando $x^2 + y^2$. Se o resultado for menor ou igual á 1, significa que o ponto está dentro do círculo. Com suficientes pontos gerados, o valor se aproxima cada vez mais de $\frac{\pi}{4}$. Ao término das iterações, o resultado final é multiplicado por 4 e é obtida uma aproximação do valor de π. Ao final do cálculo, o resultado é mostrado.

Essa função pode ser executada pelo seguinte algoritmo:

```
#include <iostream>

#define ITERATIONS 2147483647  // Maximum signed number representation in 32
bits


int main() {
   double x_coordinate, y_coordinate, origin_distance, pi;
   int inside_circle, inside_square = 0;

   srand(time(NULL));  // Seeding the pseudo-number generator.

   for (int i = 0; i < ITERATIONS; ++i) {
       x = double(rand() % ITERATIONS) / ITERATIONS;
       y = double(rand() % ITERATIONS) / ITERATIONS;

       origin_distance = x * x + y * y;

       if (origin_distance <= 1) {
           ++inside_circle;
       }
       ++inside_square;

       pi = double(4 * inside_circle)/inside_square;
   }

   std::cout << "\nFinal estimation of pi = " << pi;
   return 0;
}
```

Algoritmo 2.1: *Algoritmo escrito em C++ descrevendo o método de Monte Carlo para aproximação do valor de* π

Analisando o algoritmo, é possível extrair os elementos e componentes necessários para o projeto do sistema. Podemos observar que é necessário haver registradores para armazenar dados de variáveis internas na memória. Podemos verificar também as operações

lógicas e matemáticas para a execução do sistema. A seguir estão listados os componentes identificados com a análise do algoritmo:

I. **Registradores.** A necessidade da existência de registradores pode ser observada pela presença de variáveis internas. No sistema, sertão necessários os registradores:
   A. **i;**
   B. **inside_circle:** 32 bits, inteiro sem sinal;
   C. **inside_square:** 32 bits, inteiro sem sinal;
   D. **ITERATIONS:** 32 bits, inteiro sem sinal;
   E. **origin_distance:** 32 bits, ponto flutuante;
   F. **pi:** 32 bits, ponto flutuante;
   G. **x:** 32 bits, ponto flutuante;
   H. **y:** 32 bits, ponto flutuante;

II. **Operações.** As operações lógicas e aritméticas do sistema podem ser extraídas através de análise do funcionamento do algoritmo. Com base nisso, podemos verificar a necessidade das seguintes operações:
   A. **Atribuição:** Os registradores terão valores atribuídos durante toda a execução do sistema. Portanto, é necessário controlar a carga dos mesmos.
   B. **Comparação entre números sem sinal:** O algoritmo executa uma comparação em dois momentos diferentes. A primeira vez para executar o laço de repetição, e a segunda para analisar o valor de verdade do condicional *if*. Linhas 13 e 19.
   C. **Adição inteira;**
   D. **Adição em ponto flutuante;**
   E. **Multiplicação em ponto flutuante;**
   F. **Divisão em ponto flutuante;**

Após extrair os componentes do algoritmo, é possível capturar o comportamento do sistema em uma Máquina de Estados de Alto Nível (FMSD), que está descrita a seguir.

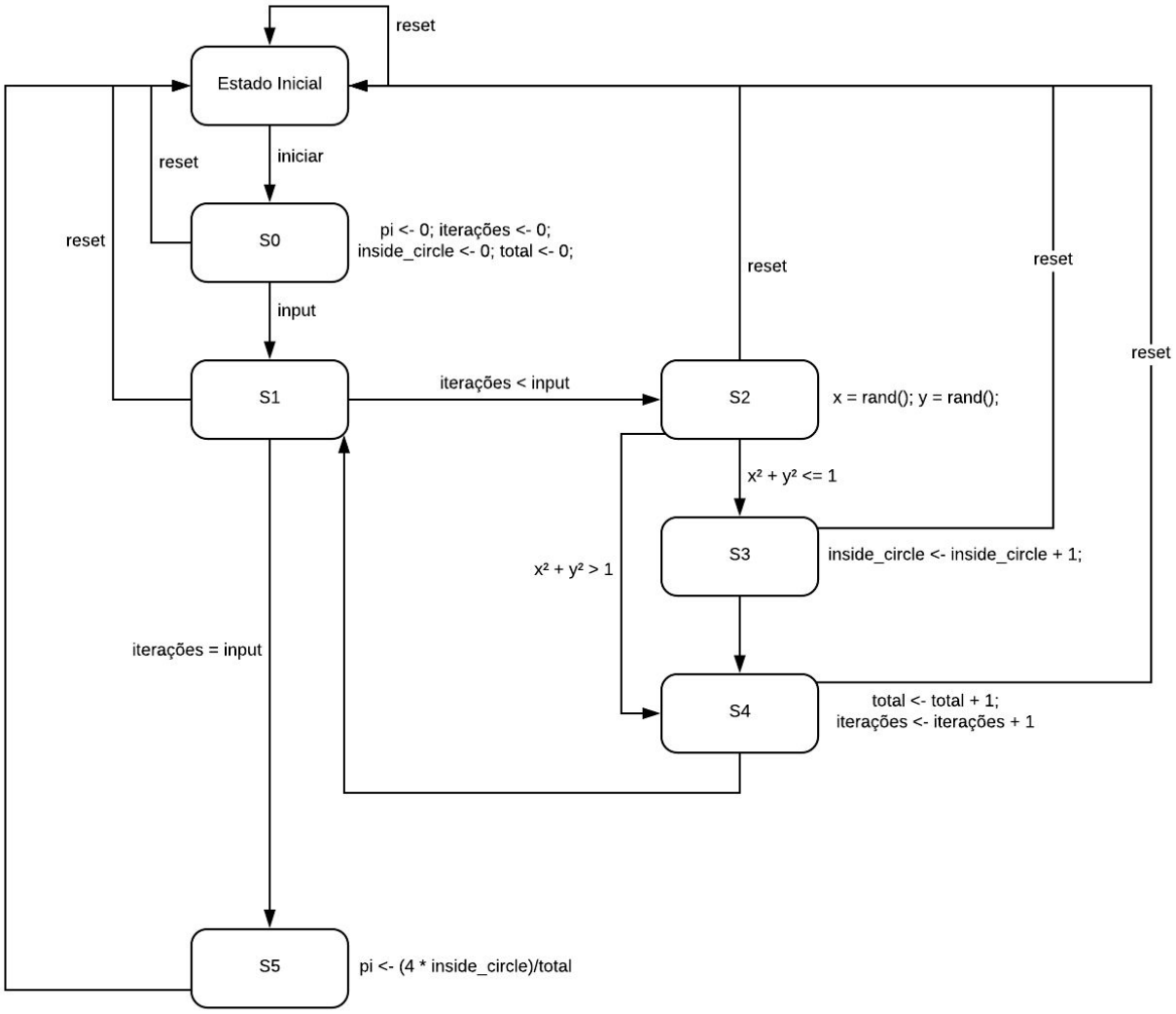

Figura 2.2: *FSMD do sistema digital. O comportamento descrito no algoritmo 2.1, aqui representado graficamente.*

2.3 Projeto do Bloco Operativo

A partir dos dados obtidos na captura do comportamento do sistema, podemos iniciar o projeto do Bloco Operativo. Abaixo, estão descritos os circuitos necessários para as atribuições dos registradores do sistema. Nas figuras, os sinais de controle são representados em azul:

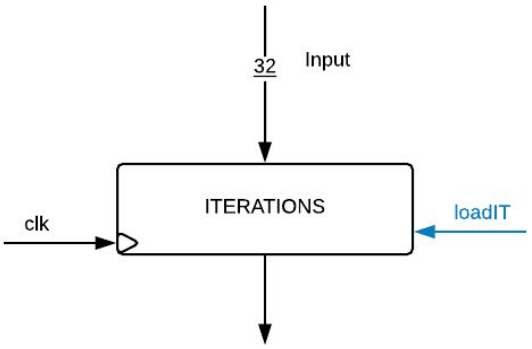

Figura 2.3: *Atribuição do registrador ITERATIONS*

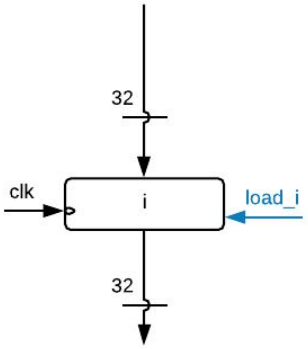

Figura 2.4: *Atribuição do registrador i*

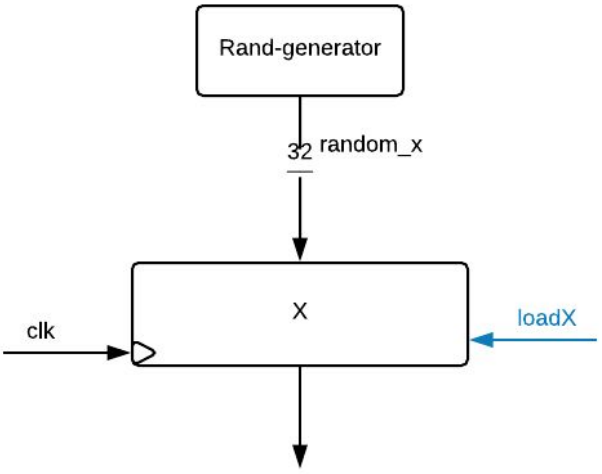

Figura 2.5: *Atribuição do registrador X*

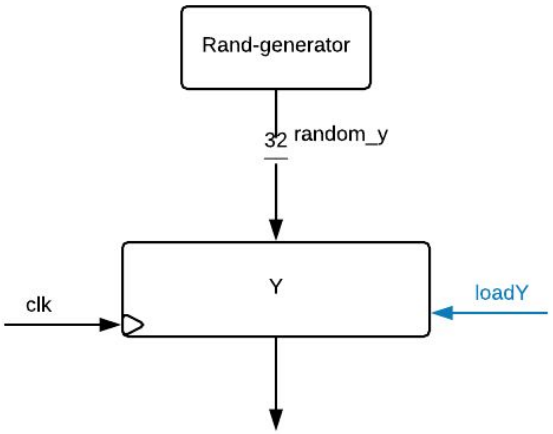

Figura 2.6: *Atribuição do registrador Y*

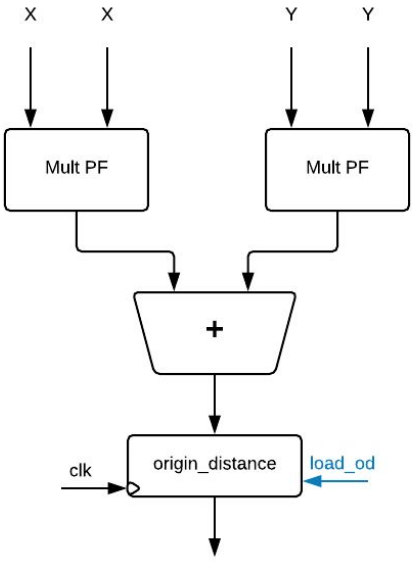

Figura 2.7: *Atribuição do registrador origin_distance*

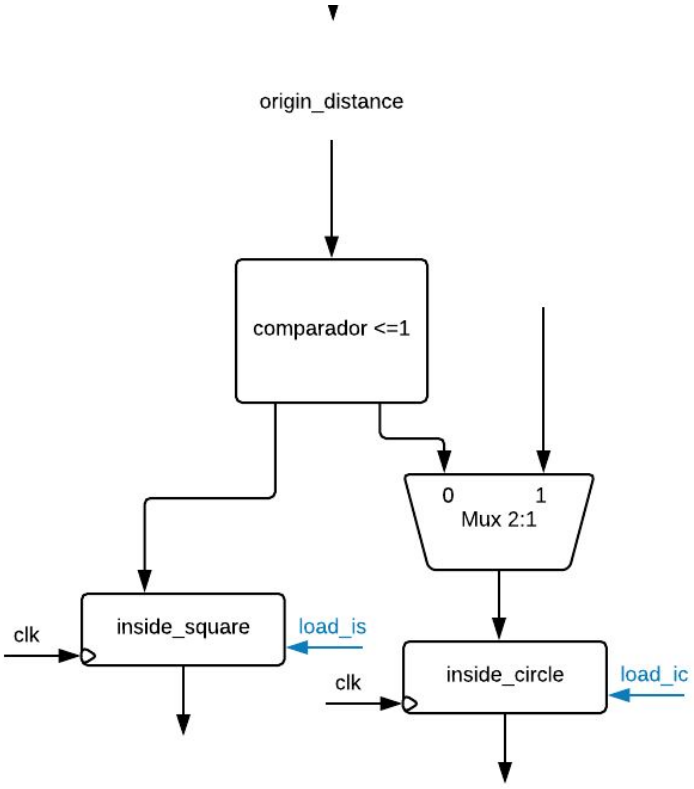

Figura 2.8: *Atribuição dos registradores inside_square e inside_circle*

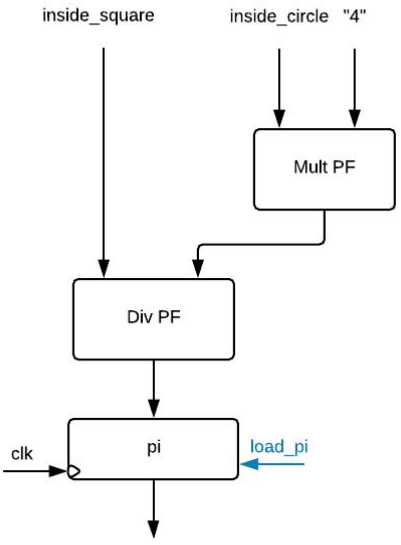

Figura 2.9: *Atribuição do registrador pi*

2.3 Projeto do Bloco de Controle

Com base no projeto do bloco operativo, podemos determinar os sinais de controle (originados no Bloco de Controle) e os sinais de status (enviados aos blocos de controle). Os sinais de controle são:

1. `loadi:` Responsável pela carga do registrador i;
2. `loadX:` Responsável pela carga do registrador X;
3. `loadY:` Responsável pela carga do registrador Y;
4. `load od:` Responsável pela carga do registrador origin_distance;
5. `load is:` Responsável pela carga do registrador inside_square;
6. `load ic:` Responsável pela carga do registrador inside_circle;
7. `load pi:` Responsável pela carga do registrador pi;

São sinais de status:

1. `i > IT:` Indica que o registrador i armazena um valor maior que IT
2. `od >` 1: Indica que o registrador origin_distance armazena um valor maior que 1
3. `done_power:` Indica que a operação de potenciação foi finalizada com sucesso
4. `done_division:` indica que a operação de divisão foi realizada com sucesso

Com os sinais estabelecidos e listados, podemos projetar uma possível arquitetura para o sistema. Na figura abaixo, vemos o diagrama contendo o Bloco de Controle, o Bloco Operativo e as trocas de sinais entre eles.

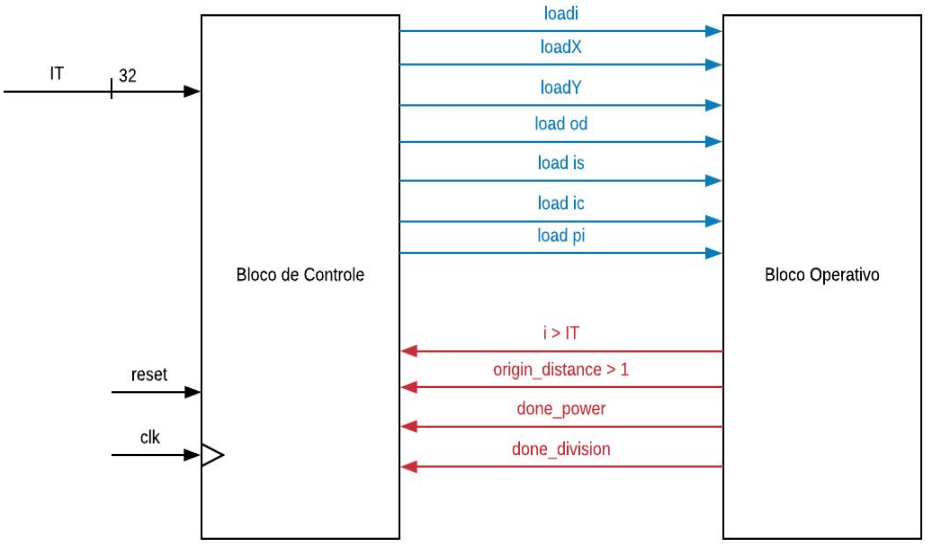

Figura 2.10: *Diagrama de interação entre Bloco de Controle e Bloco operativo.*