Universidade Federal do Rio Grande do Norte — UFRN

Instituto Metrópole Digital — IMD

**Disciplina:** Arquitetura de Computadores — DIM 0127

**Turma:** T01 24M34 (2019.1)

**Componentes:** Alexandre Alves Andrade

Francisco de Assis Campos Júnior

Kaio Henrique de Sousa

**Relatório projeto PO-PC:**

O presente projeto consiste em um modelo de circuito sequencial dedicado, que tem como objetivo calcular a raiz quadrada de um número podendo esse número ser entre 0 e 65536. O circuito implementa um método bem simples para calcular a raiz quadrada, o método das iterações, que permite uma grande aproximação da raiz quadrada real.

O circuito tem em sua parte operativa (PO) como principais componentes cinco registradores de dezesseis bits, para armazenar as variáveis, sendo endereçados de zero até quatro, também cinco multiplexadores e uma ULA com as operações aritméticas básicas (adição, multiplicação, subtração e divisão) e uma operação lógica (or) além de operadores de comparação (“igualdade” e “maior que”). Além disso ele tem em sua parte de controle (PC) com a função de transferência de estado e uma memória de 4 bits para armazenar os estados.

O primeiro passo para elaborar o projeto foi a construção do algoritmo (segue em anexo os códigos fonte), após isso elaboramos tabela a seguir que apresenta de transferência de estados, ela representa cada estado a partir de seu anterior a notação está em binária e decimal, e segue do zero a treze.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variável de Controle** | **Estado atual(EA)** | | **Próximo Estado (PE)** | |
| **Binário** | **Decimal** | **Binário** | **Decimal** |
| **0** | 0000 | 0 | 1101 | 0 |
| **1** | 0000 | 1 | 0001 | 1 |
| **x** | 0010 | 2 | 0011 | 2 |
| **x** | 0011 | 3 | 0100 | 3 |
| **x** | 0100 | 4 | 0101 | 4 |
| **x** | 0101 | 5 | 0110 | 5 |
| **x** | 0110 | 6 | 0111 | 6 |
| **x** | 0111 | 7 | 1000 | 7 |
| **x** | 1000 | 8 | 1001 | 8 |
| **x** | 1001 | 9 | 1010 | 9 |
| **0** | 1010 | 10 | 1100 | 10 |
| **1** | 1010 | 11 | 1011 | 11 |
| **x** | 1011 | 12 | 0000 | 12 |
| **x** | 1100 | 13 | 0000 | 13 |

Figura 1.1: Tabela de transferência de estados.

O diagrama abaixo tem o objetivo de ser mais claro apresentando o fluxo dos estados.

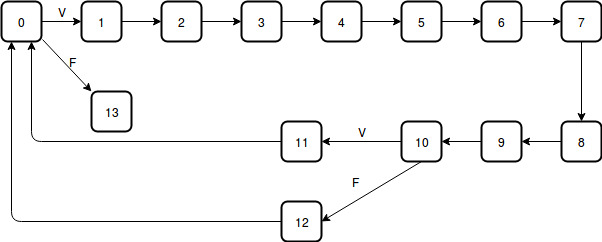


Figura 1.2: Diagrama de transferência de estados, imagem autoexplicativa.

A tabela a seguir apresenta os comandos da parte operativa em cada estado, iremos explicar cada um deles a seguir.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EA** | **M1** | **M2** | **M3** | **M4** | **M5** | **M6** | **ld\_1** | **ld\_2** | **ld\_3** | **ld\_4** | **ld\_5** | **ld\_6** | **ULA** |
| **0** | x | x | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 000 |
| **1** | x | 1 | 0 | x | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 010 |
| **2** | x | 0 | 1 | x | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 011 |
| **3** | x | 1 | 0 | x | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 001 |
| **4** | 0 | x | 1 | x | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 100 |
| **5** | x | 1 | 1 | x | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 001 |
| **6** | 1 | x | x | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 010 |
| **7** | 0 | x | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 011 |
| **8** | x | x | 1 | x | x | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 101 |
| **9** | x | 1 | x | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 101 |
| **10** | 0 | x | 1 | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 110 |
| **11** | x | x | x | 1 | x | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 111 |
| **12** | x | x | x | x | x | x | 0 | 1 | 0 | 0 | 0 | 0 | 111 |
| **13** | 1 | x | x | x | 0 | x | 0 | 0 | 0 | 0 | 0 | 0 | 111 |

Figura 1.3: tabela de PC.

A seguir apresentamos uma breve descrição do funcionamento do circuito, dado que os nomes das variáveis são apenas parte do projeto para um melhor entendimento do mesmo, apresentaremos o funcionamento com o nome que as variáveis tinham em seu algoritmo inicial, com a exceção da variável auxiliar, que foi inserida para poder ser viabilizada a construção do circuito.

No estado 0, ele verifica se a variável *sair* é igual a zero, e após verificar ela nega isso, caso que a saída seja verdadeira ela segue para o estado treze, caso contrário segue o estado 1.

Já no estado um ela utiliza dos multiplexadores para selecionar a variável *r* duas vezes e multiplicar uma pela outra, armazenando o resultado na variável *auxiliar*. Em seguida, no estado 2 são selecionadas as variáveis *x* e a *auxiliar*, e é feita a subtração delas armazenando novamente o resultado na *auxiliar*.

Em seguida, no estado 3 é realizada a soma de *x* com ele mesmo, e o resultado é armazenado na variável *rp*. No estado 4 a ULA realiza a divisão do valor armazenado em *auxiliar* pelo valor armazenado em *rp* e armazena novamente esse valor em *auxiliar*. Já no estado 5 o circuito soma os valores das variáveis *r* e *auxiliar* e o atribui a variável *res*, em seguida, no estado 6, a ULA multiplica a variável *res* por ela mesmae atribui o resultado a variável *rp*. Após isso no estado 7 ele executa a subtração do valor de *rp* menos *r* e atribui o valor a *auxiliar*.

No estado subsequente, o 8, a ULA verifica se o valor contido em *auxiliar* é igual a 0, e atribui o resultado a variável *auxiliar* novamente. Já no nono estado, a ULA compara se os valores das variáveis *r* e *rp* são iguais e armazena o resultado da comparação na variável *rp*. No décimo estado ela compara com um “or” as variáveis *auxiliar* e *rp*. Dependendo do resultado da operação do estado anterior o estado seguinte pode ser o onze, caso a saída seja verdadeira, ou 12 caso a saída seja falsa.

No estado 11 ela atribui o valor de verdadeiro a variável *sair* enquanto no 12 ele atribui o valor da variável *res* a variável *r*. E no décimo terceiro e último estado, o circuito simplesmente tem sua saída igual ao valor de res, podendo ser projetada para ela ser mostrada em um monitor por exemplo.

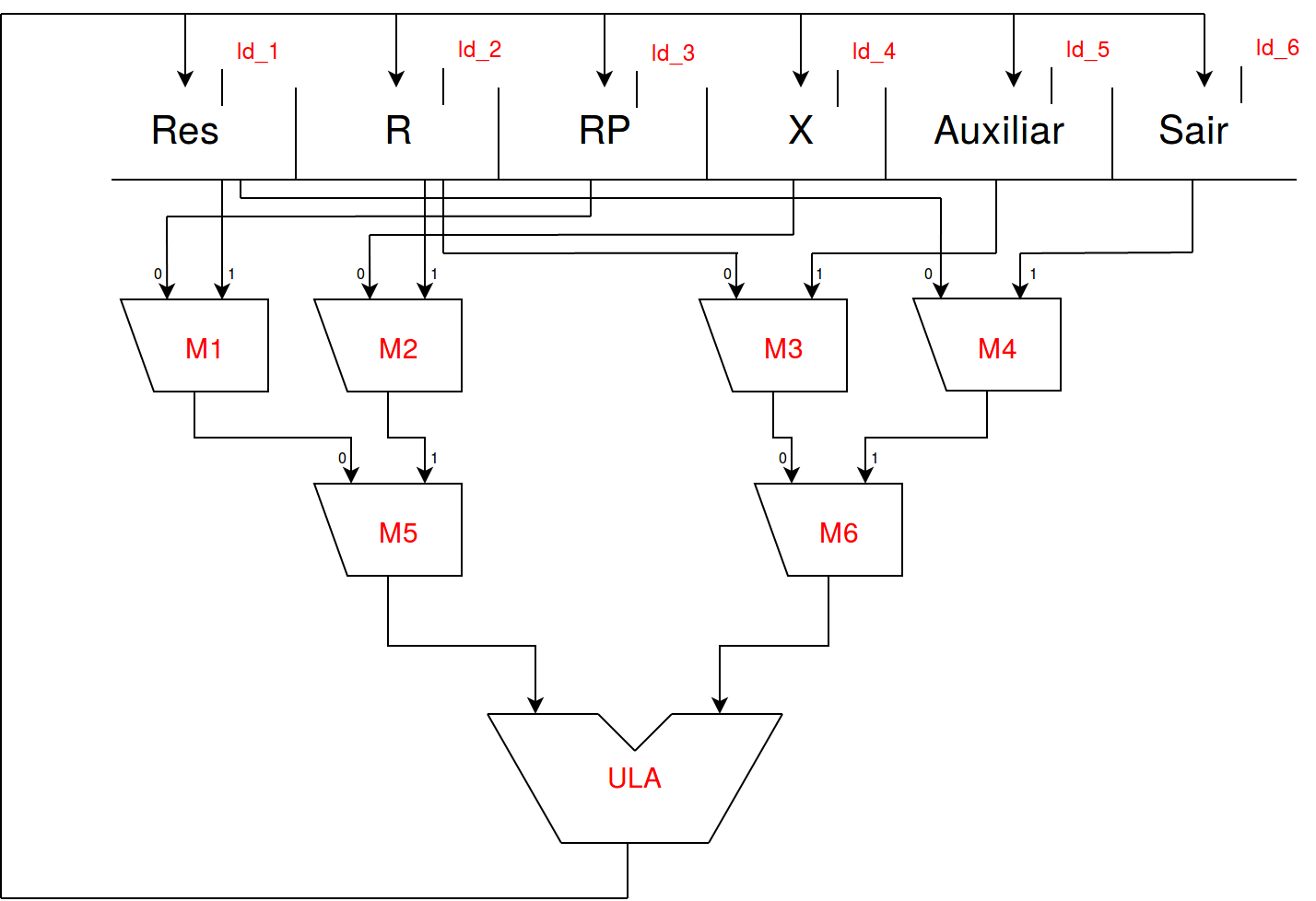


Figura 1.4: esquema do circuito.

OPERAÇÕES DA ULA

|  |  |
| --- | --- |
| 000 | Verifica se a entrada A da ULA é igual a zero. |
| 001 | Faz a soma da entrada A com a entrada B. |
| 010 | Faz a multiplicação da entrada A com a entrada B. |
| 011 | Faz a subtração da entrada A com a entrada B da ULA. |
| 100 | Faz a divisão da entrada A pela entrada B da ULA. |
| 101 | Verifica se a entrada A da ULA é igual a zero. |
| 110 | Verifica se a entrada A ou a B são diferentes de zero. |
| 111 | Insere na saída da ULA a valor na entrada A. |