ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PCS 3225 – SISTEMAS DIGITAIS II

ATIVIDADE FORMATIVA 1 - MULTIPLICADOR BINÁRIO MODIFICADO

Professor:
Dr. Marco Tulio Carvalho de Andrade

Grupo 10	**NUSP**
Guilherme Fortunato Miranda	13683786
João Pedro Dionizio Calazans	13673086
Lucas Pates Rodrigues	11545304
Matheus Tierro de Paula	11804021
Pedro Kooshi Ito Sartori	13683661

SÃO PAULO
2023

SUMÁRIO

1. APRESENTAÇÃO E ESTUDO DO ALGORITMO

O algoritmo sob análise representa um multiplicador binário baseado em somas e deslocamentos. A figura 1 apresenta o algoritmo.

Figura 1: Algoritmo do multiplicador binário

Algorithm 1 Multiplicação com somas e deslocamentos

```
1: procedure MULTIPLICADOR MODIFICADO
2:     armazenar multiplicando de n bits
3:     armazenar multiplicador de n bits
4:     iniciar produto parcial de 2n bits para ZERO
5:     // ▷ processar um bit do multiplicador de cada vez, começando pelo bit menos significativo
6:     while não terminou processamento de todos os bits do multiplicador do
7:         if bit do multiplicador = 1 then
8:             produto parcial ← multiplicando + produto parcial
9:             // ▷ soma realizada nos n bits mais significativos do produto parcial
10:            armazenar o bit de vai-um do somador em um flip-flop
11:            deslocar o conjunto do flip-flop e produto parcial para a direita
12:        else
13:            somente deslocar o conjunto do flip-flop e produto parcial para a direita
14:        passa para próximo bit do multiplicador
15:    // ▷ fim da multiplicação
16:    Resultado ← produto parcial
```

Fonte: MIDORIKAWA, Edson – PCS3225, 2023.

Esse algoritmo funciona baseado na propriedade binária de multiplicação, na qual a multiplicação por 2 é equivalente a um deslocamento para a esquerda (com a inserção de um zero no digito menos significativo), enquanto a multiplicação por 1 equivale ao mesmo número.

Dessa forma a cada iteração do while, verifica-se se o bit do multiplicador é 1 ou 0. Quando este for 0, equivale a multiplicar por 1, ou seja, o resultado parcial se mantém, apenas deslocando para a direita (concatenação de um zero no digito mais significativo) a fim de acomodar um bit a mais, já que na próxima iteração o bit analisado corresponderá a uma potência de 2 acima da iteração anterior.

Por outro lado, quando o bit analisado do multiplicador for 1, equivale a uma multiplicação por 2^k do multiplicando, sendo “k” a posição do bit em análise partindo-se da posição 0. Isso equivale a “k” deslocamentos para a esquerda, ou a inserção de “k” zeros a direita do multiplicando. Assim, soma-se esse valor ao resultado parcial e desloca-se para a direita a fim de acomodar um bit a mais.

É interessante perceber que o algoritmo utiliza o fato de que somar o multiplicando aos n bits mais significativos do produto parcial e manter os demais bits do produto parcial em suas posições, é equivalente ao processo explicado no parágrafo anterior.

Percebe-se que esse algoritmo é otimizado em relação ao algoritmo de multiplicador que utiliza somas sucessivas, uma vez que somas são realizadas apenas quando o bit analisado do multiplicador é igual a 1.

2. DEFINIÇÃO DOS CASOS DE TESTE

- Multiplicando = 0011, Multiplicador = 0110, Resultado = 00010010
- Multiplicando = 1111, Multiplicador = 1111, Resultado = 11100001
- Multiplicando = 0000, Multiplicador = 0000, Resultado = 00000000
- Multiplicando = 0001, Multiplicador = 1011, Resultado = 00001011
- Multiplicando = 0100, Multiplicador = 0100, Resultado = 00010000
- Multiplicando = 1001, Multiplicador = 0000, Resultado = 00000000

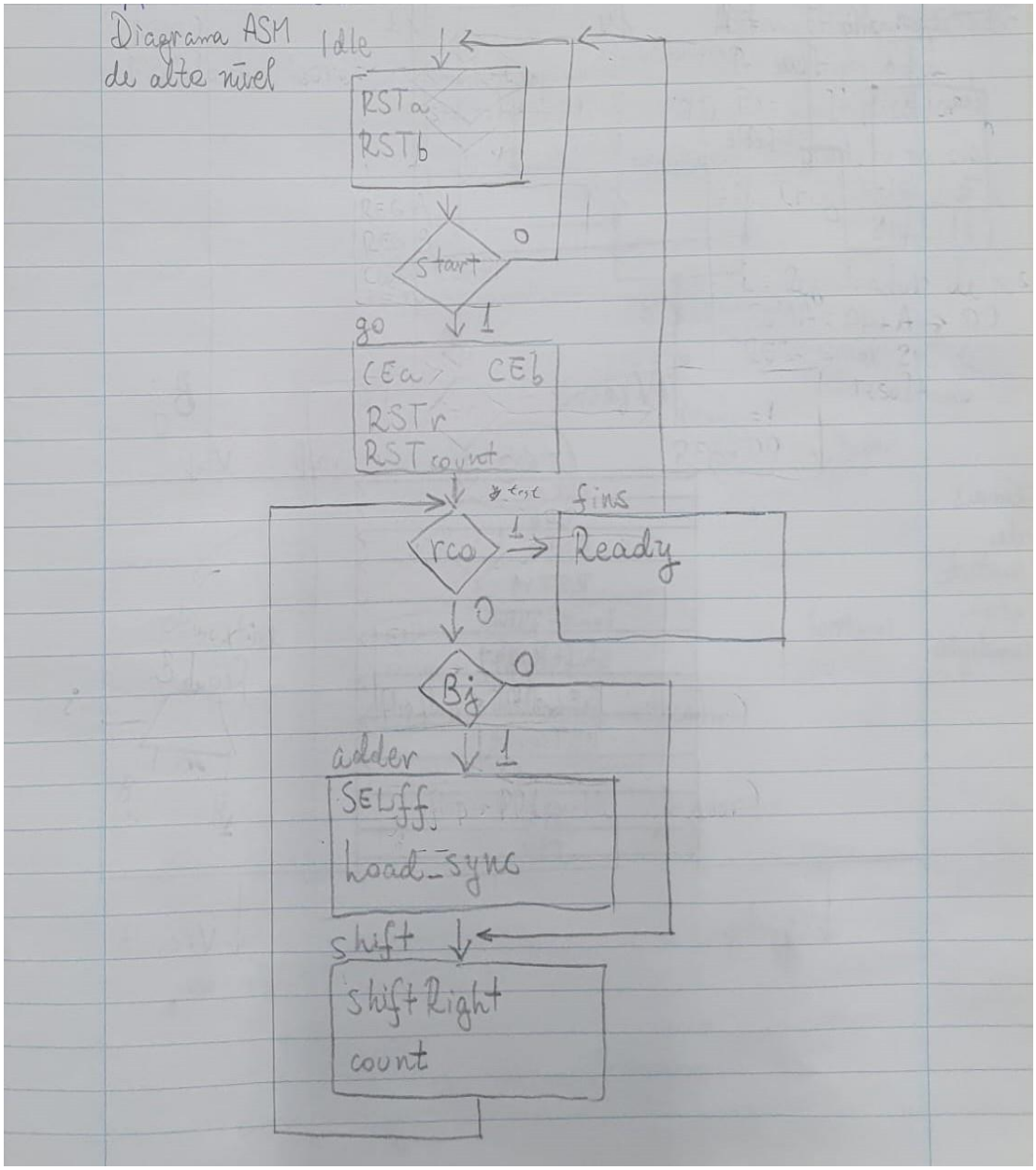
3. PSEUDOCÓDIGO

```
Multiplicador(Va,Vb) {
While(start = 0)
   ready = 0;
ra = Va                          // bit_vector(3 downto 0)
rb = Vb                                  // bit_vector(3 downto 0)
result = “00000000”
for(k=0; k < 4; k++)
   if rb(k) = 1
      intermediário = result (7 downto 4)
      FFD = carry out de (ra + intermediário) // bit do flip flop
      result = FFD & (ra + intermediário) & result (3 downto 1) //concatena e desloca para
direita
   else // rb(k) = 0
      result = ‘0’ & result(7 downto 1) // desloca para direita
ready = 1;
return result; }
```

4. DIAGRAMA ASM DE ALTO NÍVEL

Como pode ser visto na figura 2, os estados definidos na UC são: idle, go, adder, shift, fins.

Figura 2: ASM alto nível

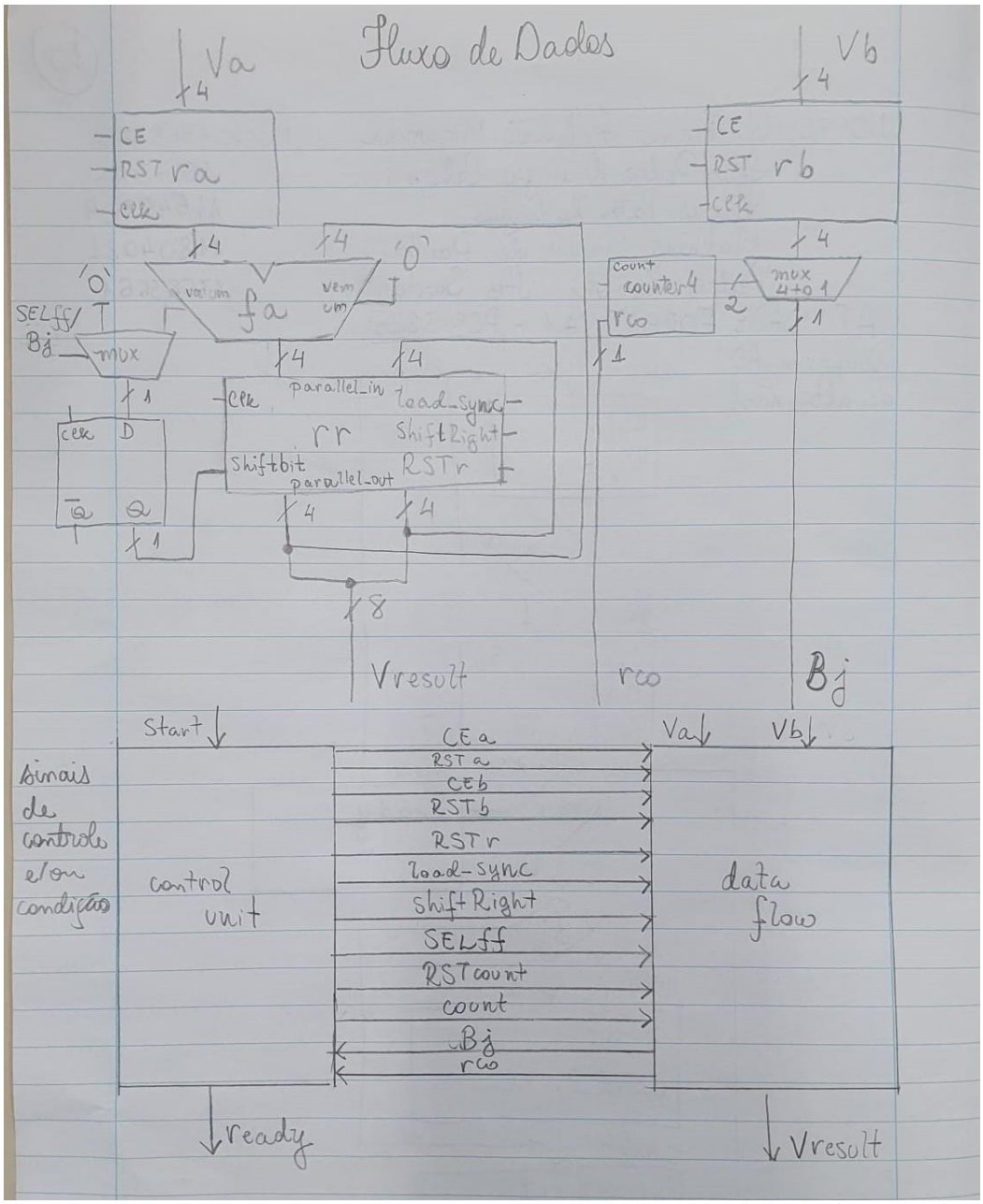


Fonte: De autoria própria.

5. FLUXO DE DADOS

Os componentes utilizados no fluxo de dados foram: dois registradores de 4 bits, um contador de 2 bits, um somador de 4 bits, um registrador de deslocamento de 8 bits, um flip-flop D, um MUX 2x1, e um MUX 4x1. A figura 3 ilustra as conexões estabelecidas, bem como os sinais de controle e de condição entre a unidade de controle e o fluxo de dados.

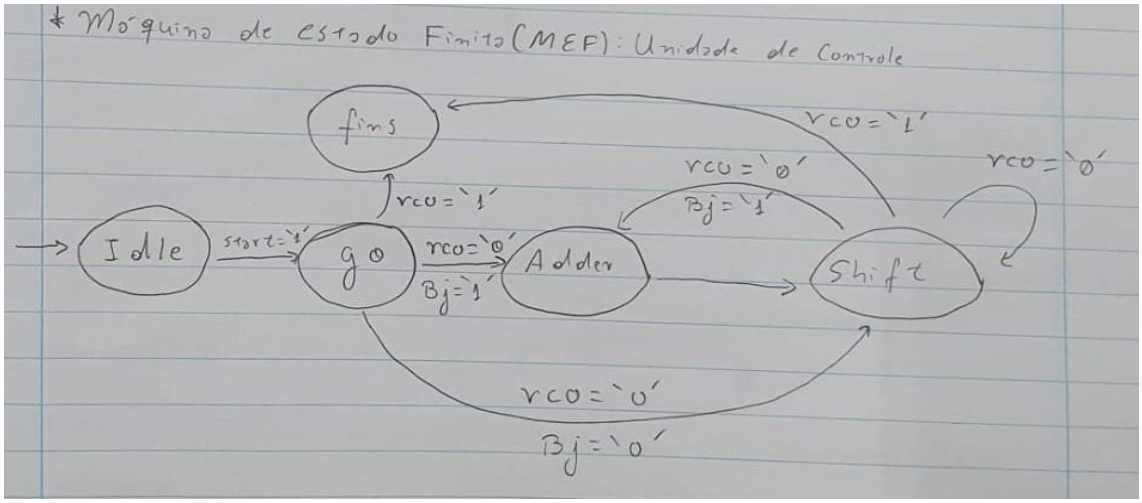
Figura 3: Fluxo de dados e sinais de controle/condição



Fonte: De autoria própria.

6. MÁQUINA DE ESTADOS DA UNIDADE DE CONTROLE

Figura 4: MEF da UC



Fonte: De autoria própria.

Observações não ilustradas na figura 4: para start = ‘0’, o estado é idle. A saída ready é igual a 1 apenas no estado fins. Se reset = ‘1’, o estado é idle.

7. CIRCUITO EM VHDL

Enviado em arquivo ZIP

8. TESTBENCH EM VHDL

Enviado em arquivo ZIP

9. SIMULAÇÃO DOS CASOS DE TESTE

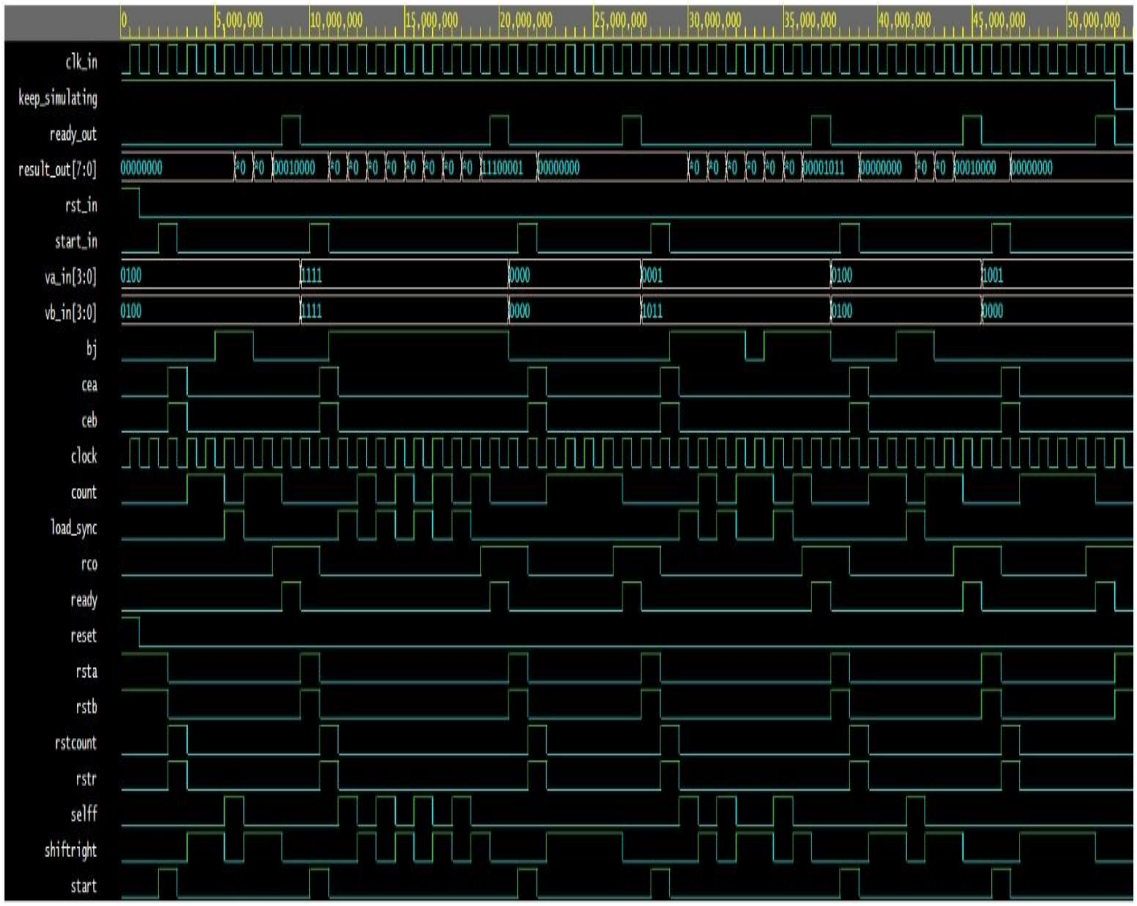
As figuras 5 e 6 são os resultados das simulações para os casos de testes listados no item 2.

Figura 5: Testes

```
[2023-08-11 22:14:42 UTC] ghdl -i '--std=08' design.vhd testbench.vhd  && ghdl -m '--std=08'  multiplicador_tb && ghdl -r '--std=08'  multiplicador_tb
analyze testbench.vhd
analyze design.vhd
elaborate multiplicador_tb
testbench.vhd:40:3:@0ms:(assertion note): simulation start
testbench.vhd:65:5:@0ms:(assertion note): simulation start
testbench.vhd:83:5:@8500ps:(assertion note): 1.OK: 3x6=00010010 (18)
testbench.vhd:100:5:@19500ps:(assertion note): 2.OK: 15x15=11100001 (225)
testbench.vhd:117:5:@26500ps:(assertion note): 3.OK: 0x0=00000000 (0)
testbench.vhd:134:5:@36500ps:(assertion note): 4.OK: 1x11=00001011 (11)
testbench.vhd:151:5:@44500ps:(assertion note): 5.OK: 4x4=00010000 (16)
testbench.vhd:168:5:@51500ps:(assertion note): 6.OK: 9x0=00000000 (0)
testbench.vhd:177:5:@52500ps:(assertion note): simulation end
Done
```

Fonte: De autoria própria via EDA Playground.

Figura 6: Sinais dos testes



Fonte: De autoria própria via EDA Playground.

10. CONSIDERAÇÕES FINAIS

O grupo considerou que a atividade formativa abrangeu bem o tema de metodologia de projeto de sistemas digitais lecionados em aula, posto que foi necessário dividir o projeto em várias etapas, desde pseudo-código e diagramas que demonstrassem o fluxo de dados e a unidade de controle, até o código final e *testbench* em VHDL.

A respeito do circuito do multiplicador modificado, destaca-se a associação de um contador de 2 bits com um multiplexador 4x1, a fim de analisar o bit do multiplicador a ser processado.

Ademais, a associação do somador com o registrador de deslocamento também se mostrou um desafio, uma vez que não é trivial a retroalimentação dos 4 últimos bits do registrador de deslocamento, além de também alimentar o somador com os 4 bits mais significativos do registrador de deslocamento.