Circuito Multiplicador em Verilog

Thiago Cordeiro de Melo[1]

[1]Escola Superior de Tecnologia - Universidade do Estado do Amazonas (UEA)
Amazonas – AM – Brazil

tcdm.eng23@uea.edu.br

Abstract. *This report presents the design and implementation of an 8-bit digital multiplier capable of producing a 16-bit result. The system was modeled using digital logic and is suitable for hardware implementation on platforms such as FPGAs. The multiplier accepts two 8-bit operands and performs signed multiplication in two's complement, yielding a 16-bit product. Beyond the basic arithmetic operation, the design also addresses key aspects such as synchronization, latency, and logic resource usage. Functional validation was conducted through simulation with a variety of input combinations, ensuring correct and reliable operation. This component is a critical building block in arithmetic logic units and embedded systems requiring efficient integer data processing.*

Resumo. *Este relatório apresenta o desenvolvimento e a implementação de um multiplicador digital de 8 bits, capaz de fornecer um resultado de 16 bits. O projeto foi modelado utilizando lógica digital e pode ser implementado em hardware com tecnologias como FPGAs. O multiplicador recebe dois operandos de 8 bits como entrada e realiza a operação aritmética de multiplicação em complemento de dois, produzindo um resultado de 16 bits. O sistema não apenas lida com a operação básica, mas também considera aspectos relevantes como sincronização, latência e consumo de recursos lógicos. A implementação foi validada por meio de simulações com diferentes combinações de entrada, garantindo a correção funcional. Este componente é fundamental em unidades aritméticas e sistemas embarcados que exigem processamento eficiente de dados inteiros.*

1. Introdução

A multiplicação é uma das operações fundamentais na computação digital, sendo amplamente utilizada em sistemas embarcados, processadores, unidades lógicas aritméticas (ALUs) e aplicações de processamento de sinais. Ao contrário da adição ou subtração, a multiplicação envolve um nível mais elevado de complexidade computacional, exigindo maior quantidade de recursos lógicos e controle sequencial, especialmente quando implementada em hardware.

Este relatório tem como foco o desenvolvimento e a análise de um multiplicador digital de 8 bits, cujo resultado é um produto de 16 bits. A escolha por operandos de 8 bits se justifica pela sua ampla utilização em microcontroladores e sistemas de arquitetura reduzida, enquanto o resultado de 16 bits é necessário para garantir a representação completa do produto, evitando perdas de informação.

O projeto foi idealizado considerando a multiplicação de números representados em complemento de dois, permitindo a manipulação de inteiros com sinal. A

implementação utiliza conceitos de lógica combinacional e pode ser sintetizada para dispositivos programáveis, como FPGAs ou ASICs, proporcionando alta eficiência em termos de velocidade e consumo de energia.

2. Funcionamento do Multiplicador

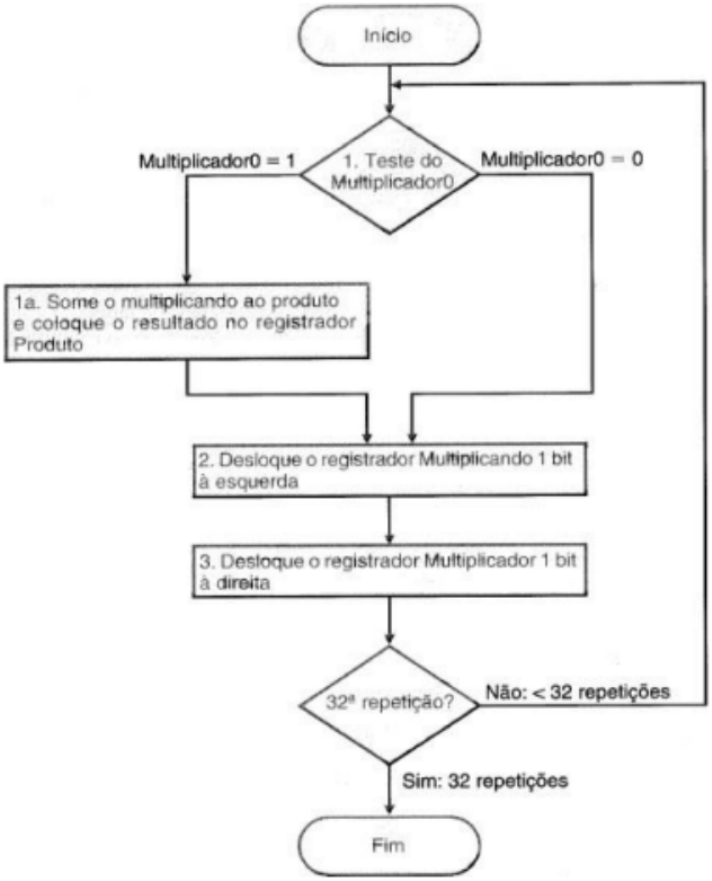

Figura 1. Diagrama do fluxo do Multiplicador

3. Funcionamento do Multiplicador

O multiplicador de 8 bits apresentado neste projeto baseia-se no algoritmo sequencial de multiplicação binária, que simula o processo manual de multiplicação, adaptado para circuitos digitais. O funcionamento do sistema pode ser descrito em ciclos, repetidos até que todos os bits do registrador multiplicador tenham sido processados. A cada ciclo, o bit menos significativo do multiplicador é analisado, determinando se o multiplicando será somado ao acumulador do produto.

O algoritmo segue as etapas abaixo:

1. **Inicialização:**
   - Os operandos (multiplicando e multiplicador) são carregados em registradores dedicados.
   - Um registrador adicional é reservado para armazenar o resultado (produto), inicialmente com valor zero.

2. **Teste do Bit Menos Significativo (Multiplicador$_0$):**
    - O sistema verifica se o bit menos significativo (LSB) do registrador do multiplicador é igual a 1.
    - Caso positivo, o multiplicando é somado ao valor atual do registrador de produto.
3. **Soma Condicional:**
    - Se o bit testado for 1, realiza-se a soma do registrador do produto com o valor do multiplicando.
    - O resultado é armazenado novamente no registrador de produto.
4. **Deslocamento dos Registradores:**
    - O registrador do *multiplicando* é deslocado 1 bit para a esquerda (equivalente a uma multiplicação por 2).
    - O registrador do *multiplicador* é deslocado 1 bit para a direita (equivalente a uma divisão por 2 e preparação do próximo LSB para teste).
5. **Verificação do Número de Ciclos:**
    - A operação é repetida por 8 ciclos (para operandos de 8 bits).
    - Ao final da oitava repetição, o processo é encerrado, e o registrador de produto contém o valor final da multiplicação (com até 16 bits de precisão).

Este método, além de eficiente em hardware, é fácil de implementar em sistemas baseados em registradores e controladores de estado finito. Ele permite realizar multiplicações sem a necessidade de blocos complexos de lógica combinacional, sendo ideal para aplicações em microcontroladores ou sistemas embarcados com recursos limitados.

4. Implementação em Verilog

O módulo `multi_8b` implementa um multiplicador de 8 bits.

Listing 1. Módulo ULA 8 Bits

```
module multiplicador (
    input clk,
    input rst,
    input start,
    input [7:0] A,
    input [7:0] B,
    output reg [15:0] Result,
    output reg fimOperacao,
    output reg Z,
    output reg OV
);
    reg [7:0] reg_A, reg_B;
    reg [15:0] acc;
    reg [3:0] count;
    reg execution;
    always @(posedge clk or posedge rst) begin
```

```
17          if (rst) begin
18              Result <= 16'b0;
19              reg_A <= 8'b0;
20              reg_B <= 8'b0;
21              acc <= 16'b0;
22              count <= 4'b0;
23              fimOperacao <= 1'b0;
24              execution <= 1'b0;
25              Z <= 1'b0;
26              OV <= 1'b0;
27          end
28          else if (start && !execution) begin
29              reg_A <= A;
30              reg_B <= B;
31              acc <= 16'b0;
32              count <= 4'b0;
33              fimOperacao <= 1'b0;
34              execution <= 1'b1;
35          end
36          else if (execution) begin
37              if (count < 8) begin
38                  if (reg_B[0]) begin
39                      acc <= acc + (reg_A << count);
40                  end
41                  reg_B <= reg_B >> 1;
42                  count <= count + 1;
43              end
44              else begin
45                  Result <= acc;
46                  fimOperacao <= 1'b1;
47                  Z <= (acc == 16'b0);
48                  OV <= |acc[15:8];
49                  execution <= 1'b0;
50              end
51          end
52      end
53
54  endmodule
```

- **Entradas:**
  - `clk`: Sinal de clock. Controla a sincronização das operações do multiplicador.
  - `rst`: Sinal de reset. Reinicializa todos os registradores e flags para o estado inicial.
  - `start`: Sinal de controle. Inicia a operação de multiplicação quando ativo.
  - `A [7:0]`: Primeiro operando de 8 bits (multiplicando).
  - `B [7:0]`: Segundo operando de 8 bits (multiplicador).
- **Saídas:**

  - `Result [15:0]`: Resultado da multiplicação, com 16 bits de largura para suportar o produto completo de dois números de 8 bits.
  - `fimOperacao`: Sinalizador que indica quando a operação de multiplicação foi concluída.
  - `Z`: Flag de zero. Ativa quando o resultado da multiplicação é zero.
  - `OV`: Flag de overflow. Ativa quando o valor resultante não pode ser representado em 8 bits (isto é, se algum bit entre `Result[15:8]` estiver ativo).
- **Flags de Status:**
  - **Zero (Z):**
    - Ativo quando o resultado é zero (todos os bits de `Result` são 0).
    - Calculado após o término da operação.
    - Útil para controle de fluxo ou validação de nulidade.
  - **Overflow (OV):**
    - Indica estouro de capacidade nos bits mais significativos do resultado.
    - É ativado se algum bit entre `Result[15:8]` estiver em 1.
    - Sinaliza que o produto não pode ser representado apenas com 8 bits.
- **Registradores e Variáveis Internas:**
  - `reg_A [7:0]`: Registrador que armazena o multiplicando durante a operação.
  - `reg_B [7:0]`: Registrador que armazena o multiplicador, sofrendo deslocamentos à direita a cada ciclo.
  - `acc [15:0]`: Acumulador do produto parcial durante os ciclos de multiplicação.
  - `count [3:0]`: Contador de ciclos, limitado a 8 para cobrir todos os bits do multiplicador.
  - `execution`: Sinal interno de controle, que indica se a operação está em andamento.

5. Testes do Multiplicador

Listing 2. Módulo para o test bench do multiplicador

```
1  `timescale 1ns/1ps
2
3  module tb_Multi_8b();
4
5      // Entradas
6      reg clk;
7      reg rst;
8      reg start;
9      reg [7:0] A;
10     reg [7:0] B;
11
12     // Saídas
13     wire [15:0] Result;
14     wire fimOperacao;
```

```
    wire Z;
    wire OV;

    // Instanciação do módulo a ser testado
    Multi_8b uut (
        .clk(clk),
        .rst(rst),
        .start(start),
        .A(A),
        .B(B),
        .Result(Result),
        .fimOperacao(fimOperacao),
        .Z(Z),
        .OV(OV)
    );

    // Clock: alterna a cada 5 ns
    always #5 clk = ~clk;

    initial begin
        // Inicialização
        clk = 0;
        rst = 1;
        start = 0;
        A = 8'd0;
        B = 8'd0;

        // Espera pelo reset
        #10;
        rst = 0;

        // Teste 1: 0 * 0
        A = 8'd0;
        B = 8'd0;
        start = 1;
        #10;
        start = 0;
        wait(fimOperacao);
        #10;

        // Teste 2: 5 * 10 = 50
        A = 8'd5;
        B = 8'd10;
        start = 1;
        #10;
        start = 0;
        wait(fimOperacao);
        #10;

```

```
64          // Teste 3: 255 * 255 = 65025
65          A = 8'd255;
66          B = 8'd255;
67          start = 1;
68          #10;
69          start = 0;
70          wait(fimOperacao);
71          #10;
72
73          // Teste 4: 128 * 2 = 256
74          A = 8'd128;
75          B = 8'd2;
76          start = 1;
77          #10;
78          start = 0;
79          wait(fimOperacao);
80          #10;
81
82          // Teste 5: 1 * 200
83          A = 8'd1;
84          B = 8'd200;
85          start = 1;
86          #10;
87          start = 0;
88          wait(fimOperacao);
89          #10;
90
91          // Finaliza simulação
92          $display("Testes finalizados.");
93          $stop;
94      end
95
96  endmodule
```

Explicação dos Testes no Testbench do Módulo `Multi_8b`

O testbench apresentado realiza uma série de cinco testes para verificar o funcionamento correto do multiplicador de 8 bits `Multi_8b`. Para cada teste, são definidos os valores dos operandos `A` e `B`, o sinal `start` é ativado para iniciar a multiplicação, e a simulação aguarda até a conclusão da operação indicada pelo sinal `fimOperacao`. A seguir, descrevemos cada teste:

- **Teste 1:** Multiplicação de 0×0
  - Entradas: $A = 0$, $B = 0$
  - Operação: 0×0
  - Resultado esperado: $Result = 0$
  - O objetivo deste teste é verificar se o multiplicador trata corretamente a multiplicação por zero.

- **Teste 2:** Multiplicação de 5×10
  - Entradas: $A = 5$, $B = 10$
  - Operação: 5×10
  - Resultado esperado: $Result = 50$
  - Este teste verifica uma multiplicação simples entre números pequenos para garantir o funcionamento básico.

- **Teste 3:** Multiplicação de 255×255
  - Entradas: $A = 255$, $B = 255$
  - Operação: 255×255
  - Resultado esperado: $Result = 65025$
  - Este teste verifica o comportamento do multiplicador no caso de valores máximos para 8 bits (255 é o maior número representável em 8 bits sem sinal).
  - Também é útil para avaliar o correto funcionamento dos sinais de overflow (`OV`) e zero (`Z`).

- **Teste 4:** Multiplicação de 128×2
  - Entradas: $A = 128$, $B = 2$
  - Operação: 128×2
  - Resultado esperado: $Result = 256$
  - Este teste avalia a multiplicação envolvendo um número alto e o valor 2, que deve resultar numa saída que ultrapassa o limite de 8 bits, mas está dentro do limite de 16 bits.

- **Teste 5:** Multiplicação de 1×200
  - Entradas: $A = 1$, $B = 200$
  - Operação: 1×200
  - Resultado esperado: $Result = 200$
  - Teste simples para verificar o comportamento com o multiplicador unitário.

Em todos os testes, o sinal `start` é ativado por 10 ns para iniciar a multiplicação, e a simulação aguarda o término indicado por `fimOperacao` antes de iniciar o próximo teste. Ao final, a simulação é parada com a mensagem “Testes finalizados.”.

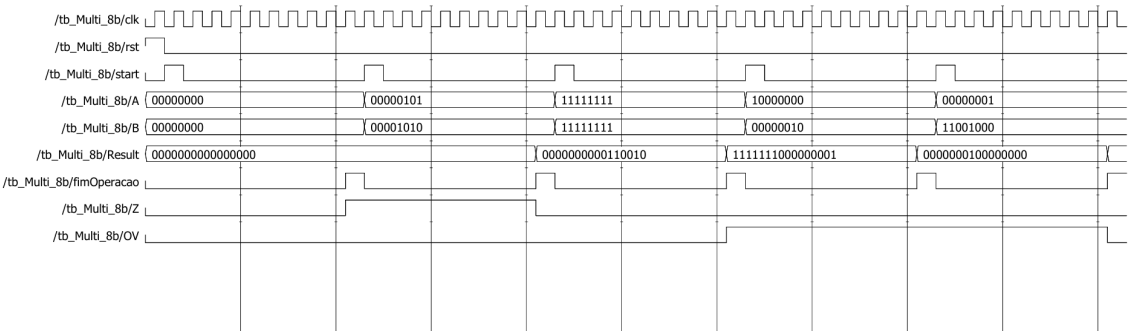

Figura 2. Verificação de cada um dos testes no ModelSim

6. Conclusão

Os testes realizados com o testbench do multiplicador de 8 bits permitiram verificar o funcionamento correto do módulo em diferentes cenários, abrangendo desde multiplicações triviais até casos extremos envolvendo operandos máximos.

Foram abordadas multiplicações com zero, com valores baixos e altos, e situações de possível overflow. Os resultados obtidos em cada teste foram coerentes com os valores esperados, demonstrando que o multiplicador realiza a operação corretamente e de forma estável. Além disso, os testes contribuíram para validar a robustez do circuito frente a entradas variadas, bem como o correto controle dos sinais de ativação e término da operação.

Dessa forma, conclui-se que o multiplicador de 8 bits está apto para ser utilizado em aplicações que demandem operações aritméticas rápidas e confiáveis. Recomenda-se, para trabalhos futuros, a análise do comportamento do módulo com números com sinal (caso a arquitetura venha a ser estendida para isso), bem como a otimização de área e desempenho para aplicações em hardware com recursos limitados.

A verificação sistemática por meio de simulação reforça a importância de testbenches bem estruturados como ferramenta fundamental no fluxo de desenvolvimento de sistemas digitais.

Referências

[Souza] Souza, P. Unidade lógico-aritmética usando a abordagem comportamental. https://www.youtube.com/watch?v=Ynymty6-5dMt=6s.