Computer Architecture – Booth algorithm

Booth multiplier

1. Introduction

Booth's multiplication algorithm is a multiplication algorithm that multiplies two binary numbers in two's complement notation. Let's understand the booth algorithm and configure it with Verilog.

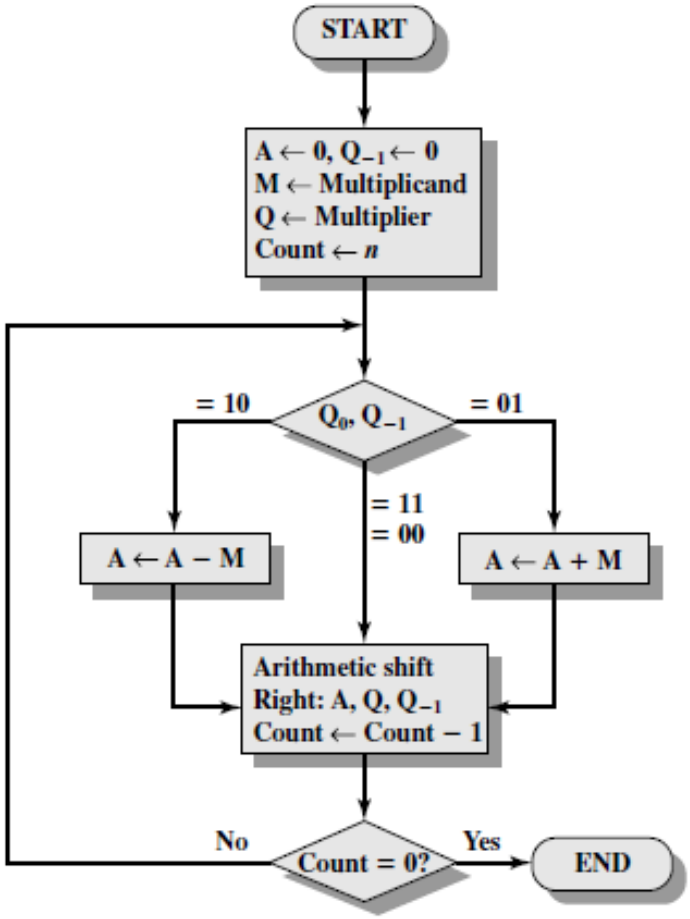

Figure 1 – Booth algorithm diagram

```verilog
module booth(input1, input2, clk, start, reset, result, count);

    input [3:0] input1, input2;
    input clk, start, reset;
    output [7:0] result;
    output reg [1:0] count;

    reg [3:0] A, Q, M;
    reg Q_1;
    wire [3:0] add, sub;

    always @(posedge clk or negedge reset) begin
        if(reset)
        begin //reset
            A <= 4'b0;
            M <= 4'b0;
            Q <= 4'b0;
            Q_1 <= 1'b0;
            count <= 2'b0;
        end

        else if(~start)
        begin //start
            A <= 4'b0;
            M <= input1;
            Q <= input2;
            Q_1 <= 1'b0;
            count <= 2'b0;
        end

        else
        begin
        case({Q[0],Q_1})
        2'b0_1 : {A, Q, Q_1} <= {add[3], add, Q};  //A + M and shift right
        2'b1_0 : {A, Q, Q_1} <= {sub[3], sub, Q}; //A + ~M and shift right
        default : {A, Q, Q_1} <= {A[3], A, Q}; //shift right
        endcase

        count = count + 1'b1;

        end
    end

    alu adder(A, M, 1'b0, add);
    alu subtracter(A, ~M, 1'b1, sub);

    assign result = {A,Q};
endmodule
```

```verilog
module alu(a, b, cin, out);
    input [3:0] a;
    input [3:0] b;
    input cin;
    output [3:0] out;

    assign out = a + b + cin;
endmodule
```

Figure 2 – Booth algorithm verilog code

```
`timescale 1ns/1ps
`include "booth.v"
module tb_booth;
    reg [3:0] input1, input2;
    reg clk, start, reset;
    wire [7:0] result;
    wire [1:0] count;

    booth uut(.input1(input1), .input2(input2), .clk(clk), .start(start), .reset(reset), .result(result), .count(count));

    always #5 clk = ~clk;

    initial begin
        $dumpfile("tb_booth.vcd");
        $dumpvars(0,tb_booth);

        start = 0;
        clk = 0;
        reset = 0;
        input1 = 4;
        input2 = 5;
        #5; start = 1;
        #640; $finish;
    end
endmodule
```

Figure 3– Booth algorithm test bench

2. Assignment

Design booth algorithm with Verilog. Write a Verilog code and analyze the waveform. Additionally, when implementing booth algorithm with Logisim-evolution, there is an additional score.

3. 결과 report (표지 필수)

- 코드를 작성한 후, Icarus Verilog로 input 값을 바꿔가며 컴파일 후에 gtkwave로 파형을 확인하며 해당 결과값에 대한 분석을 상세히 설명할 것
- Logisim-evolution을 이용하여 booth algorithm을 설계하고, 입력을 넣어보며 동작 결과를 확인하고 동작 결과가 왜 그렇게 나오는지 설명할 것 (완성 시, 추가점수 있음)

4. 결과 Report Submission

- Due date: 3월 27일 23:59 까지 (딜레이 받지 않음)
- 결과 Report(pdf)와 Logisim-evolution 파일, 작성한 Verilog 코드를 압축하여 KLAS에 업로드
- **압축 파일명 양식: 학번_이름_Booth.zip**