Hw2 Square Root Finder

103061608 張永忱

GitHub Links of this project:
https://github.com/timerg/SquareVig

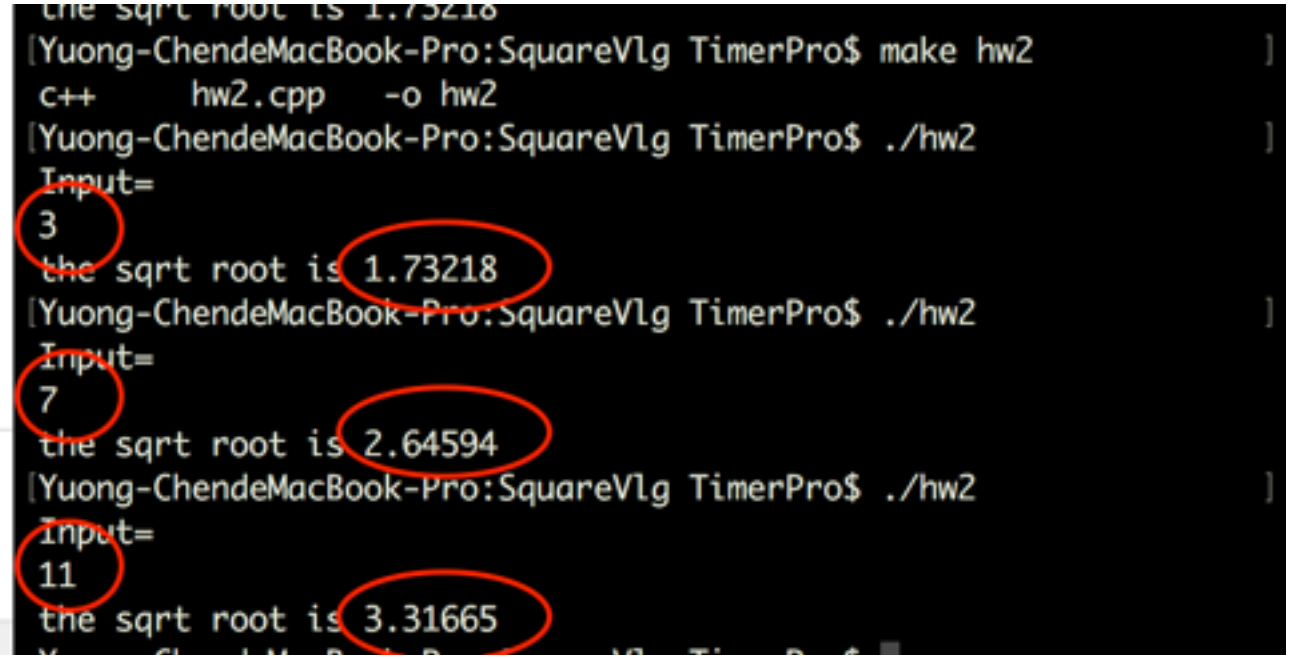
C-code: ***https://github.com/timerg/SquareVig/blob/master/hw2.cpp***
Design Folder: ***https://github.com/timerg/SquareVig/tree/master/EDA_hw2***

Content:

C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>

using namespace std;

double SqrtNumber(double num);
double test(double x, double g, double up, double low);
bool closeEnough(double a, double x);
double betterGuess(double x, double g);

int main(void){
        double num;
        cout << "Input= " << endl;
        cin  >> num ;
        cout << "the sqrt root is " << SqrtNumber(num) << endl;


                return 0;
}
double SqrtNumber(double num)                   //ex:  SqrtNumber(3)
{
   return test(num, 0, num, 0);
}

double test(double x, double g, double up, double low) {          //ex: test(3, 1)
  if (closeEnough(x, g*g))
     return g;
  else
          if ((x - g*g) >= 0 )
          return test(x, betterGuess(up, g), up, g);
          else
                  return test(x, betterGuess(low, g), g, low);
}



bool closeEnough(double a, double x) {      //ex: closeEnough(3/1, 1)

        return (fabs(a - x) < 0.001);           //  the fabs function returns the absolute value of a
floating-point number

}                                                                       // this function is used as
limitation of digits after the decimal-point

double betterGuess(double x, double g) { //ex: betterGuess(3, 1)

  return ((x + g) / 2);

}
```

Code Result


the sqrt root is 1.73218
Yuong-ChendeMacBook-Pro:SquareVlg TimerPro$ make hw2
c++ hw2.cpp -o hw2
Yuong-ChendeMacBook-Pro:SquareVlg TimerPro$ ./hw2
Input=
3
the sqrt root is 1.73218
Yuong-ChendeMacBook-Pro:SquareVlg TimerPro$ ./hw2
Input=
7
the sqrt root is 2.64594
Yuong-ChendeMacBook-Pro:SquareVlg TimerPro$ ./hw2
Input=
11
the sqrt root is 3.31665

Verilog Code

```
module square_root_finder(
    input wire rst,
    input wire clk,
    input wire [15 : 0] in,
    output wire [31 : 0] sqrt
    );
reg [1 : 0] state, state_next;
reg [0 : 0] cat, cat_temp;
reg [31 : 0] up, up_temp, low, low_temp, g, g_temp;
// up: upper Bound; low: lower bound; g -> Guess number
wire [31 : 0] inx;

assign sqrt = g;

always@( * )
    case(state)
        2'b11: begin                    // Final Result
                state_next = state;
                cat_temp = cat;
                up_temp = up;
                low_temp = low;
                g_temp = g;
            end
        2'b00: begin                    // Inital
                state_next = 2'b01;
                cat_temp = 1'b0;
                up_temp = {10'd0, in, 6'd0};
                low_temp = 32'd0;
                g_temp = 32'd0;
            end
        2'b01: begin                    // State 1: Category
            if (inx > (g * g))
                begin
                    if ((inx - g * g) < 32'd02) begin
                        state_next = 2'b11;
                        cat_temp = cat;
                        up_temp = up;
                        low_temp = low;
                        g_temp = g;

                    end
                    else begin
                        cat_temp = 1'b1;
                        state_next = 2'b10;
                        up_temp = up;
                        low_temp = low;
                        g_temp = g;
                    end
                end
            else
                begin
                    if ((g * g - inx) < 32'd02) begin
                        state_next = 2'b11;
```

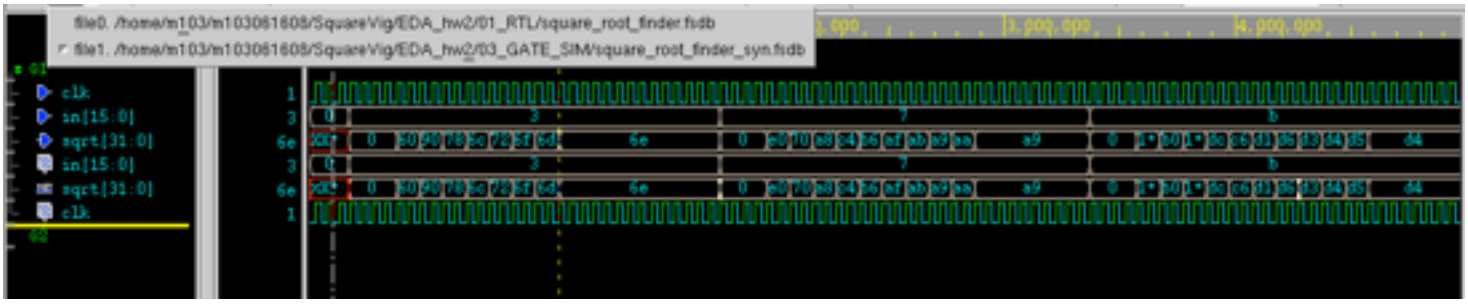
```
                cat_temp = cat;
                up_temp = up;
                low_temp = low;
                g_temp = g;
            end
            else begin
                cat_temp = 1'b0;
                state_next = 2'b10;
                up_temp = up;
                low_temp = low;
                g_temp = g;
            end
        end
    end
    2'b10: begin                    // State 2: Update Bound and Guess
        case(cat)
            1'b1: begin
                cat_temp = cat;
                up_temp = up;
                low_temp = g;
                g_temp = ((up + g) >> 1);// Use average of upper and
                state_next = 2'b01;      // lower bound as new guess number
            end
            1'b0: begin
                cat_temp = cat;
                up_temp = g;
                low_temp = low;
                g_temp = ((g + low) >> 1);
                state_next = 2'b01;
            end
        endcase
    end
endcase

// Temp -> reg(at clk) & Rst
always @ (negedge rst or  posedge clk) begin
if(~rst)begin
    state <= 2'b00;
    end
else begin
    state <= state_next;
    end
end

assign inx = {4'd0, in, 12'd0};

always @ (negedge rst or  posedge clk) begin
if(~rst)begin
    up <= {10'd0, in, 6'd0};
    end
else begin
    up <= up_temp;
    end
end
```

```
always @ (negedge rst or  posedge clk) begin
if(~rst)begin
    low <= 32'd00;
    end
else begin
    low <= low_temp;
    end
end

always @ (negedge rst or  posedge clk) begin
if(~rst)begin
    cat <= 1'b0;
    end
else begin
    cat <= cat_temp;
    end
end

always @ (negedge rst or  posedge clk) begin
if(~rst)begin
    g <= 32'd00;
    end
else begin
    g <= g_temp;
    end
end

endmodule
```

Test File

```
`timescale 1ns/1ps

`define CLK 50

module testbench;

reg clk ,rst;
reg  [15:0]in;
wire [31:0]sqrt;

square_root_finder square_root_finder1(.clk(clk), .rst(rst), .in(in), .sqrt(sqrt));

always begin #(`CLK/2) clk = ~ clk; end

initial begin
   clk = 1'b0;
   rst = 1'b1;
   in =  16'd0;
   #170 in = 16'd3; rst = 1'b0;
   #100 rst = 1'b1;

   #1500 in = 16'd7; rst = 1'b0;
   #100 rst = 1'b1;

   #1500 in = 16'd11; rst = 1'b0;
   #100 rst = 1'b1;

   #1500 $finish;
end




 initial begin
   `ifdef RTL
            $fsdbDumpfile("square_root_finder.fsdb");
                  $fsdbDumpvars;
                  `endif
   `ifdef GATE
            $sdf_annotate("../02_SYN/Netlist/square_root_finder_syn.sdf" , square_root_finder1);
      $fsdbDumpfile("square_root_finder_syn.fsdb");
      $fsdbDumpvars;
            `endif
   #6000 $finish;
  end
endmodule
```

Verily Result (both synthesized & non-synthesized)



The above three rows are from non-synthesis, where as the below three rows are from synthesized. We can seethe both results are same. So the design after synthesis works well.
p.s.
For input = 3, output = $(6e)_{16} = (110)_{10} = (01101110)_2$
In my design, the last 6 digits represented numbers after Decimal Point. So the result turns out to be: 1*1 + 1*1/2 + 1* 1/8 + 1*1/16 + 1*1/32 = 1.71875.

For input = 7, output = $(a9)_{16} = (169)_{10} = (10101001)_2 = 2.640625$

For input = 11, output = $(a9)_{16} = (212)_{10} = (11010100)_2 = 3.3125$

Performance

Area: 133285 um^2 ; Gate Count = 13328.5

```
Information: Updating design information... (UID-85)
Library(s) Used:

    slow (File: /home/m103/m103061604/EDA_hw2/db/slow.db)

Number of ports:              50
Number of nets:              662
Number of cells:             446
Number of references:         35

Combinational area:       126406.527644
Noncombinational area:    6878.995293
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          133285.522937
```

Power: 6.7582uW

```
  Cell Internal Power  = 297.1032 uW   (79%)
  Net Switching Power  =  78.4622 uW   (21%)
                         ---------
Total Dynamic Power    = 375.5653 uW  (100%)

Cell Leakage Power     =   6.7852 uW
```

Max Speed : 1/25ns = 40Meg

```
  U325/Y (NOR4X1)                                  0.28      19.28 r
  U321/Y (NAND4X1)                                 0.14      19.42 f
  U319/Y (NOR2X1)                                  0.16      19.58 r
  U314/Y (AOI211XL)                                0.14      19.71 f
  U311/Y (OAI22X1)                                 0.14      19.85 r
  cat_reg_0_/D (DFFRXL)                            0.00      19.85 r
  data arrival time                                          19.85

  clock clk (rise edge)                           20.00      20.00
  clock network delay (ideal)                      0.00      20.00
  cat_reg_0_/CK (DFFRXL)                           0.00      20.00 r
  library setup time                              -0.13      19.87
  data required time                                         19.87
  ------------------------------------------------------------------
  data required time                                         19.87
  data arrival time                                         -19.85
  ------------------------------------------------------------------
  slack (MET)                                                 0.02
```

Netlist

The setlist is too large. Please Refer to Github links below:

https://github.com/timerg/SquareVig/tree/master/EDA_hw2/02_SYN/Netlist