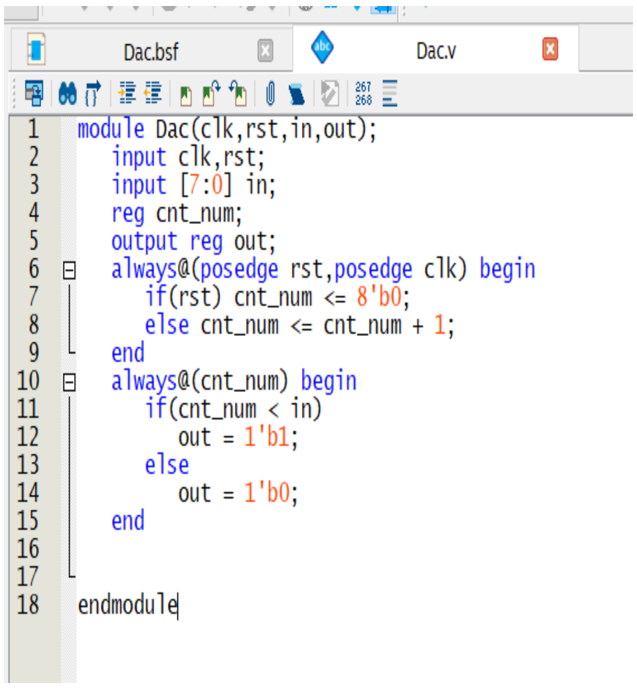
Experiment #3 - Function Generator

Sepehr Azardar
810199357

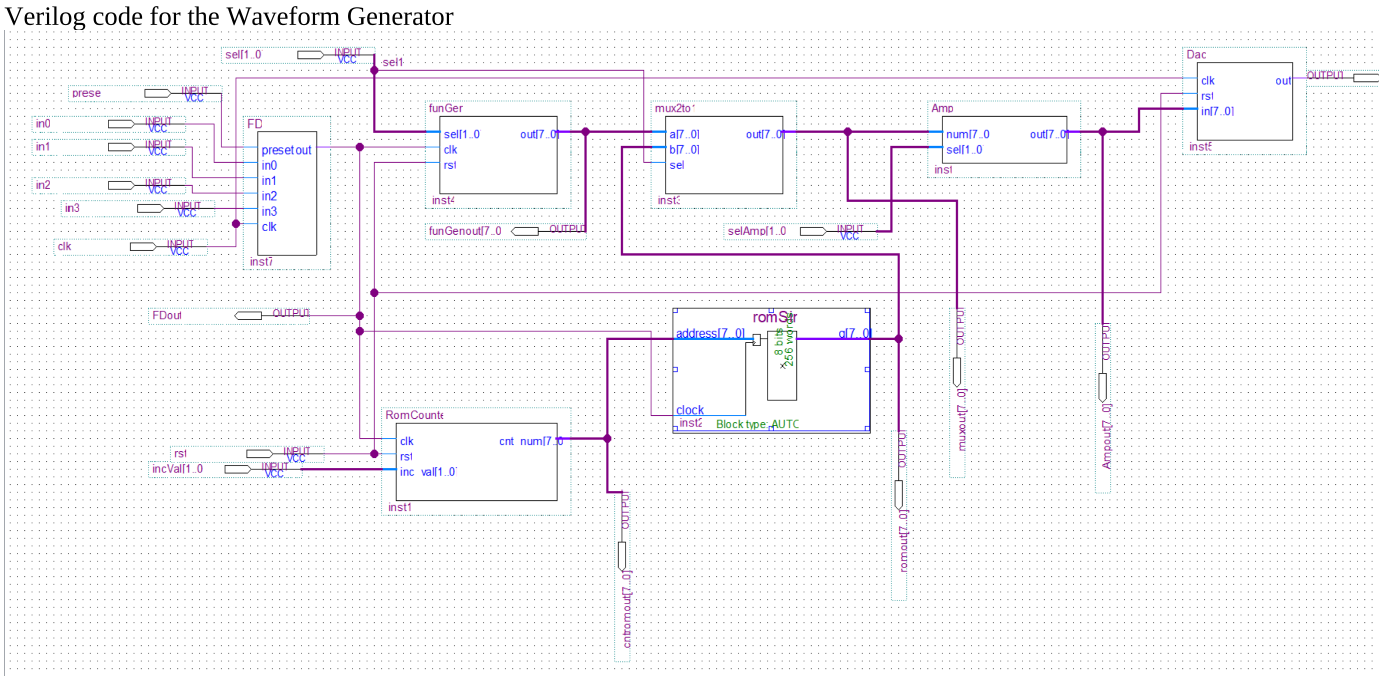
Amirhossein
Kahrobaeian
810199478

1.DAC

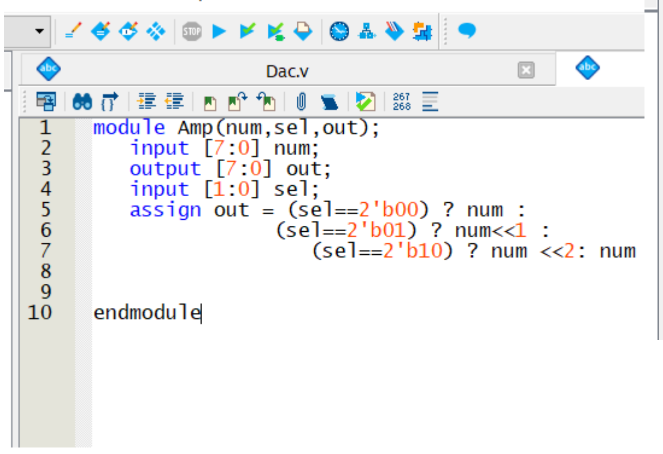


```
module Dac(clk,rst,in,out);
   input clk,rst;
   input [7:0] in;
   reg cnt_num;
   output reg out;
   always@(posedge rst,posedge clk) begin
      if(rst) cnt_num <= 8'b0;
      else cnt_num <= cnt_num + 1;
   end
   always@(cnt_num) begin
      if(cnt_num < in)
         out = 1'b1;
      else
         out = 1'b0;
   end


endmodule
```

Verilog code for the Waveform Generator

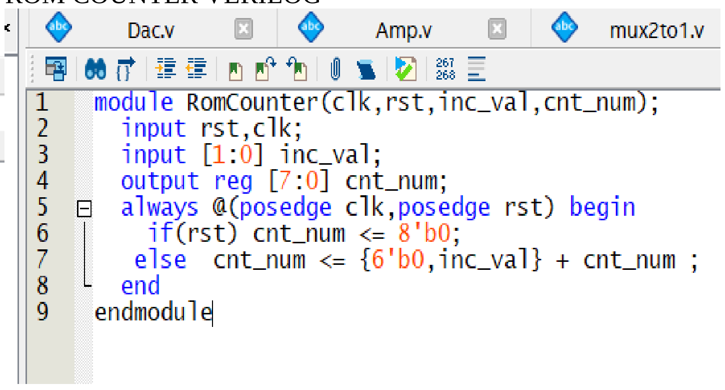


AMP VERILOG



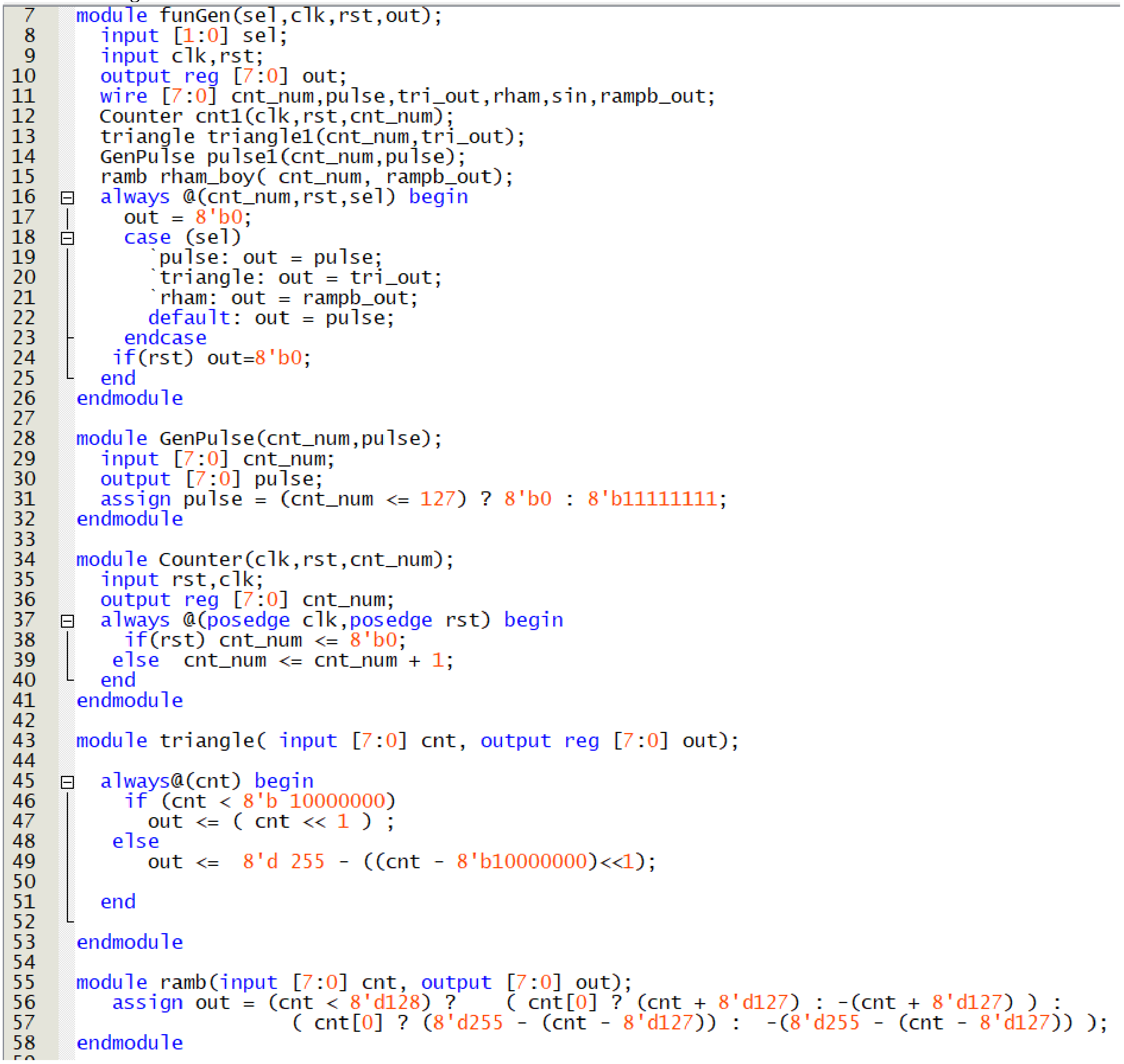
```
module Amp(num,sel,out);
   input [7:0] num;
   output [7:0] out;
   input [1:0] sel;
   assign out = (sel==2'b00) ? num :
                   (sel==2'b01) ? num<<1  :
                      (sel==2'b10) ? num <<2: num



endmodule
```

ROM COUNTER VERILOG



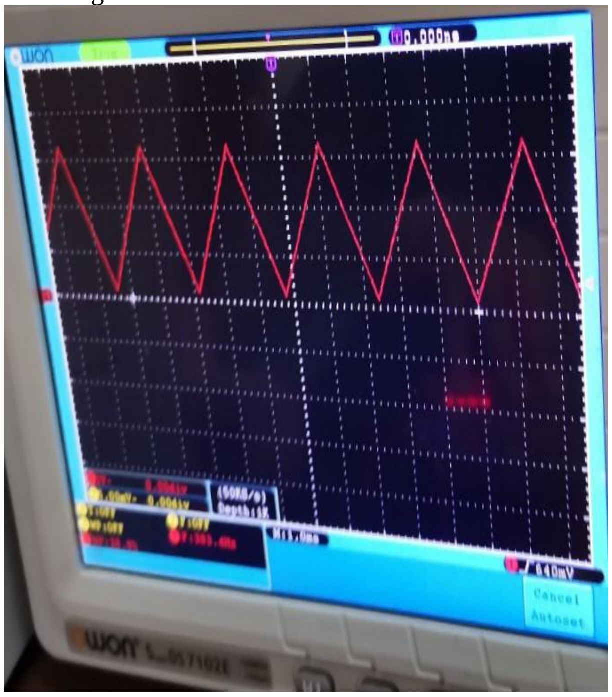
```
module RomCounter(clk,rst,inc_val,cnt_num);
  input rst,clk;
  input [1:0] inc_val;
  output reg [7:0] cnt_num;
  always @(posedge clk,posedge rst) begin
    if(rst) cnt_num <= 8'b0;
   else  cnt_num <= {6'b0,inc_val} + cnt_num ;
  end
endmodule
```

FunGen Verilog

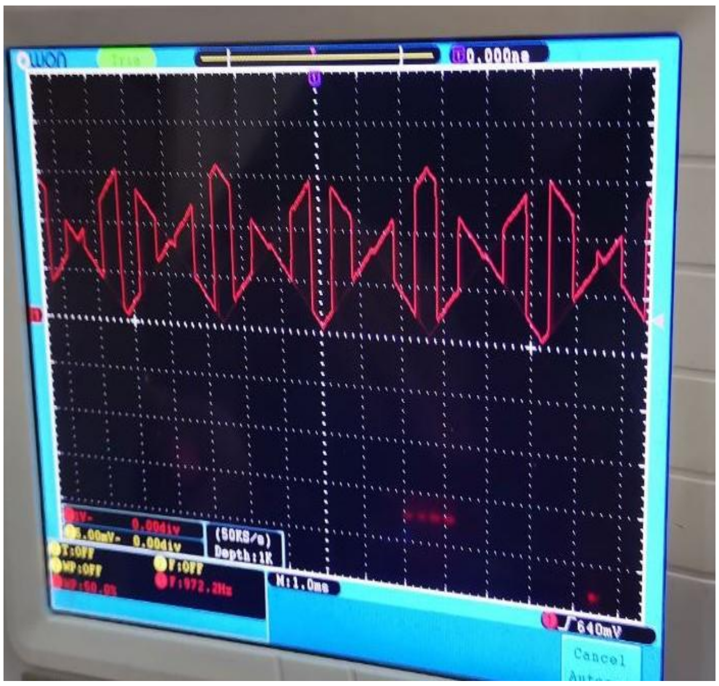


```verilog
module funGen(sel,clk,rst,out);
  input [1:0] sel;
  input clk,rst;
  output reg [7:0] out;
  wire [7:0] cnt_num,pulse,tri_out,rham,sin,rampb_out;
  Counter cnt1(clk,rst,cnt_num);
  triangle triangle1(cnt_num,tri_out);
  GenPulse pulse1(cnt_num,pulse);
  ramb rham_boy( cnt_num, rampb_out);
  always @(cnt_num,rst,sel) begin
    out = 8'b0;
    case (sel)
      `pulse: out = pulse;
      `triangle: out = tri_out;
      `rham: out = rampb_out;
      default: out = pulse;
    endcase
   if(rst) out=8'b0;
  end
endmodule

module GenPulse(cnt_num,pulse);
  input [7:0] cnt_num;
  output [7:0] pulse;
  assign pulse = (cnt_num <= 127) ? 8'b0 : 8'b11111111;
endmodule

module Counter(clk,rst,cnt_num);
  input rst,clk;
  output reg [7:0] cnt_num;
  always @(posedge clk,posedge rst) begin
    if(rst) cnt_num <= 8'b0;
   else  cnt_num <= cnt_num + 1;
  end
endmodule

module triangle( input [7:0] cnt, output reg [7:0] out);

  always@(cnt) begin
    if (cnt < 8'b 10000000)
      out <= ( cnt << 1 ) ;
   else
      out <=  8'd 255 - ((cnt - 8'b10000000)<<1);

  end

endmodule

module ramb(input [7:0] cnt, output [7:0] out);
   assign out = (cnt < 8'd128) ?    ( cnt[0] ? (cnt + 8'd127) : -(cnt + 8'd127) ) :
                  ( cnt[0] ? (8'd255 - (cnt - 8'd127)) :  -(8'd255 - (cnt - 8'd127)) );
endmodule
```

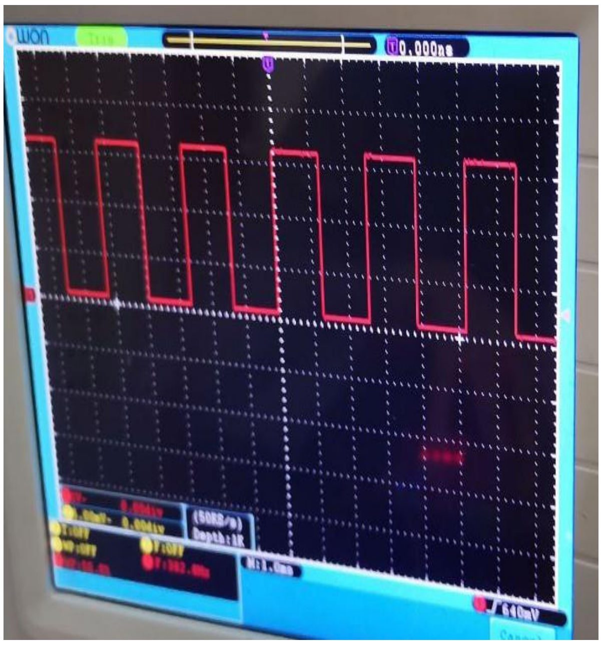
Triangle



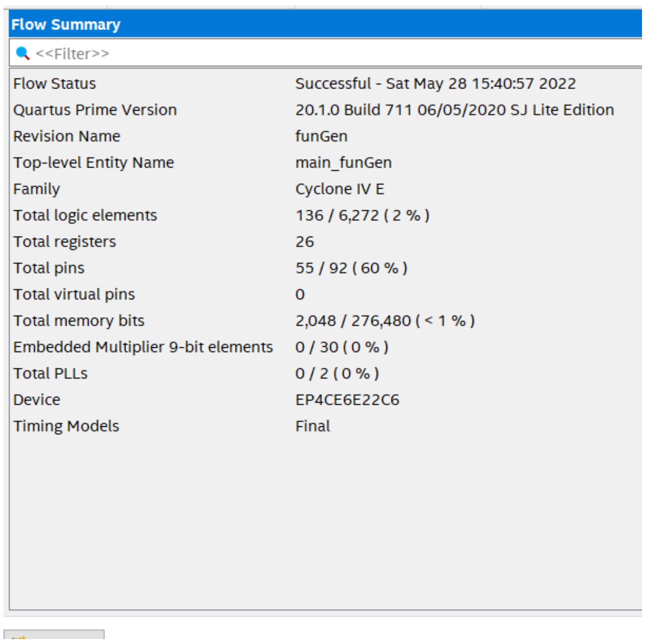
Rhomboid



Square



quartus flow summary



Flow Summary

<<Filter>>

Flow Status	Successful - Sat May 28 15:40:57 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	funGen
Top-level Entity Name	main_funGen
Family	Cyclone IV E
Total logic elements	136 / 6,272 (2 %)
Total registers	26
Total pins	55 / 92 (60 %)
Total virtual pins	0
Total memory bits	2,048 / 276,480 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP4CE6E22C6
Timing Models	Final

sin