Lab Report 7

Tuesday 2:15pm

Yen-Jung(Tim) Lu, Antonio Fontan

Introduction

Short paragraph describing what the lab is about.
In this lab, we are building a sine-wave generator using a table-driven arbitrary waveform generator. Tasks include modifying a C model, developing a corresponding Verilog implementation, creating a testbench, and validating model matches. We will adjust for different sample rates and implement a new architecture to ensure consistent output values.

Procedure

Describe what you did during the lab. How you wired up the board, what code you wrote (include snippets of the code you wrote; the rest of the code should be in the appendix), etc…
I'm not expecting an extremely verbose retelling of each step of the lab.

Part 1

We implemented an interpolated sine wave generator based on the block diagram shown in the lab guideline document.

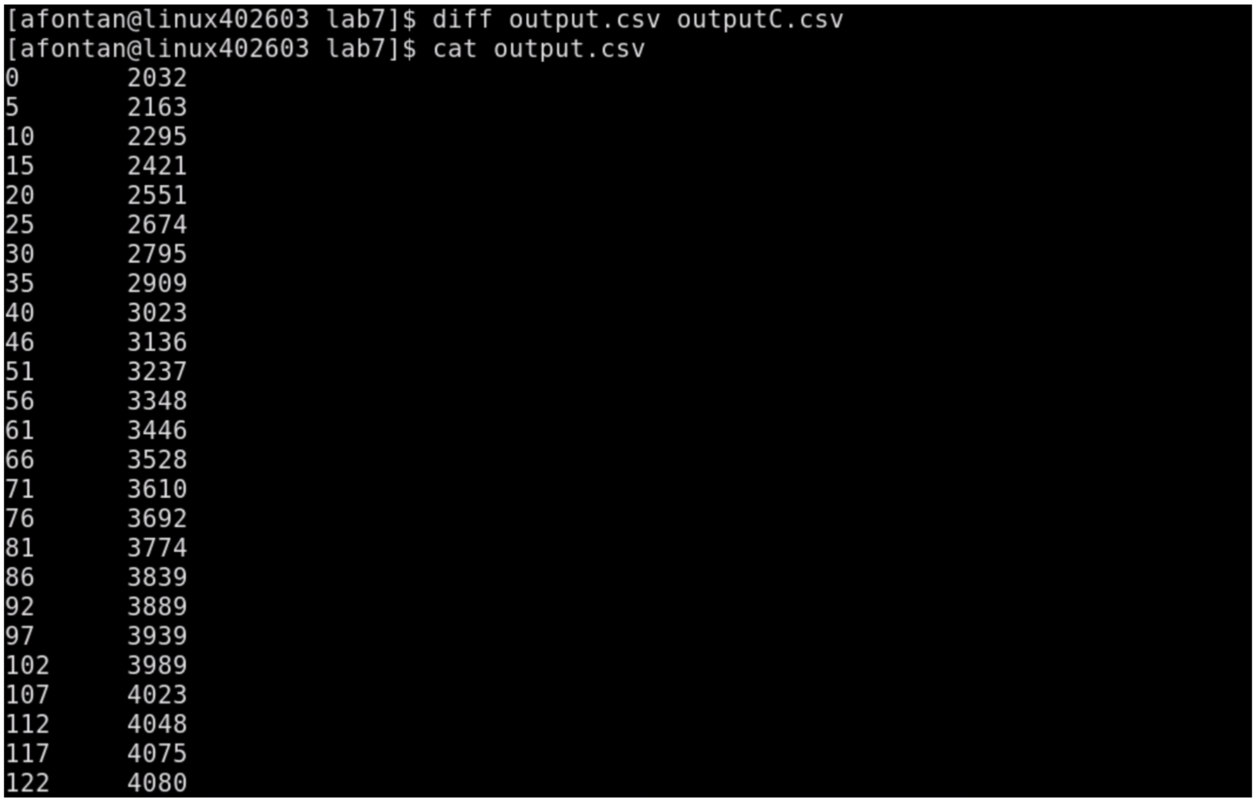
Part 2

Develop a Verilog implementation and create a testbench that generates two cycles of output at 100 samples per cycle, while also keeping the c code in mind in order to produce the same output.

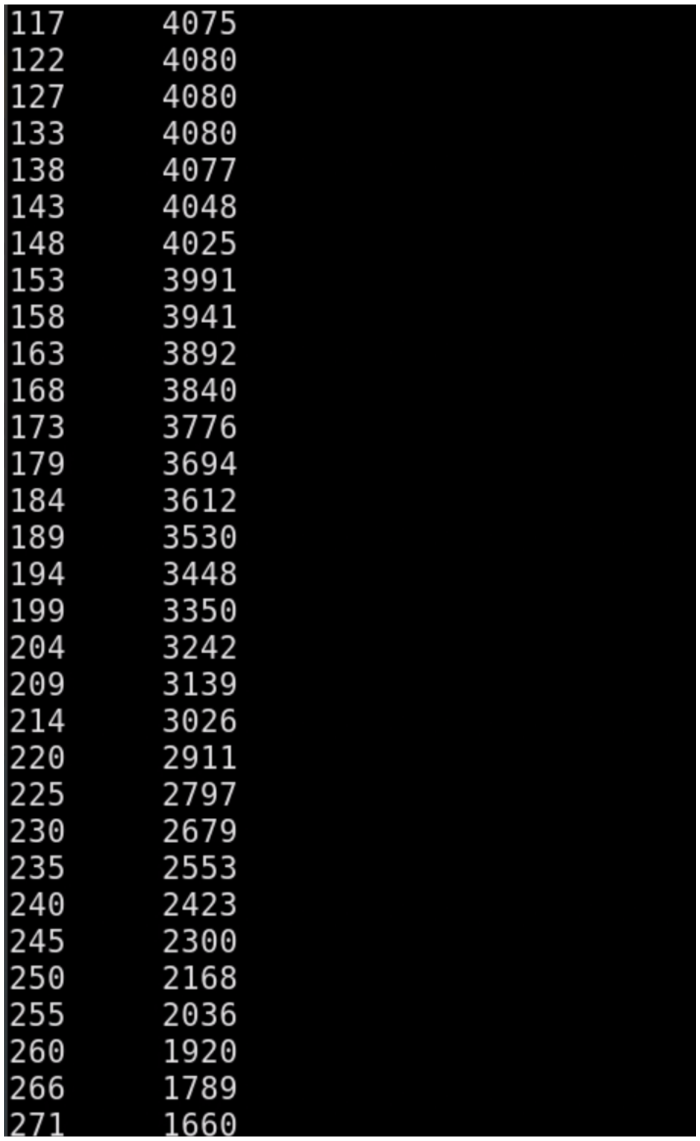
Results

Place the results of your experiments and tasks here. Tables, charts, screenshots, etc..
Answer any questions from the lab document here.

Part 1



```
[afontan@linux402603 lab7]$ diff output.csv outputC.csv
[afontan@linux402603 lab7]$ cat output.csv
0       2032
5       2163
10      2295
15      2421
20      2551
25      2674
30      2795
35      2909
40      3023
46      3136
51      3237
56      3348
61      3446
66      3528
71      3610
76      3692
81      3774
86      3839
92      3889
97      3939
102     3989
107     4023
112     4048
117     4075
122     4080
```



```
117      4075
122      4080
127      4080
133      4080
138      4077
143      4048
148      4025
153      3991
158      3941
163      3892
168      3840
173      3776
179      3694
184      3612
189      3530
194      3448
199      3350
204      3242
209      3139
214      3026
220      2911
225      2797
230      2679
235      2553
240      2423
245      2300
250      2168
255      2036
260      1920
266      1789
271      1660
```

276	1530
281	1409
286	1287
291	1173
296	1059
301	945
307	847
312	733
317	635
322	553
327	472
332	390
337	308
342	242

347	192
353	142
358	92
363	58
368	32
373	7
378	0
383	0
388	0
394	0
399	32
404	52
409	86
414	135
419	185
424	240
429	304
434	383
440	465
445	547
450	629
455	726
460	833
465	938
470	1049
475	1164
481	1280
486	1396
491	1524
496	1653
501	1775
506	1907

Part 2

```
[afontan@linux402603 lab7]$ cat rom_data.txt
01111111
10000001
10000011
10000100
10000110
10000111
10001001
10001010
```

10001100
10001110
10001111
10010001
10010010
10010100
10010101
10010111
10011000
10011010
10011100
10011101
10011111
10100000
10100010
10100011
10100101
10100110
10101000
10101001
10101011
10101100
10101110
10101111
10110000
10110010
10110011
10110101
10110110
10111000
10111001
10111010
10111100
10111101
10111111
11000000
11000001
11000011
11000100
11000101
11000111
11001000
11001001
11001010

11001100
11001101
11001110
11001111
11010001
11010010
11010011
11010100
11010101
11010111
11011000
11011001
11011010
11011011
11011100
11011101
11011110
11011111
11100000
11100001
11100010
11100011
11100100
11100101
11100110
11100111
11101000
11101001
11101010
11101011
11101100
11101100
11101101
11101110
11101111
11110000
11110000
11110001
11110010
11110011
11110011
11110100
11110101
11110101

11110110
11110110
11110111
11110111
11111000
11111001
11111001
11111010
11111010
11111010
11111011
11111011
11111100
11111100
11111100
11111101
11111101
11111101
11111110
11111110
11111110
11111110
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111110

11111110
11111110
11111110
11111101
11111101
11111101
11111100
11111100
11111100
11111011
11111011
11111010
11111010
11111010
11111001
11111001
11111000
11110111
11110111
11110110
11110110
11110101
11110101
11110100
11110011
11110011
11110010
11110001
11110000
11110000
11101111
11101110
11101101
11101100
11101100
11101011
11101010
11101001
11101000
11100111
11100110
11100101
11100100
11100011

11100010
11100001
11100000
11011111
11011110
11011101
11011100
11011011
11011010
11011001
11011000
11010111
11010101
11010100
11010011
11010010
11010001
11001111
11001110
11001101
11001100
11001010
11001001
11001000
11000111
11000101
11000100
11000011
11000001
11000000
10111111
10111101
10111100
10111010
10111001
10111000
10110110
10110101
10110011
10110010
10110000
10101111
10101110
10101100

10101011
10101001
10101000
10100110
10100101
10100011
10100010
10100000
10011111
10011101
10011100
10011010
10011000
10010111
10010101
10010100
10010010
10010001
10001111
10001110
10001100
10001010
10001001
10000111
10000110
10000100
10000011
10000001
01111111
01111110
01111100
01111011
01111001
01111000
01110110
01110101
01110011
01110001
01110000
01101110
01101101
01101011
01101010
01101000

01100111
01100101
01100011
01100010
01100000
01011111
01011101
01011100
01011010
01011001
01010111
01010110
01010100
01010011
01010001
01010000
01001111
01001101
01001100
01001010
01001001
01000111
01000110
01000101
01000011
01000010
01000000
00111111
00111110
00111100
00111011
00111010
00111000
00110111
00110110
00110101
00110011
00110010
00110001
00110000
00101110
00101101
00101100
00101011

```
00101010
00101000
00100111
00100110
00100101
00100100
00100011
00100010
00100001
00100000
00011111
00011110
00011101
00011100
00011011
00011010
00011001
00011000
00010111
00010110
00010101
00010100
00010011
00010011
00010010
00010001
00010000
00001111
00001111
00001110
00001101
00001100
00001100
00001011
00001010
00001010
00001001
00001001
00001000
00001000
00000111
00000110
00000110
00000101
```

```
00000101
00000101
00000100
00000100
00000011
00000011
00000011
00000010
00000010
00000010
00000001
00000001
00000001
00000001
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000001
00000001
00000001
00000001
00000010
00000010
00000010
00000011
00000011
```

```
00000011
00000100
00000100
00000101
00000101
00000101
00000110
00000110
00000111
00001000
00001000
00001001
00001001
00001010
00001010
00001011
00001100
00001100
00001101
00001110
00001111
00001111
00010000
00010001
00010010
00010011
00010011
00010100
00010101
00010110
00010111
00011000
00011001
00011010
00011011
00011100
00011101
00011110
00011111
00100000
00100001
00100010
00100011
00100100
```

00100101
00100110
00100111
00101000
00101010
00101011
00101100
00101101
00101110
00110000
00110001
00110010
00110011
00110101
00110110
00110111
00111000
00111010
00111011
00111100
00111110
00111111
01000000
01000010
01000011
01000101
01000110
01000111
01001001
01001010
01001100
01001101
01001111
01010000
01010001
01010011
01010100
01010110
01010111
01011001
01011010
01011100
01011101
01011111

```
01100000
01100010
01100011
01100101
01100111
01101000
01101010
01101011
01101101
01101110
01110000
01110001
01110011
01110101
01110110
01111000
01111001
01111011
01111100
01111110
```

Conclusion

Short paragraph talking about the takeaways from this lab.
Through this lab, we gained hands-on experience in building and testing sine-wave generators, employing table-driven waveform generators in two different architectures. The process involved C model modifications, Verilog implementation, and validation through testbenches.

Appendix

Place code and other data here.

Part 1:

```
module PhaseRegister(
   input wire clk,
   input wire load,
   input wire [15:0] phaseIn,
   output reg [15:0] phaseIncrement
);
   always @(posedge clk) begin
      if (load)
          phaseIncrement <= phaseIn;
   end
endmodule

module PhaseIncrement16(
   input [15:0] phaseIncrementIn1,
   input [15:0] phaseIncrementIn2,
   output [15:0] phaseIncrementOut
);
   assign phaseIncrementOut = phaseIncrementIn1 + phaseIncrementIn2;
endmodule

module PhaseIncrement7(
   input [6:0] phaseIncrementIn1,
   input [6:0] phaseIncrementIn2,
   output [6:0] phaseIncrementOut
);
   assign phaseIncrementOut = phaseIncrementIn1 + phaseIncrementIn2;
endmodule

module PhaseAccumulator(
   input wire clk,
   input wire rst,
   input wire [15:0] phaseIncrement,
   output reg [15:0] phaseAccumulated
);
   always @(posedge clk or posedge rst) begin
      if (rst)
         phaseAccumulated <= 0;
      else
         phaseAccumulated <= phaseAccumulated + phaseIncrement;
   end
```

```
endmodule


module Rounder(
   input [15:0] in,
   output [8:0] decimal,
   output [6:0] out
);
   assign decimal = in[8:0];
   assign out = in[15:9];
endmodule


module WavetableROM(
   input [6:0] address,
   output [7:0] data
);
         reg [7:0] memory [0:511];

      initial begin
            $readmemb("rom_data.txt", memory);
      end

      assign data = memory[address];


endmodule

module Interpolator(
   input [7:0] data1,
   input [7:0] data2,
   input [8:0] decimal,
   output [11:0] result
);
   wire [8:0]fraction = decimal/512;
   wire [11:0] scaledData1 = data1 << 4;
   wire [11:0] scaledData2 = data2 << 4;
   assign result = (1-fraction)*scaledData1 + fraction * scaledData2;

endmodule

module topModule(
         input [15:0] phaseIn,
         input load,
        input clk,
        input rst,
        output [11:0] result
);

wire [15:0] phaseIncrement1Out;
PhaseRegister phaser(
   clk,
   load,
   phaseIn,
   phaseIncrement1Out
);

wire [15:0] phaseIncrement2Out;
wire [15:0] phaseAccumulatedIn;
PhaseIncrement16 phaseInc1(
   phaseIncrement1Out,
   phaseAccumulatedIn,
   phaseIncrement2Out
);

wire [15:0] phaseIncremented2In = phaseIncrement2Out;
wire [15:0] phaseAccumulatedOut;
PhaseAccumulator Accumulator(
   clk,
   rst,
   phaseIncremented2In,
   phaseAccumulatedOut
);

assign phaseAccumulatedIn = phaseAccumulatedOut;
wire [6:0] out;
wire [8:0] decimalOut;
Rounder round(
   phaseAccumulatedOut,
   decimalOut,
   out
);

wire [7:0] data1Out;
WavetableROM ROM1(
   out,
   data1Out
);
```

```
wire [6:0] address2Out;
PhaseIncrement7 phaseInc2(
    out,
    7'd1,
    address2Out
);

wire [7:0] data2Out;
WavetableROM ROM2(
    address2Out,
    data2Out
);

Interpolator inter(
    data1Out,
    data2Out,
    decimalOut,
    result
);

endmodule

module Testbench;

  reg [15:0] phaseIn;
  reg load;
  reg clk;
  reg rst;
  wire [11:0] result;
  integer file;
  topModule uut(
    .phaseIn(phaseIn),
    .load(load),
    .clk(clk),
    .rst(rst),
    .result(result)
  );


reg [15:0] phase;
  initial begin
    #10;
    clk = 0;
    rst = 0;
    #10
    rst = 1;
    load = 1;
    phaseIn = 16'b1010001111100100;
    clk = ~clk;
    #10
    rst = 0;
    load = 0;
    phaseIn = 16'b1010001111100100;
    file = $fopen("output.csv", "w");
    clk = ~clk;
    #10
    load = 0;
    repeat (100) begin
      #10 clk = ~clk;
      $display("#%t, Result = %d, clk=%d", $time, result, clk);
      $fwrite(file, "%0d\n", result);
      #10 clk = ~clk;
    end
    #10;
    $fclose(file);
    #10;
    $finish;
  end

endmodule
```

part 2:

```
module Testbench;

  reg [15:0] phaseIn;
  reg load;
  reg clk;
  reg rst;
  wire [11:0] result;
  integer file;
  topModule uut(
    .phaseIn(phaseIn),
    .load(load),
    .clk(clk),
    .rst(rst),
    .result(result)
  );
```

```
reg [15:0] phase;
  initial begin
    #10;
    clk = 0;
    rst = 0;
    #10
    rst = 1;
    load = 1;
    phaseIn = 16'b1010001111100100;
    clk = ~clk;
    #10
    rst = 0;
    load = 0;
    phaseIn = 16'b1010001111100100;
    file = $fopen("output.csv", "w");
    clk = ~clk;
    #10
    load = 0;
    repeat (75) begin
      #10 clk = ~clk;
      $display("#%t, Result = %d, clk=%d", $time, result, clk);
      $fwrite(file, "%0d\n", result);
      #10 clk = ~clk;
    end
    #10;
    $fclose(file);
    #10;
    $finish;
  end

endmodule
```

part 3:

```
module PhaseRegister(
    input wire clk,
    input wire load,
    input wire [15:0] phaseIn,
    output reg [15:0] phaseIncrement
);
    always @(posedge clk) begin
       if (load)
            phaseIncrement <= phaseIn;
    end
endmodule

module controlPath(
            input rst,
            input clk,
            output sample
);

    always @(posedge clk or posedge rst) begin
       if (rst)
            sample = 0;
                        else
                                    sample = 1;
    end

endmodule

module PhaseIncrement16(
    input [15:0] phaseIncrementIn1,
    input [15:0] phaseIncrementIn2,
    output [15:0] phaseIncrementOut
);
    assign phaseIncrementOut = phaseIncrementIn1 + phaseIncrementIn2;
endmodule

module PhaseIncrement7(
    input [6:0] phaseIncrementIn1,
    input [6:0] phaseIncrementIn2,
    output [6:0] phaseIncrementOut
);
    assign phaseIncrementOut = phaseIncrementIn1 + phaseIncrementIn2;
endmodule

module PhaseAccumulator(
    input wire clk,
    input wire rst,
    input wire [15:0] phaseIncrement,
    output reg [15:0] phaseAccumulated
);
    always @(posedge clk or posedge rst) begin
        if (rst)
            phaseAccumulated <= 0;
        else
```

```
            phaseAccumulated <= phaseAccumulated + phaseIncrement;
    end
endmodule


module Rounder(
    input [15:0] in,
    output [8:0] decimal,
    output [6:0] out
);
    assign decimal = in[8:0];
    assign out = in[15:9];
endmodule


module WavetableROM(
    input [6:0] address,
    output [7:0] data
);
            reg [7:0] memory [0:511];

        initial begin
                $readmemb("rom_data.txt", memory);
        end

        assign data = memory[address];


endmodule

module Interpolator(
    input [7:0] data1,
    input [7:0] data2,
    input [8:0] decimal,
            input sample,
    output reg [11:0] result
);
    wire [8:0]fraction = decimal/512;
    wire [11:0] scaledData1 = data1 << 4;
    wire [11:0] scaledData2 = data2 << 4;
            always @(posedge sample) begin
        if (sample)
            result = (1-fraction)*scaledData1 + fraction * scaledData2;
    end
```

```
endmodule

module topModule(
        input [15:0] phaseIn,
    input load,
        input clk,
        input rst,
        output [11:0] result
);

wire [15:0] phaseIncrement1Out;
PhaseRegister phaser(
    clk,
    load,
    phaseIn,
    phaseIncrement1Out
);


wire [15:0] phaseIncrement2Out;
wire [15:0] phaseAccumulatedIn;
PhaseIncrement16 phaseInc1(
    phaseIncrement1Out,
    phaseAccumulatedIn,
    phaseIncrement2Out
);

wire [15:0] phaseIncremented2In = phaseIncrement2Out;
wire [15:0] phaseAccumulatedOut;
PhaseAccumulator Accumulator(
    clk,
    rst,
    phaseIncremented2In,
    phaseAccumulatedOut
);

assign phaseAccumulatedIn = phaseAccumulatedOut;
wire [6:0] out;
wire [8:0] decimalOut;
Rounder round(
    phaseAccumulatedOut,
    decimalOut,
    out
);
wire sample;

controlPath control(
            rst,
            clk,
            sample,
);

wire [7:0] data1Out;
WavetableROM ROM1(
    out,
    data1Out
);


wire [6:0] address2Out;
PhaseIncrement7 phaseInc2(
    out,
    7'd1,
    address2Out
);


Interpolator inter(
    data1Out,
    data2Out,
    decimalOut,
            sample,
    result
);

endmodule
```