Design and Implementation of a Threshold-based 8-bit DAC on ZedBoard

Bathula Venkata Naga Motheendranadh Reddy N.Sundeep Teja

November 3, 2025

Abstract

This report presents the design, simulation and FPGA implementation of a threshold-based digital-to-analog converter (DAC) model on the ZedBoard (Xilinx Zynq-7000). The design converts an 8-bit digital input into quantized analog steps based on a 3.3 V reference and maps the result to four LEDs on the board. The project includes synthesizable Verilog, a top-level wrapper, testbench, and XDC constraints for the ZedBoard.

Contents

1 Introduction

A digital-to-analog converter (DAC) maps discrete digital codes into corresponding analog voltages. The presented design is a *threshold-based* DAC: it computes which discrete analog step an input code corresponds to rather than generating a continuous analog waveform. This approach is useful for FPGA demonstration and control applications where an external DAC or PWM reconstruction may be used for an actual analog output.

2 Design Requirements

- Input: 8-bit digital value (0–255) from board switches.
- Reference: 3.3 V (provided as $\texttt{Vref}_mV = 3300$).
- `Output: 4-bit quantized step (0-8) shown on 4 LEDs (LD0-LD3).`
- `Implementation target: ZedBoard (Xilinx Zynq-7000).`

3 Theory

Given an 8-bit digital input D and a reference voltage V_{ref}, the analog voltage equivalent is [$\mathrm{V}_{out} = \frac{D}{255} \times V_{ref}.]The design divides the analog range into 0.4\,\mathrm{V} steps (400 mV per step). The step index is therefore ec$
$\left\lfloor \frac{V_{out}}{0.4} \right\rfloor = \left\lfloor \frac{(V_{ref_mV} \times D)/255}{400} \right\rfloor .]All computations in the Verilog implementation use integer arithmetic for synthesis c$

4 Verilog Implementation

4.1 Core module: $\texttt{dac}_2.v$

```
‘timescale 1ns / 1ps
// Synthesizable Threshold-based DAC
module dac_2 (
input wire [7:0] digital_in, // 8-bit digital input
input wire [11:0] Vref_mV, // 12-bit reference (e.g., 3300)
output reg [3:0] analog_step // 4-bit quantized output
);
reg [19:0] analog_value_mV;

‘‘‘
always @(*) begin
    analog_value_mV = (Vref_mV * digital_in) / 255;
    analog_step = analog_value_mV / 400;
end
‘‘‘

endmodule
```

4.2 Top-level wrapper: $\texttt{dac}_top.v$

```
‘timescale 1ns / 1ps
module dac_top (
input wire clk,
input wire rstn,
input wire [7:0] sw,
```

```
output wire [3:0] led
);
wire [11:0] Vref_mV = 12'd3300;

```
dac_2 u_dac (
    .digital_in(sw),
    .Vref_mV(Vref_mV),
    .analog_step(led)
);
```

endmodule
```

4.3 Testbench: $\texttt{tb}_d ac_2.v$

```
`timescale 1ns / 1ps
module tb_dac_2;
reg [7:0] digital_in;
reg [11:0] Vref_mV;
wire [3:0] analog_step;
real analog_value_mV;

```
dac_2 uut (
    .digital_in(digital_in),
    .Vref_mV(Vref_mV),
    .analog_step(analog_step)
);

initial begin
    Vref_mV = 12'd3300;
    $display("Time(ns)\tDigital␣Input\tAnalog(V)\tStep␣Output");
    $monitor("%0t\t%0d\t\t%0.2f\t\t%0d", $time, digital_in, analog_value_mV,
        analog_step);

    digital_in = 0; analog_value_mV = (digital_in / 255.0) * (Vref_mV / 1000.0); #10;
    digital_in = 64; analog_value_mV = (digital_in / 255.0) * (Vref_mV / 1000.0); #10;
    digital_in = 128; analog_value_mV = (digital_in / 255.0) * (Vref_mV / 1000.0);
        #10;
    digital_in = 192; analog_value_mV = (digital_in / 255.0) * (Vref_mV / 1000.0);
        #10;
    digital_in = 255; analog_value_mV = (digital_in / 255.0) * (Vref_mV / 1000.0);
        #10;

    #10 $finish;
end
```

endmodule
```

5 XDC Constraints

Use the following XDC mapping (adjust pins by your ZedBoard revision if needed):

```
1
2  ## Clock Input
3
4  set_property PACKAGE_PIN E3 [get_ports clk]
5  set_property IOSTANDARD LVCMOS33 [get_ports clk]
6
7  ## Reset Button (BTN0)
8
9  set_property PACKAGE_PIN T18 [get_ports rstn]
10 set_property IOSTANDARD LVCMOS33 [get_ports rstn]
11
12 ## Switches (SW0-SW7)
13
14 set_property PACKAGE_PIN F22 [get_ports {sw[0]}]
15 set_property PACKAGE_PIN G22 [get_ports {sw[1]}]
16 set_property PACKAGE_PIN H22 [get_ports {sw[2]}]
17 set_property PACKAGE_PIN F21 [get_ports {sw[3]}]
18 set_property PACKAGE_PIN H19 [get_ports {sw[4]}]
19 set_property PACKAGE_PIN H18 [get_ports {sw[5]}]
20 set_property PACKAGE_PIN H17 [get_ports {sw[6]}]
21 set_property PACKAGE_PIN M15 [get_ports {sw[7]}]
22 set_property IOSTANDARD LVCMOS33 [get_ports {sw[*]}]
23
24 ## LEDs (LD0-LD3)
25
26 set_property PACKAGE_PIN T22 [get_ports {led[0]}]
27 set_property PACKAGE_PIN T21 [get_ports {led[1]}]
28 set_property PACKAGE_PIN U22 [get_ports {led[2]}]
29 set_property PACKAGE_PIN U21 [get_ports {led[3]}]
30 set_property IOSTANDARD LVCMOS33 [get_ports {led[*]}]
```

6 Simulation and Results

Run the provided testbench in Vivado or ModelSim to observe printed monitor values and waveforms. Example output (expected):

Digital Input	Analog Voltage (V)	Step Output
0	0.00	0 64
0.83	2 128	1.65
4 192	2.48	6 255
3.30	8	

Include waveform screenshots from Vivado's simulator here (replace the file path below with your actual images):

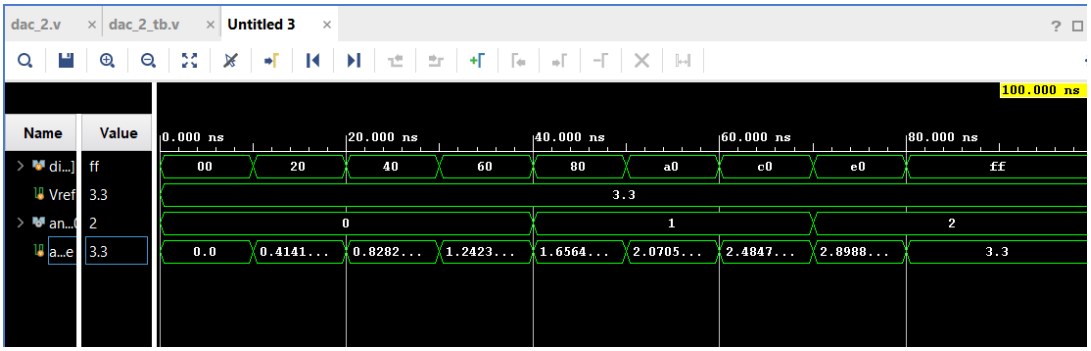

Figure 1: Simulation waveform (replace with your capture).

7 Synthesis and Implementation Notes

- Ensure `dac_top.vis set as the top module in Vivado.`

- `Use explicit-width constants (e.g., 12'd3300) to avoid port-width mismatch warnings.`
- `Avoid declaring variables inside combinational always blocks; declare regs outside the block for synthesis.`

8 Extensions and Future Work

- Replace the step output with a PWM generator and low-pass filter to produce a true analog voltage on an FPGA pin.
- Implement higher resolution (e.g., 10-bit) and increase the number of output LEDs or use an external resistor ladder.
- Add calibration to compensate for reference voltage variations.

9 Conclusion

This project demonstrates a compact and synthesizable threshold-based DAC design using integer arithmetic in Verilog. It is suitable for FPGA demonstrations, classroom labs, and as a base for building a PWM or resistor-ladder DAC.

References

- Xilinx Zynq-7000 SoC Technical Reference Manual
- FPGA4Fun, PWM DAC examples and sigma-delta implementations