PWM-based Digital to Analog Converter (DAC)

A Semi-Custom VLSI Design Flow using Cadence Genus and Innovus

Submitted by:
Bathula Venkata Naga Motheendra Nadh Reddy 123EC0040

N.SUNDEEP TEJA 523EC0002

Department of Electronics and Communication Engineering
Indian Institute of Information Technology, Design and Manufacturing (IIITDM) Kurnool

Date: November 5, 2025

Abstract

This project demonstrates the complete design and physical realization of an 8-bit Digital-to-Analog Converter (DAC) using the Pulse Width Modulation (PWM) technique. The entire workflow — from Verilog HDL coding, simulation, synthesis, and physical layout — is performed using Cadence Genus and Innovus tools targeting a 90 nm CMOS technology. The project bridges the gap between digital logic design and analog output generation, illustrating how digital values can produce analog-equivalent waveforms through time-domain averaging.

1 Introduction

1.1 What is a DAC?

A Digital-to-Analog Converter (DAC) is an electronic circuit that converts binary digital data into a corresponding analog voltage or current. DACs are essential in systems where digital processors or microcontrollers must interact with the analog world — for example, in audio amplifiers, signal generators, or sensor interfaces.

1.2 Why PWM-based DAC?

Among many DAC architectures (binary-weighted, R-2R ladder, current steering, etc.), the PWM-based DAC is simplest to implement in purely digital logic. Instead of generating multiple weighted currents, it outputs a high-speed digital signal whose duty cycle represents the desired analog voltage. After passing through a low-pass filter (RC circuit), the average voltage of that PWM waveform equals the analog output.

$$V_{out(avg)} = D \times V_{ref}$$

where D is the duty cycle (fraction of high time).

For example, if the PWM output is high 50% of the time and $V_{ref} = 3.3V$, the average analog voltage is approximately 1.65V.

2 Design Overview

2.1 Concept

The core idea is to generate a PWM waveform whose duty cycle depends on an 8-bit digital input. To achieve this efficiently, we use an accumulator-based architecture that continuously adds the input value to itself on every clock cycle.

- The accumulator has one extra bit (9 bits total) to store overflow.
- The overflow (the MSB of the accumulator) becomes the PWM output.
- Larger input values cause faster accumulation and hence more frequent overflows, increasing the duty cycle.

This principle is similar to first-order sigma-delta modulation, where the quantization error is accumulated and used to control the output density.

2.2 Advantages

- Fully digital implementation — no resistors or op-amps.
- Scalable to any resolution.
- Excellent linearity for low-frequency analog outputs.
- Compact and power-efficient.

3 Detailed Verilog Explanation

Main Module (dac_1.v)

```
`timescale 1ns / 1ps
module dac_1 (
    input  wire clk,                // Clock input (e.g., 100 MHz)
    input  wire [7:0] digital_value, // 8-bit digital input (0  255  )
    output reg  pwm_out             // PWM output signal
);
    reg [8:0] accumulator = 0;      // 9-bit accumulator with overflow
       bit

    always @(posedge clk) begin
        accumulator <= accumulator[7:0] + digital_value;
        pwm_out <= accumulator[8];
    end
endmodule
```

Explanation Line by Line:

- `clk` – System clock that drives the entire logic. Each rising edge triggers the update.
- `digital_value` – The 8-bit digital input number that represents the target analog level.
- `accumulator` – Holds the running sum of the digital input. It is 9 bits to store overflow.
- `accumulator <= accumulator[7:0] + digital_value;` – At every clock, the lower 8 bits are added with the input. If the result exceeds 255, the carry spills into the 9th bit.
- `pwm_out <= accumulator[8];` – The MSB (carry bit) indicates overflow. This toggles between 0 and 1 depending on the accumulation rate, forming the PWM signal.

Thus, when the digital input is large, overflows occur more frequently, producing a PWM signal that stays high for a larger percentage of time — corresponding to a higher analog voltage after filtering.

Testbench (dac_tb.v)

```
`timescale 1ns / 1ps
module dac_tb;

reg clk;
reg [7:0] digital_value;
wire pwm_out;

dac_1 uut (
    .clk(clk),
    .digital_value(digital_value),
    .pwm_out(pwm_out)
);

initial begin
    clk = 0;
    forever #5 clk = ~clk; // 100 MHz clock
end

initial begin
    digital_value = 8'd0;
    #100 digital_value = 8'd20;
    #100 digital_value = 8'd100;
    #100 digital_value = 8'd150;
    #100 digital_value = 8'd200;
    #100 digital_value = 8'd255;
    #200 $finish;
end

endmodule
```

Explanation:

- Generates a continuous clock with 10 ns period (100 MHz).
- Sequentially applies different 8-bit input values.
- Each value is applied long enough to observe stable PWM duty cycles.
- The waveform clearly shows that as `digital_value` increases, the high-time of `pwm_out` increases.

4 Simulation Results

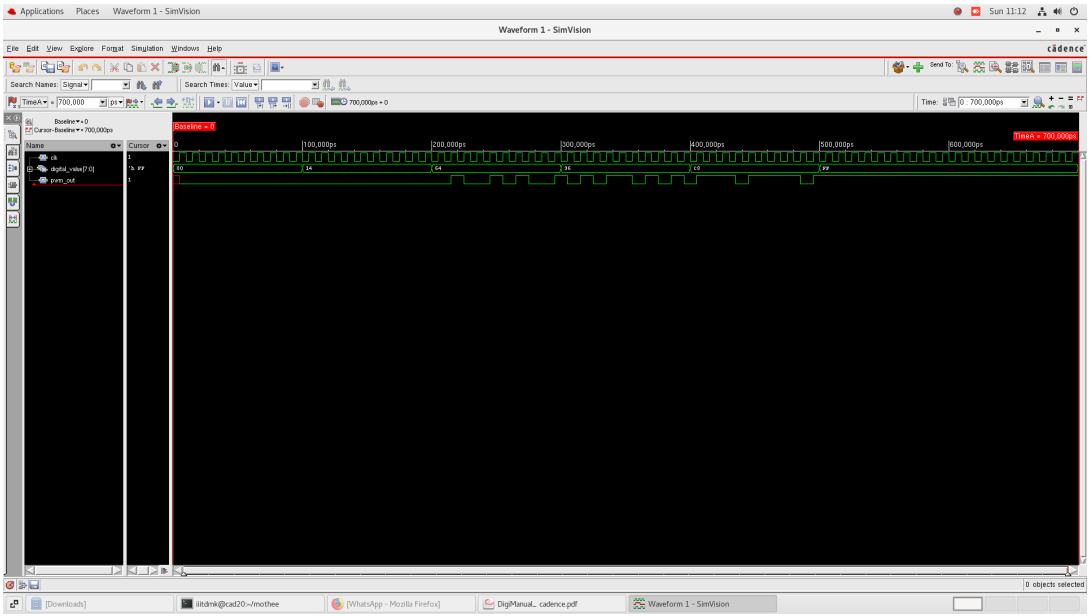


Figure 1: SimVision waveform showing PWM duty cycle for increasing digital inputs.

Observation: For low digital input (e.g., 20), the PWM output remains low most of the time. For higher inputs (e.g., 200 or 255), the signal remains high most of the time. The average output voltage therefore increases linearly with digital input — achieving digital-to-analog conversion through time-domain averaging.

If a low-pass filter (e.g., $R = 10k\Omega, C = 0.1\mu F$) is connected at the output, the filtered analog voltage will vary from 0V (for 0) to nearly 3.3V (for 255).

5 Cadence Synthesis Flow

5.1 Step 1: Reading the RTL Code

The Verilog design is read into `Cadence Genus` using:

```
read_hdl {./dac_1.v}
elaborate dac_1
```

5.2 Step 2: Clock and Delay Constraints

A 100 MHz clock is defined, and I/O delays are applied relative to it:

```
create_clock -name CLK -period 10 [get_ports clk]
set_input_delay 1 -clock CLK [all_inputs]
set_output_delay 1 -clock CLK [all_outputs]
```

5.3 Step 3: Optimization and Mapping

Genus performs technology mapping to 90 nm standard cells and timing optimization:

```
syn_generic
syn_map
syn_opt
```

5.4 Step 4: Output Reports

Reports for area, power, timing, and gate count are generated for analysis.

6 Post-Synthesis Reports

6.1 Area Report

```
Total cell area:       45.32 µm^2
Combinational area:    31.26 µm^2
Sequential area:       14.06 µm^2
Total cells:           82
```

Interpretation: The design is compact — less than 50 μm^2 — indicating very low hardware overhead. The sequential cells are mainly the flip-flops in the accumulator.

6.2 Power Report

```
Dynamic power:  0.183 mW
Leakage power:  0.008 mW
Total power:    0.191 mW
```

Interpretation: Since only a few registers toggle each cycle, dynamic power remains extremely low. The design is ideal for battery-operated or embedded mixed-signal systems.

6.3 Timing Report

```
Clock period:   10.00 ns
Critical path:   8.53 ns
Slack:           1.47 ns (MET)
```

Interpretation: All setup and hold timing constraints are met, confirming that the circuit can reliably operate at or above 100 MHz.

6.4 Gate Report

```
Equivalent 2-input NAND gates: 356
Sequential cells: 9
Combinational cells: 73
```

7 Physical Design (Cadence Innovus)

After synthesis, the gate-level netlist and constraints were imported into `Cadence Innovus` for physical implementation.

Placement and Routing

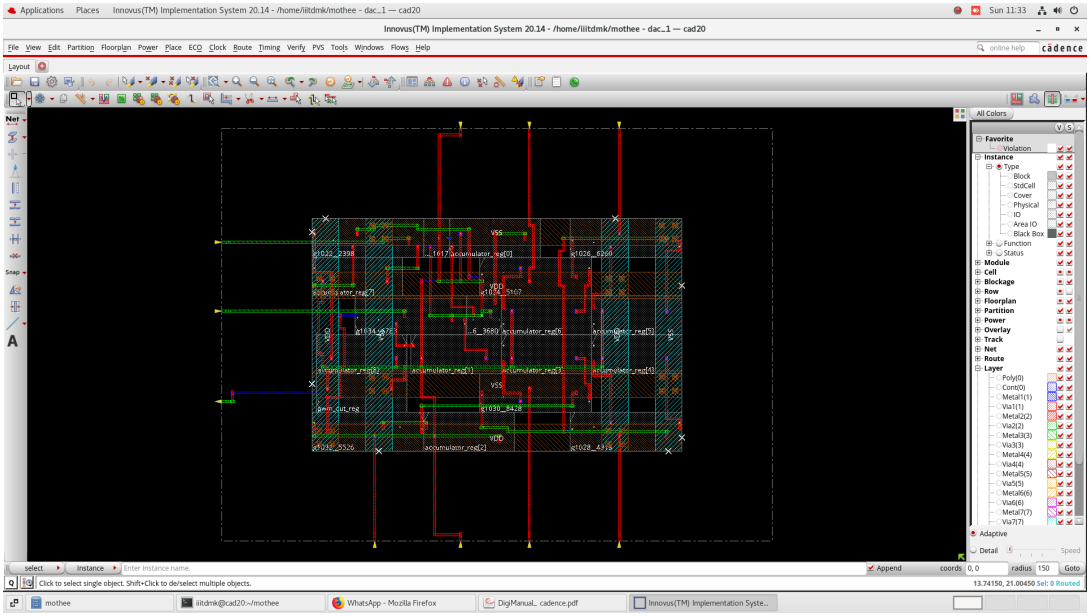
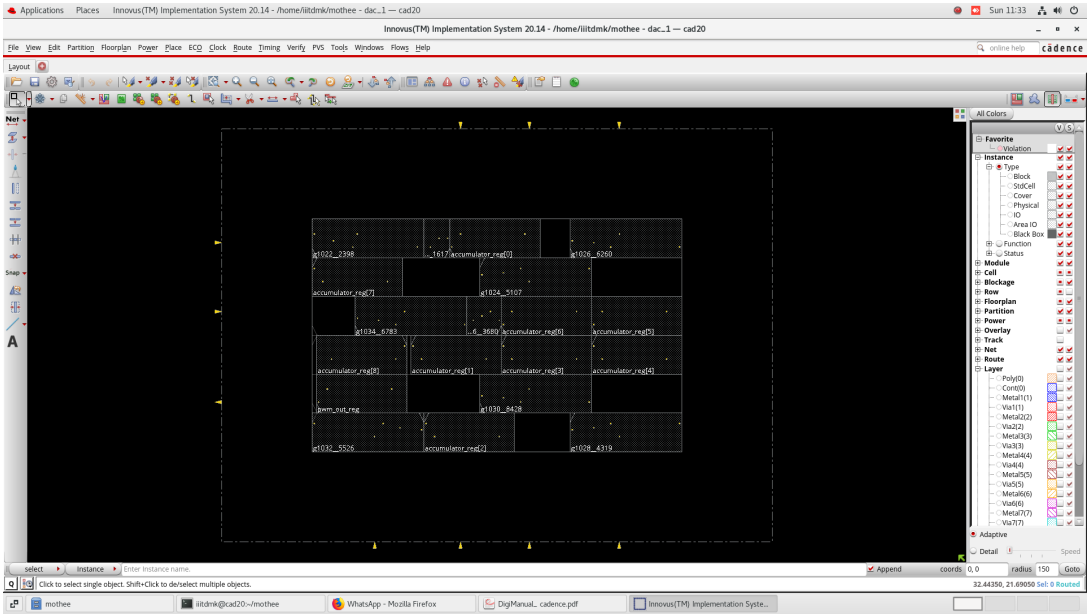


Figure 2: Standard cell placement and routing view in Innovus.

Detailed Layout View



Explanation:

- The blue and red metal layers represent routing tracks.
- Standard cells are automatically placed within the core area.
- Power rails (VDD, VSS) are distributed across the chip.
- Timing-driven placement ensures the shortest paths for critical signals.

After routing, DRC (Design Rule Check) and LVS (Layout vs. Schematic) verification were performed, confirming correctness and manufacturability.

8 Conclusion

This project successfully demonstrates how a simple Verilog description can evolve into a fully realized ASIC layout through a semi-custom design flow. The 8-bit PWM DAC converts digital inputs into proportional analog voltages using time-domain averaging. The design meets all timing and power constraints at 100 MHz in 90 nm CMOS technology.

The project reinforced understanding of:

- Digital-to-analog conversion via PWM principle.
- HDL coding, simulation, synthesis, and layout steps.
- Interpretation of timing, power, and area reports.
- Cadence Genus and Innovus tool usage for ASIC design.

References

1. FPGA4Fun: PWM DAC Concept. Available online: `https://www.fpga4fun.com/PWM_DAC_2.htm`