28Gb/s DSP-DAC-based Transmitter in 45nm with 4-tap FFE Equalizer

Wafeeq Jaleel
Ohio State University, jaleel.5@osu.edu

***Abstract* – This paper presents a transmitter design for PAM4 to achieve a 28Gb/s data rate. The transmitter operates and incorporates 4-tap TX FFE in DSP-DAC. For minimal power consumption, the output driver is implemented as voltage mode and serializer with transmission gate 2:1 MUX. Clock generation divides the input PLL into four phases, which are implemented with Verilog coding.**

Index Terms – 28Gb/s, DSP-DAC, PAM4, quarter rate clock

INTRODUCTION

In rapidly growing high-speed data communication, meeting data rate specifications while reducing signal distortion and minimizing power consumption has become important. This report discusses the design choices of a transmitter to achieve the data rate specification of CEI-28G-VSR. The project implements PAM4 modulation with 4-tap TX FFE in a GPDK 45nm CMOS process.

PAM4 modulation ensures better spectrum efficiency. It has four amplitude levels represented by two bits. This allows the bandwidth to double compared to NRZ modulation. However, the PAM4 eye height is reduced by 9-dB SNR in comparison to NRZ. Since this presents an issue, we need to implement a robust output driver and pre-driver. The design of FFE was implemented with DSP-DAC design as it is more flexible in tweaking the coefficient values than traditional FFE circuits. However, the digital process consumes more power than the traditional FIR filter. To serialize the data, the clock was built at the quarter rate, assuming an external PLL is provided.

The report is divided into three sections: Transmitter Architecture, clock architecture, and full model simulation. The next section transmitter architecture will discuss the design choices and simulations for individual components.

TRANSMITTER ARCHITECTURE

The transmitter architecture has three important sections to it. As shown in Figure 1, the block diagram, DSP-DAC, serializer, and output drivers are used to reduce the ISI of the signal caused by the channel. The equalization technique used in the transmitter is pre-emphasis equalization, which inverts the channel response by using a form of an FIR filter [1]. Even though the serializer's goal is to serialize the data, the project implements it with a Transmission gate to decrease the power consumption. Output driver can provide a 1.0V peak-to-peak differential swing. Design choices and simulation results are discussed in detail in the following sub-sections. All the simulations are run with an input data period of 35.7ps and a clock period of 142.8ps.

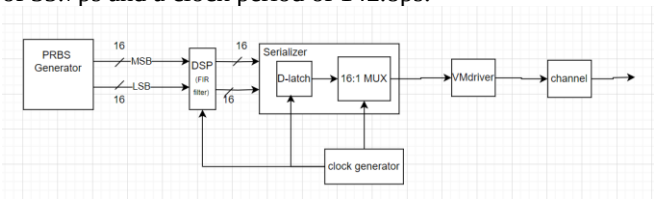

FIGURE 1: BLOCK DIAGRAM

1. OUTPUT DRIVER

The voltage mode driver is a simplified design with lower power consumption to drive the output signal. It is a straightforward implementation compared to current mode drivers. Figure 2 shows the VM driver, which consists of two NMOS and two PMOS. W/L for all the transistors was at 6u/45n. The sizing was selected based on its ability to drive up to VDD or VSS at a 28Gb/s data rate. Figure 3 shows the transient simulation of the output signal. We can observe a slight delay caused by the driver and some overshooting. Another advantage of this driver compared to traditional CMOS drivers is, that it can provide differential outputs.

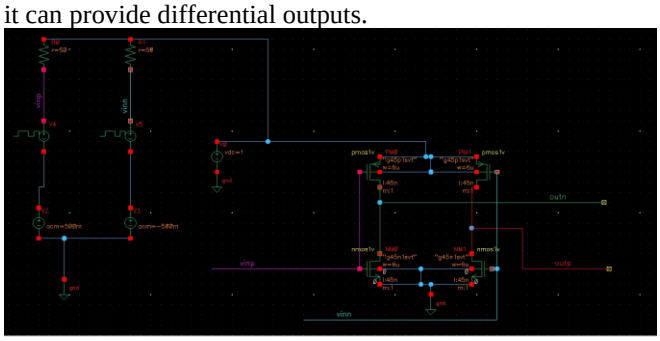

FIGURE 2: VM DRIVER CIRCUIT DESIGN

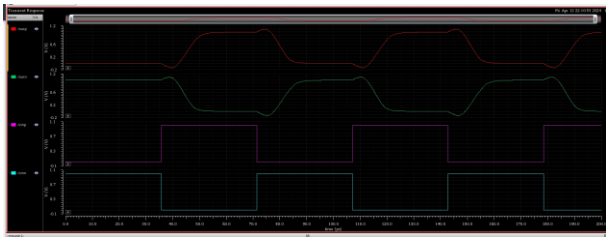

FIGURE 3: TRANSIENT SIMULATION OF VM DRIVER

2. 16:1 SERIALIZER

The serializer was designed with transmission gate 2:1MUX as the base. TG design offers a better signal output compared to the CMOS MUX design. Another architecture that was considered for the MUX was using CML. But CML consumes higher power compared to TG design. TG is known to have lower resistance, which results in minimal signal distortions. It also has less propagation delay and less area compared to conventional CMOS MUX. Figure 4 (a) shows the circuit design of a traditional TG-based 2:1 MUX used and the transient simulation in Figure 4(b).

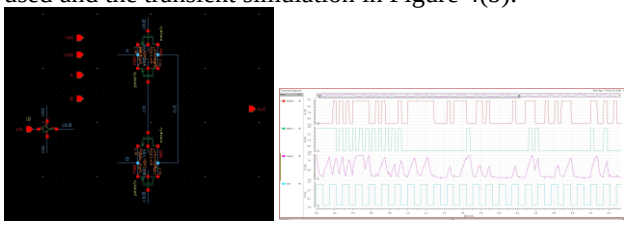

FIGURE 4: 2:1 TG MUX AND ITS TRANSIENT OUTPUT

The 2:1 MUX was scaled to 16:1 MUX. Since the data rate is high, quarter rate clocking was used in serializing. Clock architecture will be discussed in detail in the next section. Figure 5 presents the full serializer design. Each level has a different clock phase, which will ensure the data are correctly aligned. The output of the 16:1 MUX is in Figure 6. Even though the signal is distorted, it captures the input data shape somewhat accurately and is amplified with output drivers.

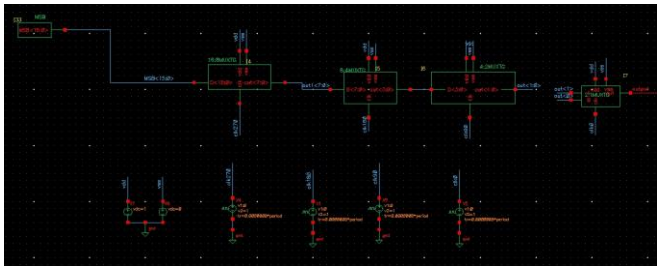

FIGURE 5: 16:1 MUX DESIGN

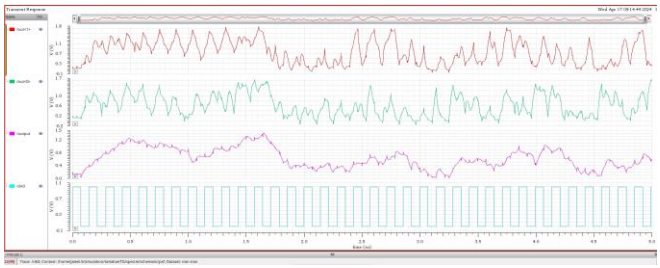

FIGURE 6: 16:1 MUX TRANSIENT OUTPUT

3. DSP-DAC

DSP-DAC is more flexible in covering many numbers of taps. However, the downside is it consumes more power than traditional delay-based transmitters [1]. Since the FFE only has 4 tap resolution, the effect on power is lower. The FFE filter is modeled after a conventional FIR filter. The block diagram is shown below in Figure 10 and detailed code snippets in Figure 7-9. The delay block will delay it by one UI and apply the coefficient. 32-bit precision, as recommended by IEEE standards, is used for the floating-point coefficient. Then all the signals are added to result in a 32-bit output. This is then serialized to get a one-bit signal.

```
module DSP(
    input wire clk,
    input wire reset,
    input wire in_data,
    output reg out_data
);

always @(posedge clk or posedge reset) begin
    if (reset) begin
        out_data <= 1'b0; // Reset output
    end else begin
        #0.25; // Unit delay
        out_data <= in_data; // Assign input to output after one time unit
    end
end

endmodule
```

FIGURE 7: ONE UNIT DELAY VERILOG CODE

```
module \4inputmultiplier  (
    input wire clk,
    input wire rst,
    input wire signal1,
    input wire signal2,
    input wire signal3,
    input wire signal4,
    output reg [31:0] result
);

// Define internal floating-point number
reg [31:0] pre2 = 32'hbe4ccccd;
reg [31:0] pre1 = 32'hbe4ccccd;
reg [31:0] main = 32'h40a00000;
reg [31:0] post1 = 32'h00000000;

reg [31:0] pre2oneBfraction;
reg [31:0] pre2Floatfraction;

reg [31:0] pre1oneBfraction;
reg [31:0] pre1Floatfraction;

reg [31:0] mainoneBfraction;
reg [31:0] mainFloatfraction;

reg [31:0] postoneBfraction;
reg [31:0] postFloatfraction;

reg [31:0] pre2R, pre1R, mainR, postR;
```

FIGURE 8: COEFFICIENT MULTIPLIER AND ADDER CODE 1

Figure 11 shows the output result of the adder. It produced the expected value. Out1 to Out4 are the delayed signals and the mult1 to mult4 gives the output of each tap after the multiplication and addition.

```
always @(posedge clk or posedge rst) begin
    if (rst) begin
        result <= 0;
        pre2oneBfraction <= 0;
        pre2Floatfraction <= 0;

        pre1oneBfraction <= 0;
        pre1Floatfraction <= 0;

        mainoneBfraction <= 0;
        mainFloatfraction <= 0;

        postoneBfraction <= 0;
        postFloatfraction <= 0;

    end else begin

        pre2oneBfraction <= {signal1, {30{1'b0}}};
        pre2Floatfraction <= pre2[22:0];

        pre1oneBfraction <= {signal2, {30{1'b0}}};
        pre1Floatfraction <= pre1[22:0];

        mainoneBfraction <= {signal3, {30{1'b0}}};
        mainFloatfraction <= main[22:0];

        postoneBfraction <= {signal4, {30{1'b0}}};
        postFloatfraction <= post1[22:0];


        pre2R <= (pre2oneBfraction * pre2Floatfraction) >>> 22;
        pre1R <= (pre1oneBfraction * pre1Floatfraction) >>> 22;
        mainR <= (mainoneBfraction * mainFloatfraction) >>> 22;
        postR <= (postoneBfraction * postFloatfraction) >>> 22;

        assign result = pre2R+pre1R+mainR+postR;
    end
end

endmodule
```

FIGURE 9: COEFFICIENT MULTIPLIER AND ADDER CODE 2

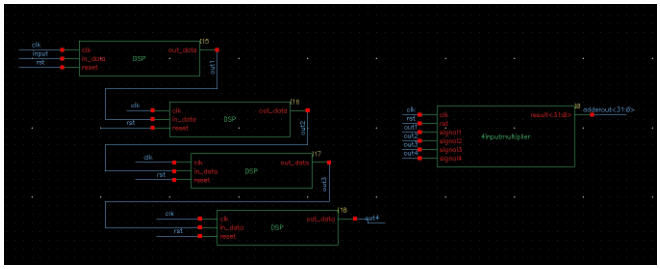

FIGURE 10: FINAL BLOCK DIAGRAM OF FFE

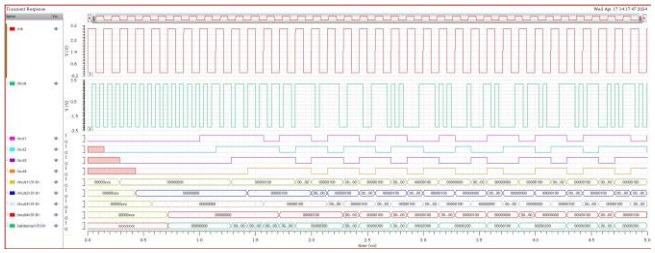

FIGURE 11: TRANSIENT SIMULATION OF FFE

On the other hand, DAC is based on the VM driver topology. It simply converts the digital signals to analog output. This was chosen because of its simplicity and high driving strength. The circuit design of the VM is the same as the output driver shown in Figure 3.

4. INPUT PRBS GENERATOR AND PAM4 MODULATION

As mentioned in the introduction, using PAM4 modulation helps with achieving a higher data rate. Firstly, the VPRBS source provides different random patterns depending on the seed selected. MSB and LSB are generated separately to ensure design flexibility down the line. Figure 12(a) shows the MSB block and 12(b) highlights the transient output of MSB and LSB combined in PAM4 model.

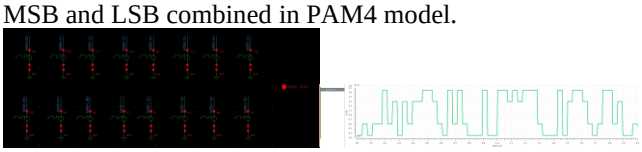

FIGURE 12: INPUT GENERATOR AND TRANSIENT OUTPUT

Since the FFE equalizer was implemented in DSP, it was easier to convert the signals in the PAM4 mod using Verilog. Figure 13 shows the code for one lane and the output in Figure 14. This output is fed to the input of the FFE filter.

```
module PAM4Mod(
    input wire clk,
    input wire rst_n,
    input wire prbs1,
    input wire prbs2,
    output reg [1:0] pam4
);

// PAM4 modulation
always @ (posedge clk or negedge rst_n) begin
    if (~rst_n) begin
        pam4 <= 2'b00;
    end else begin
        case ({prbs1, prbs2})
            2'b00: pam4 <= 2'b00;   // 00: +3
            2'b01: pam4 <= 2'b01;   // 01: +1
            2'b10: pam4 <= 2'b10;   // 10: -1
            2'b11: pam4 <= 2'b11;   // 11: -3
            default: pam4 <= 2'b00;
        endcase
    end
end

endmodule
```

FIGURE 13: PAM4 MODULATOR

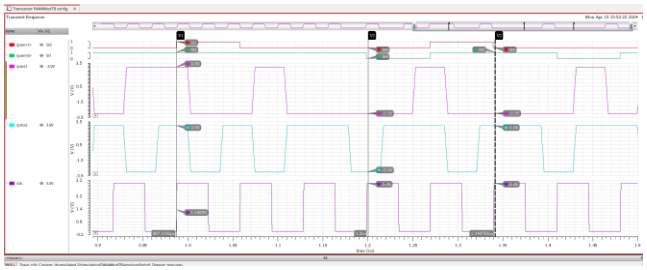

FIGURE 14: TRANSIENT OUTPUT OF PAM4 MODULATOR

CLOCK ARCHITECTURE

The external PLL was generated by vpulse with a period of 142.8ps, which is four times the UI of the data rate. The quarter-rate clock generation architecture consists of a series of delays to move the clock by 90 degrees. It was implemented in the digital domain to get better clocking times. The Verilog code is shown in Figure 15. As shown each clock is delayed by 0.25 of the clock periods. The output of the clocks from the digital block is attached to a VM driver to convert it into an analog signal as shown in Figure 16.

Figure 17 shows the output, and you can observe a slight delay. But the frequency is halved as expected.

```
module clockgenVerilog(
    input wire clk,
    output wire clk_90,
    output wire clk_180,
    output wire clk_270
);

reg clk_q1 = 0;
reg clk_q2 = 0;
reg clk_q3 = 0;

// 90 degrees phase shift
always @(posedge clk)
    clk_q1 <= ~clk_q1;

// 180 degrees phase shift
always @(posedge clk_q1)
    clk_q2 <= ~clk_q2;

// 270 degrees phase shift
always @(posedge clk_q2)
    clk_q3 <= ~clk_q3;

assign clk_90 = clk_q1;
assign clk_180 = clk_q2;
assign clk_270 = clk_q3;

endmodule
```

FIGURE 15: VERILOG CODE FOR CLOCK GENERATION

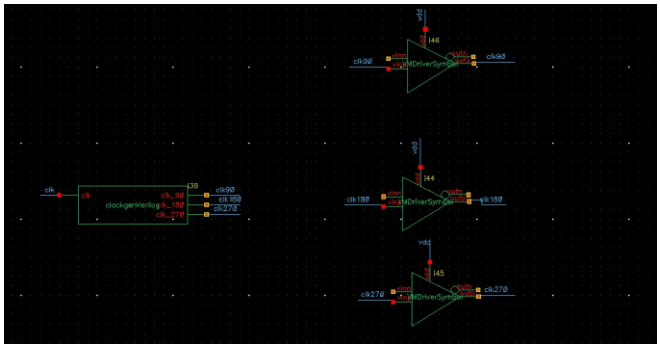

FIGURE 16: CLOCK GENERATOR CIRCUIT DESIGN

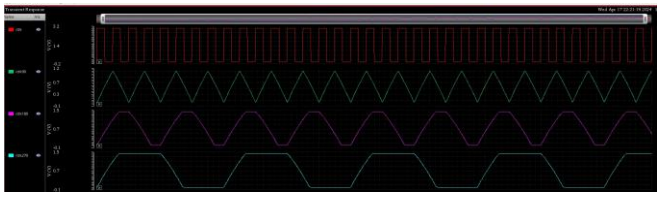

FIGURE 17: TRANSIENT OUTPUT OF CLOCK GENERATOR

IMPLEMENTATION AND MEASUREMENTS

The full test bench is shown in Figure 18. For the test bench, I used vpulse to generate the quarte rate clocks, to ensure the system does not fail due to clocking. The eye diagram of the full design is shown in Figure 19, with a VM of less than 0.100mV. The circuit does not work as intended when all the elements are put together. Reducing the data rate was not helping the eye margin either. Figure 20 shows an eye margin for 2.8GB/s which is still the same as before.

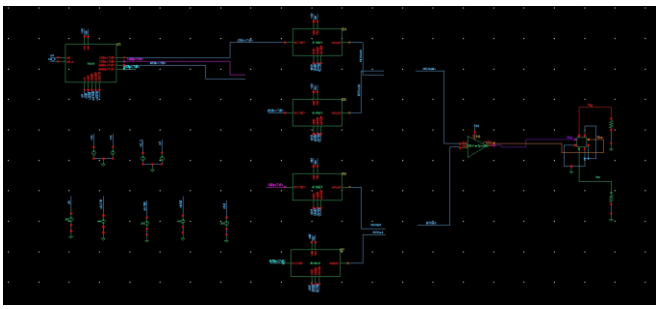

FIGURE 18: FULL DESIGN TEST BENCH

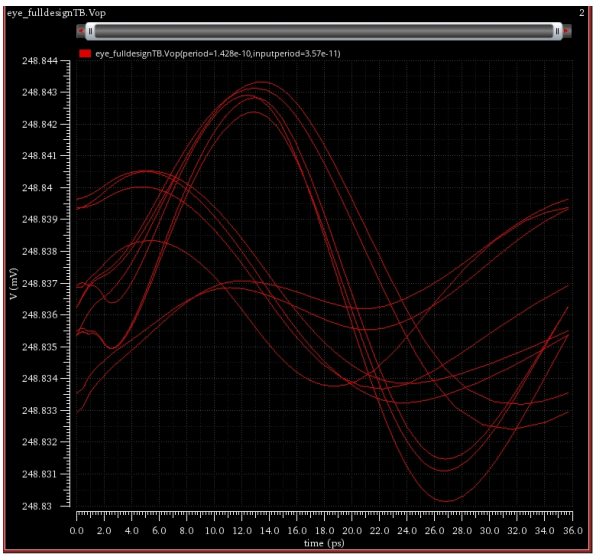

FIGURE 19: EYE DIAGRAM AT 28GB/S

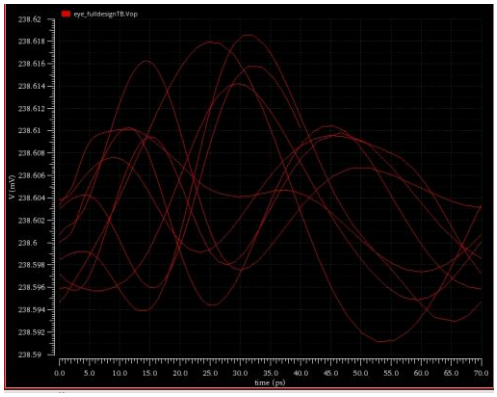

FIGURE 20: EYE DIAGRAM AT 2.8GB/S

The circuit may have been failing mainly because of the DSP system I designed. Even though the blocks functioned

individually, they were dropping voltage, when connected to other circuit elements. At that time the DAC produced great results, but I should have designed the DAC with a robust topology. Another addition to the circuit would be having a pre-driver. It could contribute to extending the bandwidth of the signal.

CONCLUSION

This report presents a DSP-DAC-based transmitter that aims to work at a 28Gb/s data rate in PAM4 modulation. The input signal was generated using VRBS, separating the LSB and MSB with 8 input lanes. The FFE equalizer is implemented in the digital domain with a traditional FIR filter. This was implemented in Verilog and a VMdriver is used for DAC. For serialization, a quarter-rate clock was used. The 16:1 serializer was built from a 2:1 transmission gate MUX. Finally, the output driver is modeled with a traditional voltage mode driver. The voltage margin of the output is 0.1mV. It does not meet the intended requirements.

In future implementations, I would consider using driver capacitance in the serializer and output driver circuitry for feed-forward charge injection. Also, using an SST driver for FFE implementation would have made them less susceptible to DAC conversion noise.

REFERENCES

[1] E. Groen et al., "10-to-112-Gb/s DSP-DAC-Based Transmitter in 7-nm FinFET With Flex Clocking Architecture," in IEEE Journal of Solid-State Circuits, vol. 56, no. 1, pp. 30-42, Jan. 2021, doi: 10.1109/JSSC.2020.3036981.