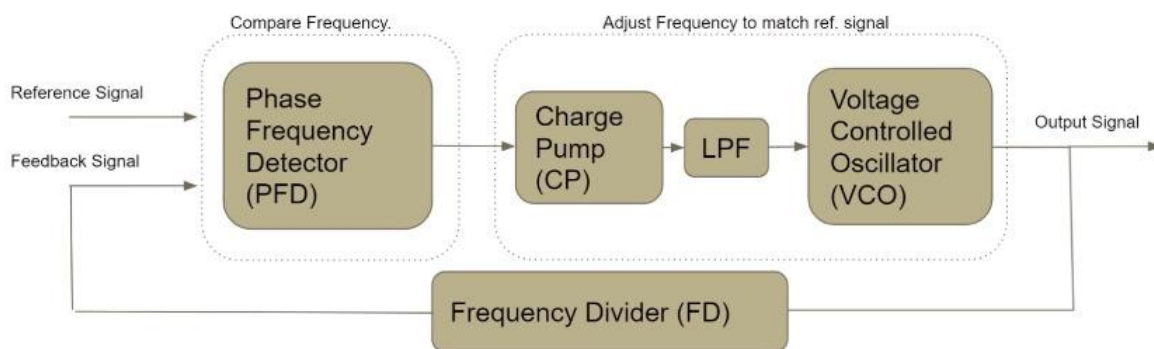


## PLL

### Phase Locked Loop

The pll module is a phase-locked loop that generates a stable clock (CLK) for the RISC-V core.



The above diagram shows basic block diagram of the PLL to be implemented. The functionality of PLL can be described as two processes.

- Comparing frequency of reference and output (PFD)
- Adjusting frequency to match reference signal (CP and VCO)

### Phase Frequency Detector

The phase frequency detector (PFD) is responsible for comparing the reference signal given as input and the output signal. Then it should produce output which clearly shows the difference of phase. This phase difference is not just in terms of magnitude but it should also show whether the output is leading or lagging behind the reference. The output of PFD is in digital form.

### Charge pump

The CP converts the digital output from PFD to an analog signal. This analog signal is what would control the VCO. The analog output from CP is passed through a low pass filter before connecting to the VCO. This low pass filter can help smoothen the signal in addition to stabilizing the feedback loop.

### VCO and Frequency Divider

Voltage controlled oscillators are the actual parts which produce alternating digital clock signal. The frequency of this clock signal can be controlled by input voltage, hence the name. VCO can be implemented using simple inverters. A PLL with a frequency

divider on its feedback loop is called a clock multiplier PLL. Such a PLL can make clock signals which are multiples of the reference signals.

#### Source Code

```
module avsdpll (
    output reg CLK,
    input wire VCO_IN,
    input wire ENb_CP,
    input wire ENb_VCO,
    input wire REF
);
    real period, lastedge, refpd;

    initial begin
        lastedge = 0.0;
        period = 25.0; // 25ns period = 40MHz
        CLK <= 0;
    end

    // Toggle clock at rate determined by period
    always @(CLK or ENb_VCO) begin
        if (ENb_VCO == 1'b1) begin
            #(period / 2.0);
            CLK <= (CLK === 1'b0);
        end
        else if (ENb_VCO == 1'b0) begin
            CLK <= 1'b0;
        end
    end
```

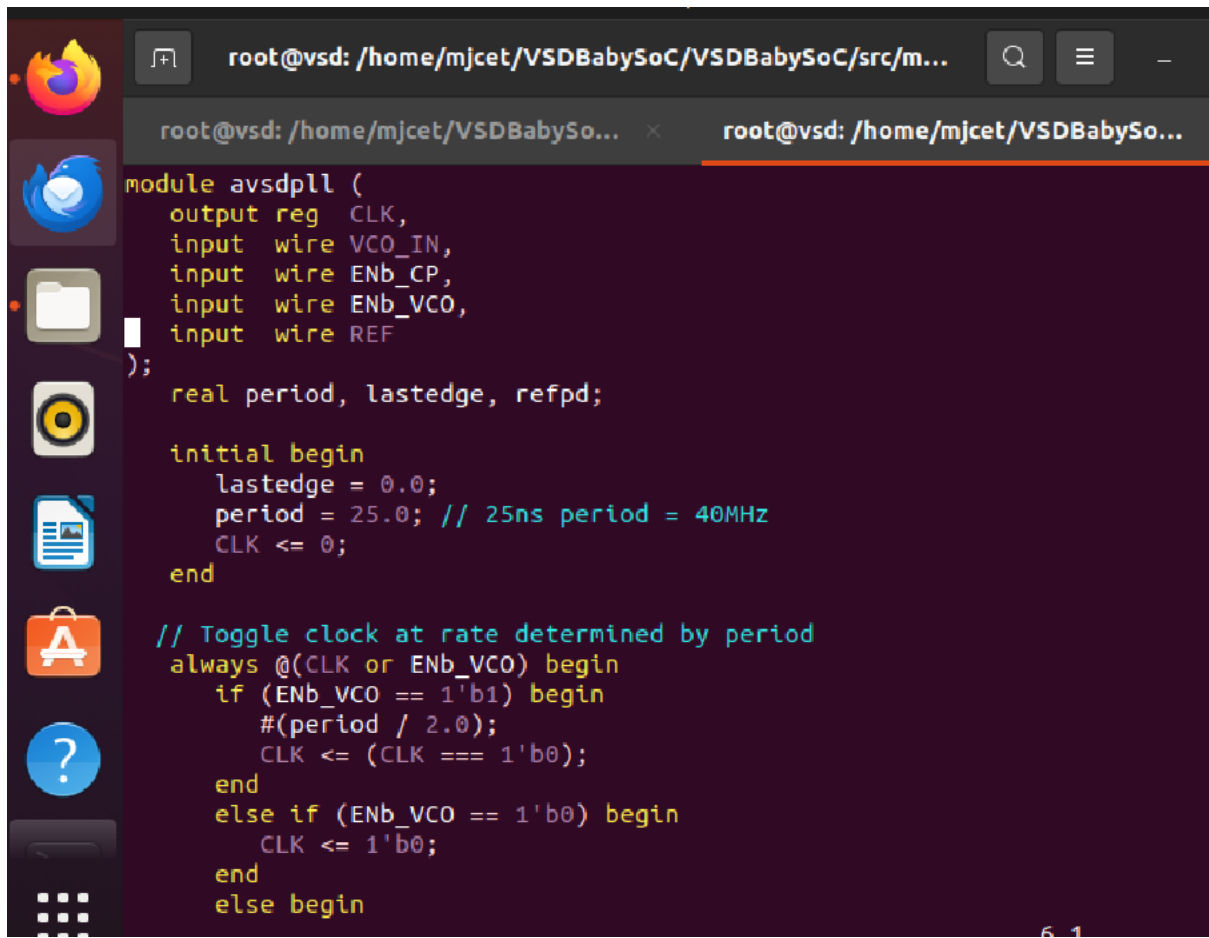
```
    else begin
        CLK <= 1'bx;
    end
end

// Update period on every reference rising edge
always @(posedge REF) begin
    if (lastedge > 0.0) begin
        refpd = $realtime - lastedge;

        // Adjust period towards 1/8 the reference period
        //period = (0.99 * period) + (0.01 * (refpd / 8.0));

        period = (refpd / 8.0) ;
    end

    lastedge = $realtime;
end
endmodule
```

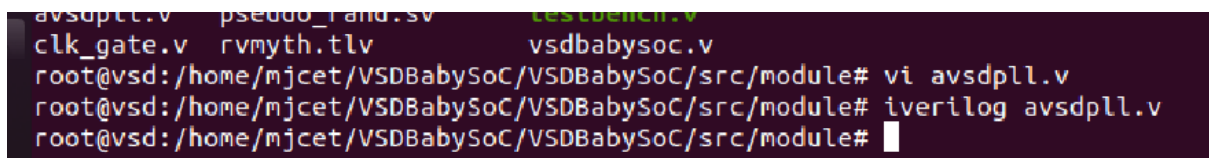


```
root@vsd: /home/mjcet/VSDBabySoC/VSDBabySoC/src/m...
root@vsd: /home/mjcet/VSDBabySoC/VSDBabySoC/src/m...

module avsdpll (
    output reg CLK,
    input wire VCO_IN,
    input wire ENb_CP,
    input wire ENb_VCO,
    input wire REF
);
    real period, lastedge, refpd;

    initial begin
        lastedge = 0.0;
        period = 25.0; // 25ns period = 40MHz
        CLK <= 0;
    end

    // Toggle clock at rate determined by period
    always @(CLK or ENb_VCO) begin
        if (ENb_VCO == 1'b1) begin
            #(period / 2.0);
            CLK <= (CLK === 1'b0);
        end
        else if (ENb_VCO == 1'b0) begin
            CLK <= 1'b0;
        end
        else begin
            // ...
        end
    end
end
```



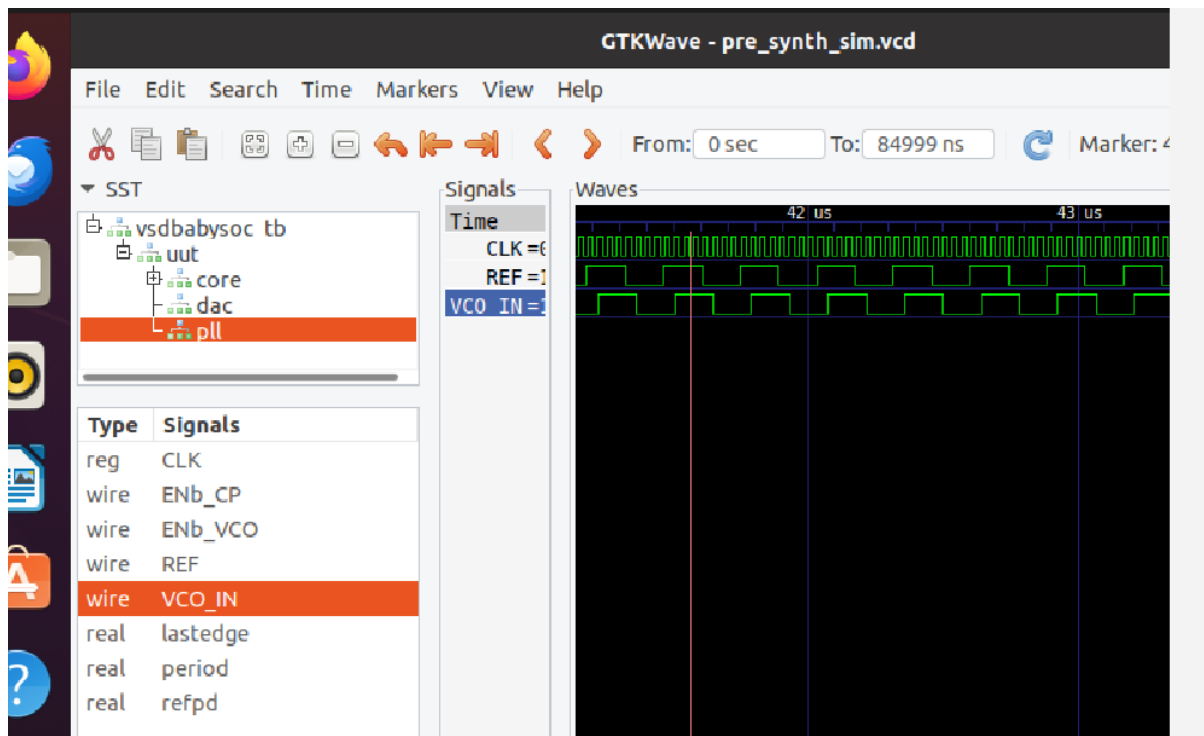
```
avsdpll.v pseudo_rand.sv testbench.v
clk_gate.v rvmyth.tlv vsdbabysoc.v
root@vsd: /home/mjcet/VSDBabySoC/VSDBabySoC/src/module# vi avsdpll.v
root@vsd: /home/mjcet/VSDBabySoC/VSDBabySoC/src/module# iverilog avsdpll.v
root@vsd: /home/mjcet/VSDBabySoC/VSDBabySoC/src/module#
```

## Simulation of PLL

### Pre-Synthesis Simulation

Run the following command to perform a pre-synthesis simulation:

```
iverilog -o output/pre_synth_sim/pre_synth_sim.out -DPRE_SYNTH_SIM \
    -I src/include -I src/module \
    src/module/testbench.v src/module/vsdbabysoc.v
cd output/pre_synth_sim
./pre_synth_sim.out
```



### Explanation:

- -DPRE\_SYNTH\_SIM: Defines the PRE\_SYNTH\_SIM macro for conditional compilation in the testbench.
- The resulting pre\_synth\_sim.vcd file can be viewed in GTKWave.

### Viewing Waveform in GTKWave

After running the simulation, open the VCD file in GTKWave: `gtkwave output/pre_synth_sim/pre_synth_sim.vcd`

### Post-Synthesis Simulation

To run a post-synthesis simulation, use:

```
iverilog -o output/post_synth_sim/post_synth_sim.out -DPOST_SYNTH_SIM \
-I src/include -I src/module \
src/module/testbench.v output/synthesized/vsdbabysoc.synth.v
cd output/post_synth_sim
./post_synth_sim.out
```