PES UNIVERSITY

100 Feet Ring Road, Banashankari Stage III, Dwaraka Nagar, Bengaluru, Karnataka 560085

Mini project submitted in partial fulfillment of Requirement for the award of engineering in third semester computer science and engineering

Digital Design & Computer Organization Lab
Title:- Parallel-in-Parallel-Out and Parallel-in-Serial-Out

Submitted By:-
Vishal S Salanke
PES1UG21CS843

UNDER THE GUIDANCE OF
Dr. NAGEGOWDA K S

Group with :-

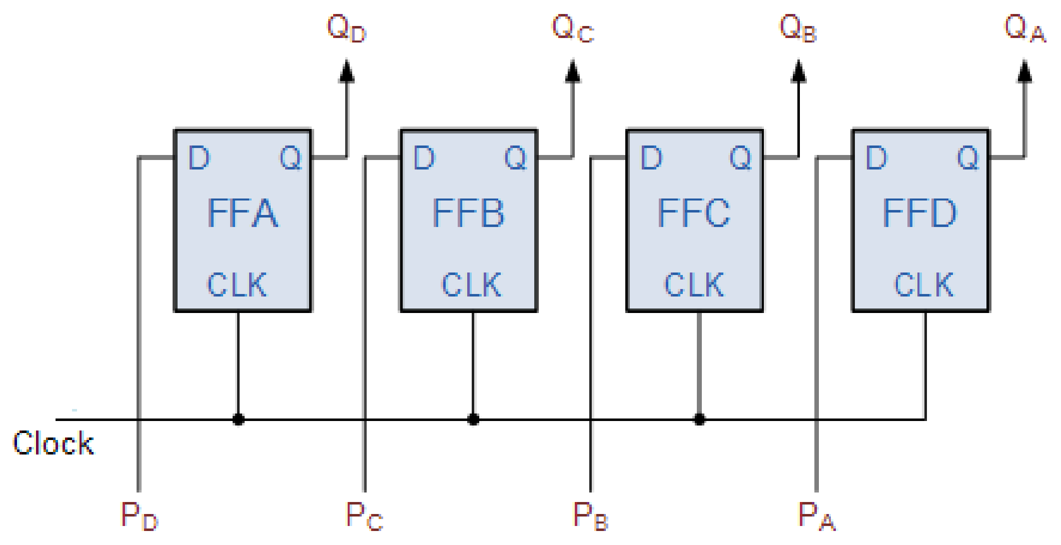
Hari Prasad G	PES1UG21CS811
Pavan D N	PES1UG21CS827
Venkatesh A	PES1UG21CS840
Vishal S Salanke	PES1UG21CS843

Abstract

The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data. Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format.

Parallel-in-Parallel-Out

Circuit Diagram:-



Module Code:-

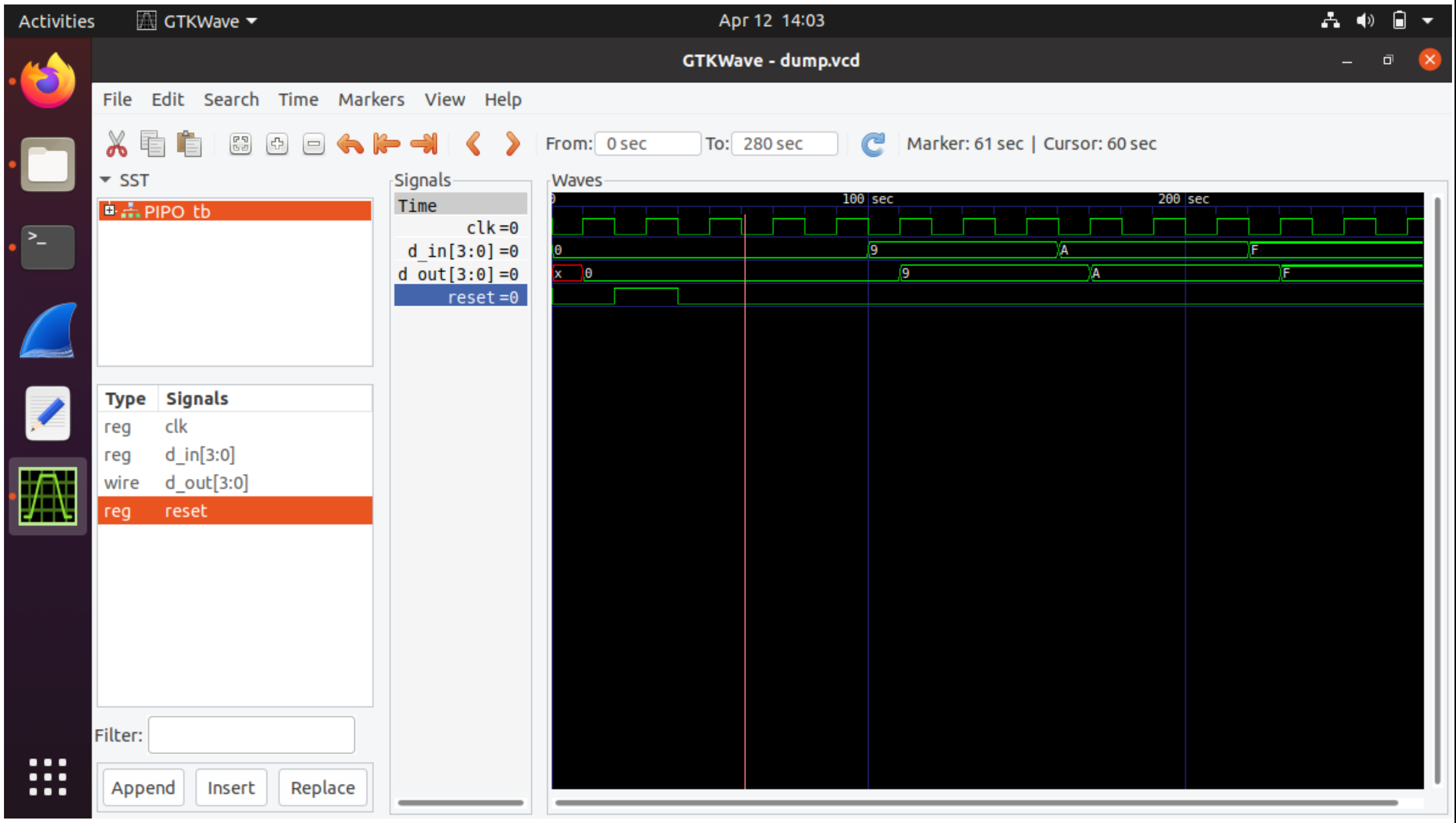
```
module invert (input wire i, output wire o);
   assign o = !i;
endmodule

module and2 (input wire i0, i1, output wire o);
  assign o = i0 & i1;
endmodule

module df (input wire clk, in, output wire d_out);
  reg df_out;
  always@(posedge clk) df_out <= in;
  assign d_out = df_out;
endmodule

module dfr (input wire clk, reset, in, output wire d_out);
  wire reset_, df_in;
  invert invert_0 (reset, reset_);
  and2 and2_0 (in, reset_, df_in);
  df df_0 (clk, df_in, d_out);
endmodule

module PIPO(input wire clk, reset,input wire[3:0] d_in, output wire[3:0]
d_out);
dfr d0(clk,reset,d_in[0],d_out[0]);
dfr d1(clk,reset,d_in[1],d_out[1]);
dfr d2(clk,reset,d_in[2],d_out[2]);
dfr d3(clk,reset,d_in[3],d_out[3]);
endmodule
```

Test_Bench Code:-

```
module PIPO_tb;
reg clk, reset;
reg[3:0] d_in;
wire[3:0] d_out;
PIPO p1(clk,reset,d_in,d_out);

initial
begin
$dumpfile("dump.vcd");
$dumpvars(0, PIPO_tb);
clk=0;
reset=0;
d_in=4'b0000;
#20 reset=1;
#20 reset=0;
#60 d_in= 4'b1001;
#60 d_in= 4'b1010;
#60 d_in= 4'b1111;
```

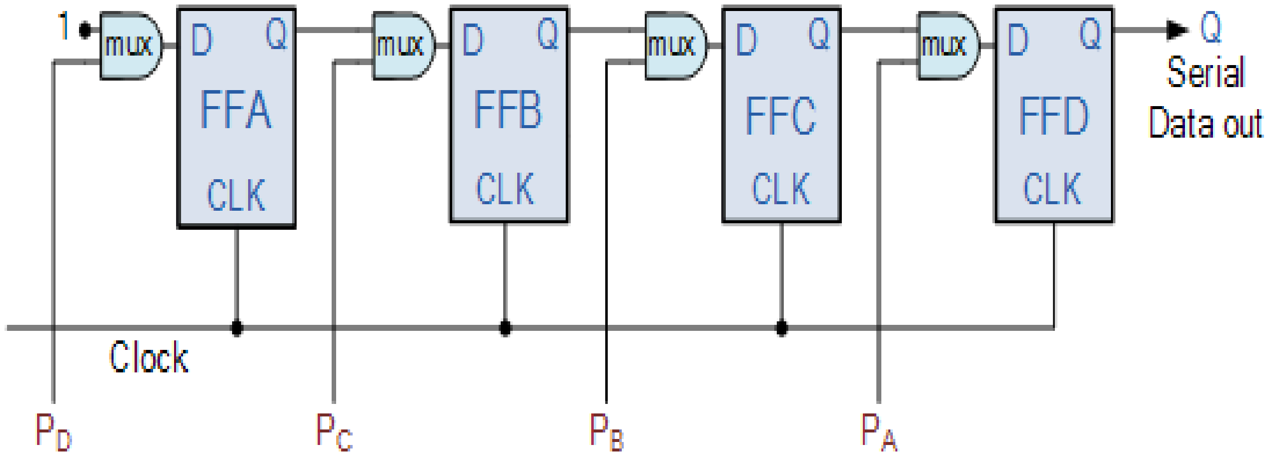
```
#60 $finish;
end
always #10 clk = ~clk;

endmodule
```

OUTPUT



Parallel-in-Serial-Out

Circuit Diagram:-

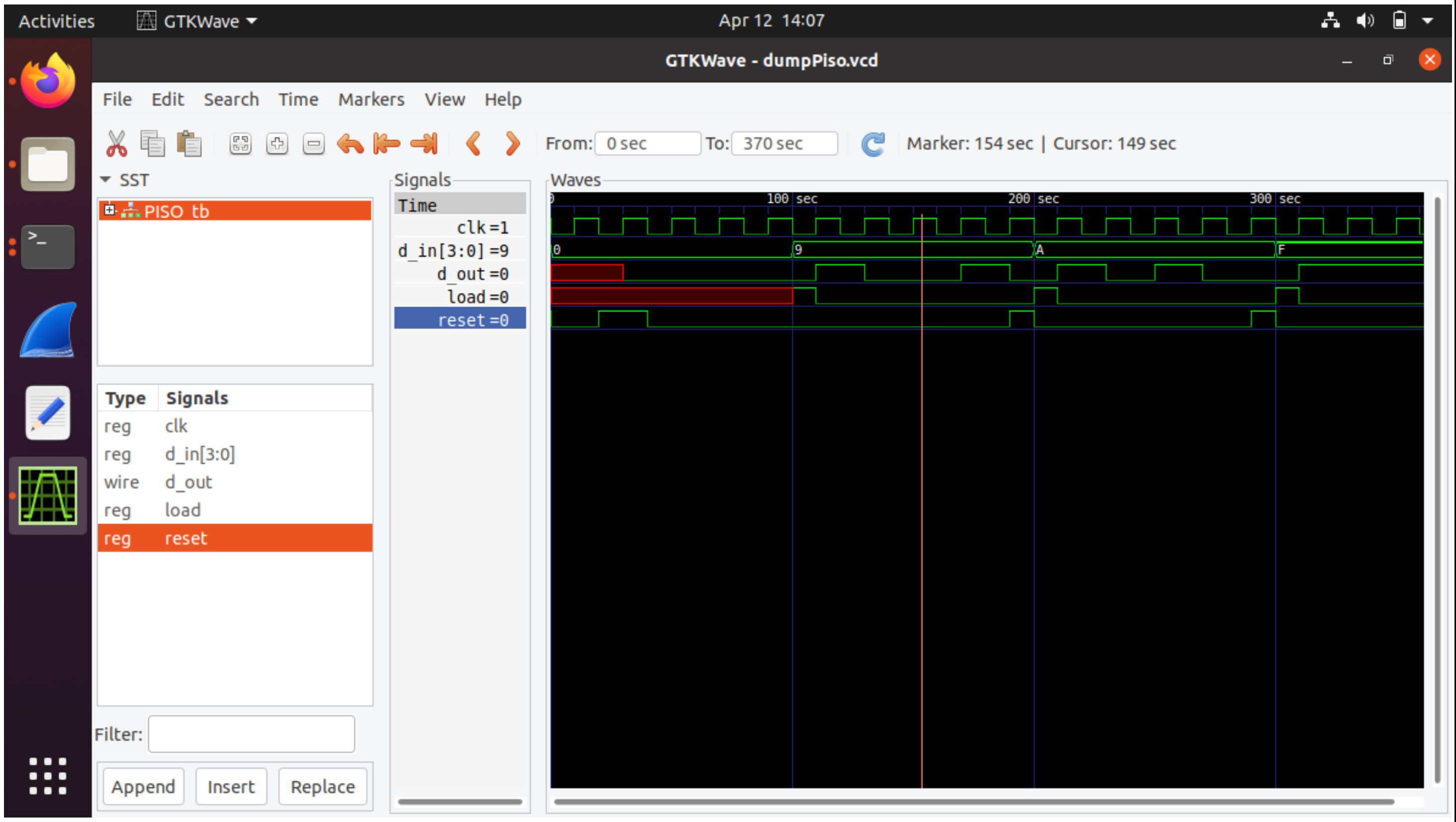


Module Code:-

```
module invert (input wire i, output wire o);
   assign o = !i;
endmodule

module and2 (input wire i0, i1, output wire o);
  assign o = i0 & i1;
endmodule

module mux2 (input wire i0, i1, j, output wire o);
  assign o = (j==0)?i0:i1;
endmodule

module df (input wire clk, in, output wire d_out);
reg df_out;
always@(posedge clk) df_out <= in;
assign d_out = df_out;
endmodule
```

```
module dfr (input wire clk, reset, in, output wire d_out);
wire reset_, df_in;
invert invert_0 (reset, reset_);
and2 and2_0 (in, reset_, df_in);
df df_0 (clk, df_in, d_out);
endmodule

module PISO(input wire clk, reset, load, input wire[3:0] d_in, output wire
d_out);
wire trms[6:0];
mux2 m0(1'b1, d_in[0], load, trms[0]);
dfr d0(clk,reset,trms[0],trms[1]);
mux2 m1(trms[1], d_in[1], load, trms[2]);
dfr d1(clk,reset,trms[2],trms[3]);
mux2 m2(trms[3], d_in[2], load, trms[4]);
dfr d2(clk,reset,trms[4],trms[5]);
mux2 m3(trms[5], d_in[3], load, trms[6]);
dfr d3(clk,reset,trms[6],d_out);
endmodule
```

OUTPUT



```
clk=1
d_in[3:0]=9
d_out=0
load=0
reset=0
```

Type	Signals
reg	clk
reg	d_in[3:0]
wire	d_out
reg	load
reg	reset