Title:

Introduction to Design of Pulse Counters for Distance and Pulse Width/Rate Measurements.

Introduction:

The main objective of this experiment is to familiar with pulse counter, their applications in defense and bioengineering for distance measurement and pulse monitoring of heart, designing pulse counter and implementing it in FPGA or CPLD.

Theory and Methodology:

Apparatus:

1. A Windows-based (XP or 7 or 10) PC with standard word processors (i.e. Microsoft Office) and PDF readers (i.e. Adobe Acrobat Reader/Writer, Foxit Reader/Phantom) installed.
2. Necessary EDA tools such Active-HDL, Precision RTL, ISE WebPack.

Precautions:

A PC with a standard Anti-Virus program installed was used.

Simulation and Measurement:

//Half Period Measurement

//FSM:

```
timeunit 1ns;
timeprecision 1ps;

extern module FSM (
input logic PULSE,
input logic SYS_CLK,
input logic A_RESET,

output logic INC,
output logic SCLR,
output logic LOAD
);

module FSM(.*);

logic [2:0] P_STATE, N_STATE;
always_comb
begin:NSOL

begin: NSL
```

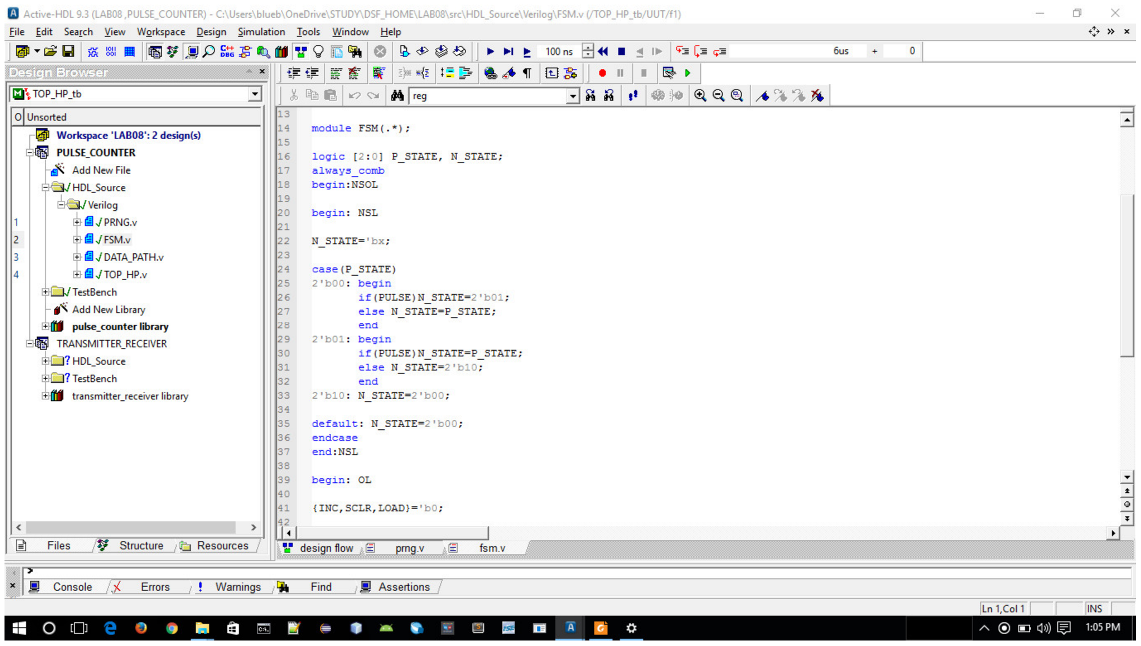
```
N_STATE='bx;

case(P_STATE)
2'b00: begin

if(PULSE)N_STATE=2'b01;
 else N_STATE=P_STATE;
 end
2'b01: begin

if(PULSE)N_STATE=P_STATE;
 else N_STATE=2'b10;
 end
2'b10: N_STATE=2'b00;

default: N_STATE=2'b00;
endcase
end:NSL

begin: OL

{INC,SCLR,LOAD}='b0;

case(P_STATE)
2'b00: SCLR=1'b1;
2'b01: INC=1'b1;
2'b10: LOAD=1'b1;
default:;
endcase
end: OL

end:NSOL

always@(posedge SYS_CLK, posedge A_RESET)
begin:PSR

if(A_RESET)P_STATE<=2'b00;
  else
 P_STATE<=N_STATE;
end:PSR

endmodule:FSM
```

//DATA_PATH:

```
timeunit 1ns;
timeprecision 1ps;

module DATA_PATH ( input logic SCLR, logic INC, logic LOAD, logic SYS_CLK, output logic
[4:0] OUT_REG);

logic[4:0] COUNT;
always_ff@(posedge SYS_CLK)
begin:COUNTER
if(SCLR)COUNT<='b0;
else if(INC)COUNT<=COUNT+1;
end:COUNTER

always_ff@(posedge SYS_CLK)
begin: STORE

if(LOAD)OUT_REG<=COUNT;
end: STORE
endmodule :DATA_PATH
```

//Top LEVEL

```
timeunit 1ns;
timeprecision 1ps;

module TOP_HP ( input logic PULSE, input logic SYS_CLK, input logic A_RESET, output logic
[4:0] OUT_REG);

logic INC, SCLR, LOAD;

FSM f1(.*);
DATA_PATH d1 (.*);

endmodule
```



Active-HDL 9.3 (LAB08 ,PULSE_COUNTER) - C:\Users\blueb\OneDrive\STUDY\DSF_HOME\LAB08\src\HDL_Source\Verilog\FSM.v (/TOP_HP_tb/UUT/f1)

```
module FSM(.*);

logic [2:0] P_STATE, N_STATE;
always_comb
begin:NSOL

begin: NSL

N_STATE='bx;

case(P_STATE)
2'b00: begin
       if(PULSE)N_STATE=2'b01;
       else N_STATE=P_STATE;
       end
2'b01: begin
       if(PULSE)N_STATE=P_STATE;
       else N_STATE=2'b10;
       end
2'b10: N_STATE=2'b00;

default: N_STATE=2'b00;
endcase
end:NSL

begin: OL

{INC,SCLR,LOAD}='b0;
```

Figure : SOURCE CODE

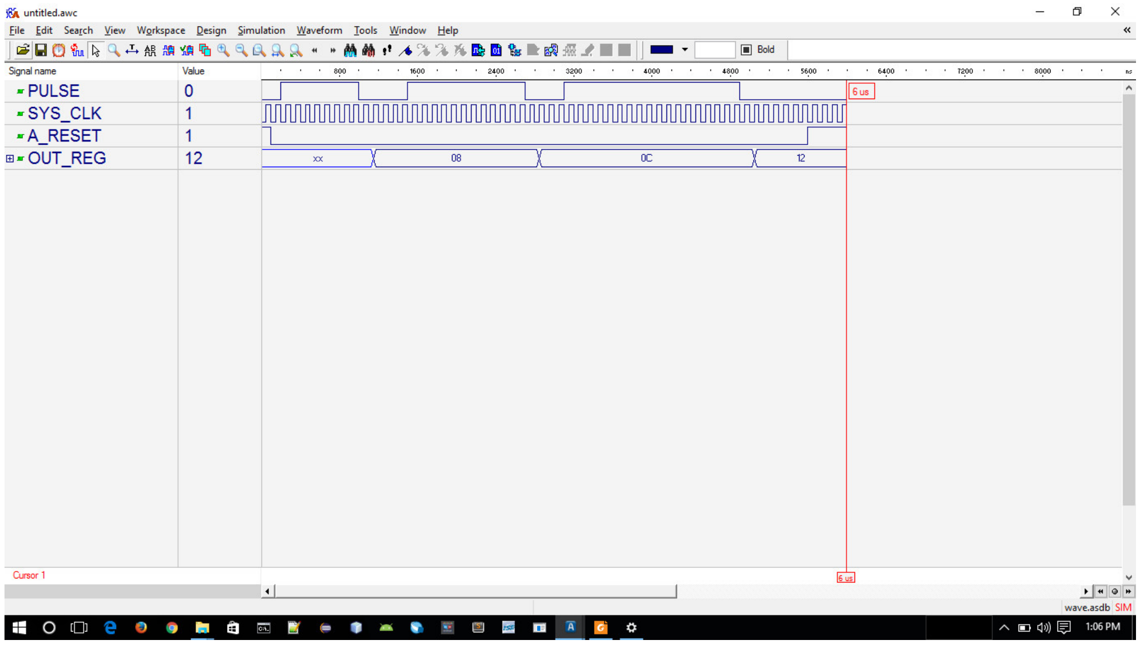
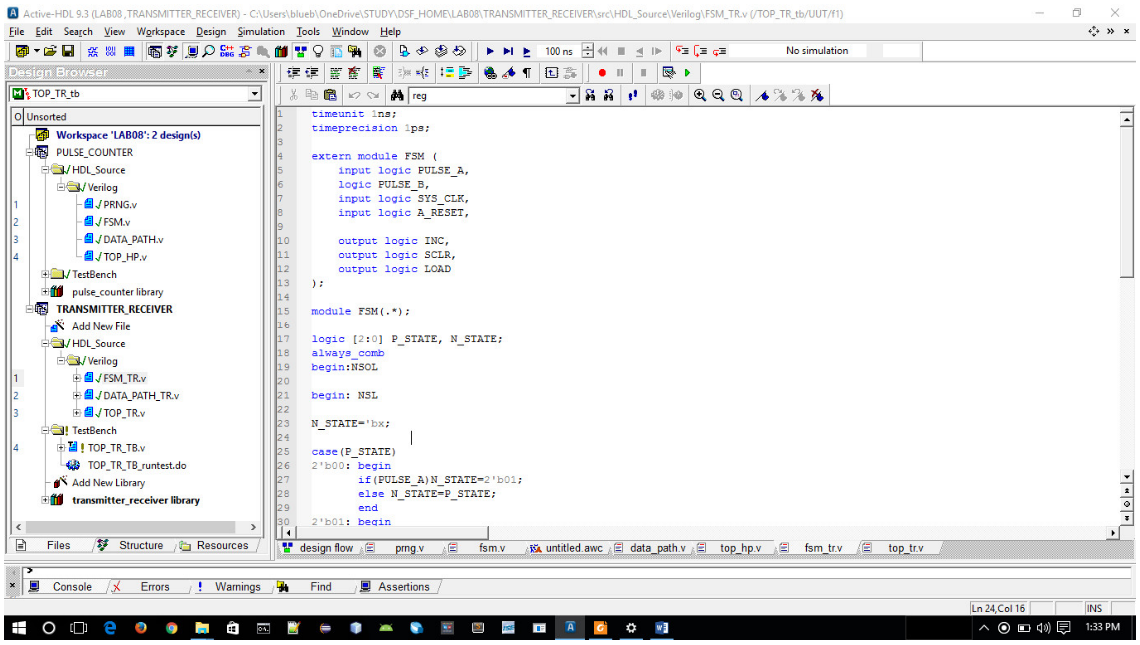


Figure : Functional verification

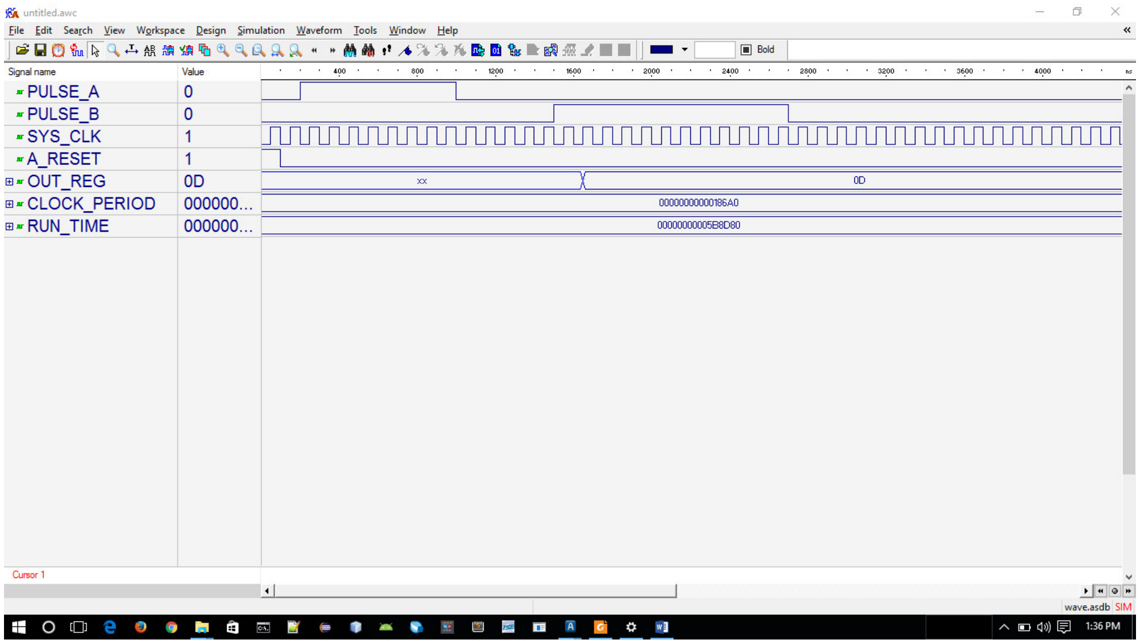
```
//Transmit Echo

//TOP LEVEL

timeunit 1ns;
timeprecision 1ps;

module TOP_TR ( input logic PULSE_A, input logic PULSE_B, input logic SYS_CLK, input logic
A_RESET, output logic [4:0] OUT_REG);

logic INC, SCLR, LOAD;

FSM f1(.*);
DATA_PATH d1 (.*);

endmodule

//FSM

timeunit 1ns;
timeprecision 1ps;

extern module FSM (input logic PULSE_A, logic PULSE_B, input logic SYS_CLK, input logic
A_RESET, output logic INC, output logic SCLR, output logic LOAD );

module FSM(.*);

logic [2:0] P_STATE, N_STATE;
always_comb
begin:NSOL

begin: NSL

N_STATE='bx;

case(P_STATE)
2'b00: begin

if(PULSE_A)N_STATE=2'b01;
 else N_STATE=P_STATE;
 end
2'b01: begin
```

```
if(PULSE_A)N_STATE=P_STATE;
 else N_STATE=2'b10;
 end
2'b10: begin

if(PULSE_B)N_STATE=2'b11;
else N_STATE=P_STATE;
end
2'b11:N_STATE=2'b11;
default: N_STATE=2'b00;
endcase
end:NSL

begin: OL

{INC,SCLR,LOAD}='b0;

case(P_STATE)
2'b00: SCLR=1'b1;
2'b01: INC=1'b1;
2'b10: INC=1'b1;
2'b11: LOAD=1'b1;
default:;
endcase
end: OL

end:NSOL

always@(posedge SYS_CLK, posedge A_RESET)
begin:PSR

if(A_RESET)P_STATE<=2'b00;
else P_STATE<=N_STATE;
end:PSR

endmodule:FSM
```



```
timeunit 1ns;
timeprecision 1ps;

extern module FSM (
    input logic PULSE_A,
    logic PULSE_B,
    input logic SYS_CLK,
    input logic A_RESET,

    output logic INC,
    output logic SCLR,
    output logic LOAD
);

module FSM(.*);

logic [2:0] P_STATE, N_STATE;
always_comb
begin:NSOL

begin: NSL

N_STATE='bx;

case(P_STATE)
2'b00: begin
       if(PULSE_A)N_STATE=2'b01;
       else N_STATE=P_STATE;
       end
2'b01: begin
```

Figure : Source Code



Signal name	Value
PULSE_A	0
PULSE_B	0
SYS_CLK	1
A_RESET	1
OUT_REG	0D
CLOCK_PERIOD	000000...
RUN_TIME	000000...

```
OUT_REG       xx  0D
CLOCK_PERIOD  00000000000186A0
RUN_TIME      00000000005B8D80
```

Figure : functional simulation

<u>Reference:</u>

1. *D.J. Smith, HDL Chip Design: A Practical Guide for Designing, Synthesizing & Simulating ASUC & FPGA using VHDL or Verilog , Madison, AL, USA, Doone Publications. 1996, 6th Printing-1999(minor revisions and code updates for FPGA synthesis)*