TÉCNICO LISBOA

Projecto e Controlo em Lógica Digital

2nd Laboratory

Work #1 – (Breitling)

Objective: Implement a micro-stopwatch.

1.1 Implement a counter

Objective: Implement a counter. The counter should be incremented in each clock cycle of the 50MHz clock of the DE2 board. The counter output should be connected to the available LEDs so that they blink at different frequencies. It is intended to prepare a module to perform a division of the clock frequency to get a clock with a microsecond period.

- Launch the Quartus II and open the DE2_top project;
- Identify the signal clk_50 as well as the signals corresponding to the LEDs (LEDR- red LEDs; LEDG- green LEDs);
- Implement a counter in which the input is the clk_50 and connect the outputs to the LEDs.

1.2 7-segment decoder

Objective: To decode a number in binary so that it is shown in the 7-segment display using a decimal base. The number is input using switches from the board and the number should appear in decimal in the display. It is intended to prepare a module that decodes the binary counting of the clock for the visualization of the elapsed time.

- Launch the Quartus II and open the DE2_top project;
- Identify the signals from the switches (SW) as well as the signals corresponding to a 7-segment display;
- Implement the code that allows to make the decoding. The inputs will be the signals from the switches and the outputs, the segments of the display.

(Sugestion: recycle the code from the previous laboratory)

1.3 Micro-Stopwatch

Objective: Implement a stopwatch with microsecond resolution. The stopwatch should have as control lines a reset (from a pushbutton) and a start/stop signal from a switch. The elapsed time is to be shown in the 7-segment displays in the format

sm mm μμμ (s-seconds, m- thousandth of second, μ-micro seconds).

The 1MHz clock is built from the 50MHz clock using a counter (“sounds familiar”). The numbers are shown in the 7-segment displays using a decoder (“sounds familiar”). It is necessary to implement a counter for the elapsed time and the control of the stopwatch.

Reconfigure the stopwatch in such a way that the start/stop signal is given as an external input (GPIO). This external input will be a square wave from a signal generator with a certain time at “high” in order to check the proper working of the stopwatch with the required precision.

Support Material

DISCLAIMER:
THE FOLLOWING CONTENT DO NOT BIND THE AUTHOR AND MAY BE INCORRECT OR INCOMPLETE, EVEN INTENTIONALLY! THEY SHOULD BE UNDERSTOOD AS MOTIVATIONAL MATERIAL.

Asynchronous Counters

Let’s implemente two asynchronous counters (the clock is not connected to all the output states. The clock of the “next” bit is the previous bit) of four bits each connected one to the other in such a way to obtain an 8 bit counter. Each counter will have 4 bits, hence we will use 4 flip-flops T. The T flip-flop is a primitive of the VerilogHDL that is instantiated as:

```
tff instance_name(IN,CLK,CLEAR,PRESET,OUT)
```

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

counter4asy counter40(LEDR[11:8], Cont[4]); //use module counterasy
counter4asy counter41(LEDR[15:12], LEDR[11]);

module counter4asy(q, clk);//module that implements a 4 bit asynchronous
counter

output [3:0] q;

input clk;
//The clock is negated as we want to change values in the falling edges to
get a count in the rising edge
tff tff0(.t(1), .clk(!clk),.clrn(1), .prn(1), .q(q[0]));
tff tff1(.t(1), .clk(!q[0]),.clrn(1), .prn(1), .q(q[1]));
tff tff2(.t(1), .clk(!q[1]),.clrn(1), .prn(1), .q(q[2]));
tff tff3(.t(1), .clk(!q[2]),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implement a counter and show the counts in the leds.
It can be seen that the clock of the following flip-flop comes from the switching of the previous flip-flop.

Synchronous counters

Let's implemente two synchronous counters (the clock is common to all flip-flops of the counter) of four bits connected to each other to obtain na 8 bit counter. Each counter will have 4 bits, hence 4 flip-flops are used.

The counter have na enable and rco enabling to connect them.

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

wire rco;//rco to enable to activate the second counter when the first one
reaches 15
counter4sinc counter42(LEDG[3:0], rco, Cont[4], 1);
counter4sinc counter43(LEDG[7:4], , Cont[4], rco);

module counter4sinc(q, rco, clk,enb );//module that implements a 4 bit
synchronous counter with enable and rco

output [3:0] q, rco=(q[0] && q[1] && q[2] && q[3])  ;

input clk, enb;

wire a[4:0];

tff tff0(.t(enb), .clk(clk),.clrn(1), .prn(1), .q(q[0]));
and and0(a[0], q[0], enb);
tff tff1(.t(a[0]), .clk(clk),.clrn(1), .prn(1), .q(q[1]));
and and1(a[1],q[1],enb);
and and2(a[2], a[1], a[0]);
tff tff2(.t(a[2]), .clk(clk),.clrn(1), .prn(1), .q(q[2]));
and and3(a[3],q[2],enb);
and and4(a[4], a[3], a[2]);
tff tff3(.t(a[1]), .clk(clk),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implemente a counter and show the value in the LEDs.

Counter in Verilog – The easy way

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

assign LEDR[7:4] = Cont[7:4];
```

TÉCNICO LISBOA

Projecto e Controlo
em Lógica Digital

2nd Laboratory

Work #1 – (Breitling)

Objective: Implement a micro-stopwatch.

1.1 Implement a counter

Objective: Implement a counter. The counter should be incremented in each clock cycle of the 50MHz clock of the DE2 board. The counter output should be connected to the available LEDs so that they blink at different frequencies. It is intended to prepare a module to perform a division of the clock frequency to get a clock with a microsecond period.

- Launch the Quartus II and open the DE2_top project;
- Identify the signal clk_50 as well as the signals corresponding to the LEDs (LEDR- red LEDs; LEDG- green LEDs);
- Implement a counter in which the input is the clk_50 and connect the outputs to the LEDs.

1.2 7-segment decoder

Objective: To decode a number in binary so that it is shown in the 7-segment display using a decimal base. The number is input using switches from the board and the number should appear in decimal in the display. It is intended to prepare a module that decodes the binary counting of the clock for the visualization of the elapsed time.

- Launch the Quartus II and open the DE2_top project;
- Identify the signals from the switches (SW) as well as the signals corresponding to a 7-segment display;
- Implement the code that allows to make the decoding. The inputs will be the signals from the switches and the outputs, the segments of the display.

(Sugestion: recycle the code from the previous laboratory)

1.3 Micro-Stopwatch

Objective: Implement a stopwatch with microsecond resolution. The stopwatch should have as control lines a reset (from a pushbutton) and a start/stop signal from a switch. The elapsed time is to be shown in the 7-segment displays in the format

sm mm µµµ (s-seconds, m- thousandth of second, µ-micro seconds).

The 1MHz clock is built from the 50MHz clock using a counter ("sounds familiar"). The numbers are shown in the 7-segment displays using a decoder ("sounds familiar"). It is necessary to implement a counter for the elapsed time and the control of the stopwatch.

Reconfigure the stopwatch in such a way that the start/stop signal is given as an external input (GPIO). This external input will be a square wave from a signal generator with a certain time at "high" in order to check the proper working of the stopwatch with the required precision.

Support Material

DISCLAIMER:
THE FOLLOWING CONTENT DO NOT BIND THE AUTHOR AND MAY BE INCORRECT OR INCOMPLETE, EVEN INTENTIONALLY! THEY SHOULD BE UNDERSTOOD AS MOTIVATIONAL MATERIAL.

Asynchronous Counters

Let’s implemente two asynchronous counters (the clock is not connected to all the output states. The clock of the “next” bit is the previous bit) of four bits each connected one to the other in such a way to obtain an 8 bit counter. Each counter will have 4 bits, hence we will use 4 flip-flops T. The T flip-flop is a primitive of the VerilogHDL that is instantiated as:

```
tff instance_name(IN,CLK,CLEAR,PRESET,OUT)
```

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

counter4asy counter40(LEDR[11:8], Cont[4]); //use module counterasy
counter4asy counter41(LEDR[15:12], LEDR[11]);

module counter4asy(q, clk);//module that implements a 4 bit asynchronous
counter

output [3:0] q;

input clk;
//The clock is negated as we want to change values in the falling edges to
get a count in the rising edge
tff tff0(.t(1), .clk(!clk),.clrn(1), .prn(1), .q(q[0]));
tff tff1(.t(1), .clk(!q[0]),.clrn(1), .prn(1), .q(q[1]));
tff tff2(.t(1), .clk(!q[1]),.clrn(1), .prn(1), .q(q[2]));
tff tff3(.t(1), .clk(!q[2]),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implement a counter and show the counts in the leds.
It can be seen that the clock of the following flip-flop comes from the switching of the previous flip-flop.

Synchronous counters

Let’s implemente two synchronous counters (the clock is common to all flip-flops of the counter) of four bits connected to each other to obtain na 8 bit counter. Each counter will have 4 bits, hence 4 flip-flops are used.

The counter have na enable and rco enabling to connect them.

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

wire rco;//rco to enable to activate the second counter when the first one
reaches 15
counter4sinc counter42(LEDG[3:0], rco, Cont[4], 1);
counter4sinc counter43(LEDG[7:4], , Cont[4], rco);

module counter4sinc(q, rco, clk,enb );//module that implements a 4 bit
synchronous counter with enable and rco

output [3:0] q, rco=(q[0] && q[1] && q[2] && q[3])  ;

input clk, enb;

wire a[4:0];

tff tff0(.t(enb), .clk(clk),.clrn(1), .prn(1), .q(q[0]));
and and0(a[0], q[0], enb);
tff tff1(.t(a[0]), .clk(clk),.clrn(1), .prn(1), .q(q[1]));
and and1(a[1],q[1],enb);
and and2(a[2], a[1], a[0]);
tff tff2(.t(a[2]), .clk(clk),.clrn(1), .prn(1), .q(q[2]));
and and3(a[3],q[2],enb);
and and4(a[4], a[3], a[2]);
tff tff3(.t(a[1]), .clk(clk),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implemente a counter and show the value in the LEDs.

Counter in Verilog – The easy way

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

assign LEDR[7:4] = Cont[7:4];
```

2nd Laboratory

Work #1 – (Breitling)

Objective: Implement a micro-stopwatch.

1.1 Implement a counter

Objective: Implement a counter. The counter should be incremented in each clock cycle of the 50MHz clock of the DE2 board. The counter output should be connected to the available LEDs so that they blink at different frequencies. It is intended to prepare a module to perform a division of the clock frequency to get a clock with a microsecond period.

- Launch the Quartus II and open the DE2_top project;
- Identify the signal clk_50 as well as the signals corresponding to the LEDs (LEDR- red LEDs; LEDG- green LEDs);
- Implement a counter in which the input is the clk_50 and connect the outputs to the LEDs.

1.2 7-segment decoder

Objective: To decode a number in binary so that it is shown in the 7-segment display using a decimal base. The number is input using switches from the board and the number should appear in decimal in the display. It is intended to prepare a module that decodes the binary counting of the clock for the visualization of the elapsed time.

- Launch the Quartus II and open the DE2_top project;
- Identify the signals from the switches (SW) as well as the signals corresponding to a 7-segment display;
- Implement the code that allows to make the decoding. The inputs will be the signals from the switches and the outputs, the segments of the display.

(Sugestion: recycle the code from the previous laboratory)

1.3 Micro-Stopwatch

Objective: Implement a stopwatch with microsecond resolution. The stopwatch should have as control lines a reset (from a pushbutton) and a start/stop signal from a switch. The elapsed time is to be shown in the 7-segment displays in the format

sm mm µµµ (s-seconds, m- thousandth of second, µ-micro seconds).

The 1MHz clock is built from the 50MHz clock using a counter (“sounds familiar”). The numbers are shown in the 7-segment displays using a decoder (“sounds familiar”). It is necessary to implement a counter for the elapsed time and the control of the stopwatch.

Reconfigure the stopwatch in such a way that the start/stop signal is given as an external input (GPIO). This external input will be a square wave from a signal generator with a certain time at “high” in order to check the proper working of the stopwatch with the required precision.

Support Material

DISCLAIMER:
THE FOLLOWING CONTENT DO NOT BIND THE AUTHOR AND MAY BE INCORRECT OR INCOMPLETE, EVEN INTENTIONALLY! THEY SHOULD BE UNDERSTOOD AS MOTIVATIONAL MATERIAL.

Asynchronous Counters

Let's implemente two asynchronous counters (the clock is not connected to all the output states. The clock of the "next" bit is the previous bit) of four bits each connected one to the other in such a way to obtain an 8 bit counter. Each counter will have 4 bits, hence we will use 4 flip-flops T. The T flip-flop is a primitive of the VerilogHDL that is instantiated as:

```
tff instance_name(IN,CLK,CLEAR,PRESET,OUT)
```

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

counter4asy counter40(LEDR[11:8], Cont[4]); //use module counterasy
counter4asy counter41(LEDR[15:12], LEDR[11]);

module counter4asy(q, clk);//module that implements a 4 bit asynchronous
counter

output [3:0] q;

input clk;
//The clock is negated as we want to change values in the falling edges to
get a count in the rising edge
tff tff0(.t(1), .clk(!clk),.clrn(1), .prn(1), .q(q[0]));
tff tff1(.t(1), .clk(!q[0]),.clrn(1), .prn(1), .q(q[1]));
tff tff2(.t(1), .clk(!q[1]),.clrn(1), .prn(1), .q(q[2]));
tff tff3(.t(1), .clk(!q[2]),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implement a counter and show the counts in the leds.
It can be seen that the clock of the following flip-flop comes from the switching of the previous flip-flop.

Synchronous counters

Let’s implemente two synchronous counters (the clock is common to all flip-flops of the counter) of four bits connected to each other to obtain na 8 bit counter. Each counter will have 4 bits, hence 4 flip-flops are used.

The counter have na enable and rco enabling to connect them.

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

wire rco;//rco to enable to activate the second counter when the first one
reaches 15
counter4sinc counter42(LEDG[3:0], rco, Cont[4], 1);
counter4sinc counter43(LEDG[7:4], , Cont[4], rco);

module counter4sinc(q, rco, clk,enb );//module that implements a 4 bit
synchronous counter with enable and rco

output [3:0] q, rco=(q[0] && q[1] && q[2] && q[3])  ;

input clk, enb;

wire a[4:0];

tff tff0(.t(enb), .clk(clk),.clrn(1), .prn(1), .q(q[0]));
and and0(a[0], q[0], enb);
tff tff1(.t(a[0]), .clk(clk),.clrn(1), .prn(1), .q(q[1]));
and and1(a[1],q[1],enb);
and and2(a[2], a[1], a[0]);
tff tff2(.t(a[2]), .clk(clk),.clrn(1), .prn(1), .q(q[2]));
and and3(a[3],q[2],enb);
and and4(a[4], a[3], a[2]);
tff tff3(.t(a[1]), .clk(clk),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implemente a counter and show the value in the LEDs.

Counter in Verilog – The easy way

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

assign LEDR[7:4] = Cont[7:4];
```

2nd Laboratory

Work #1 – (Breitling)

Objective: Implement a micro-stopwatch.

1.1 Implement a counter

Objective: Implement a counter. The counter should be incremented in each clock cycle of the 50MHz clock of the DE2 board. The counter output should be connected to the available LEDs so that they blink at different frequencies. It is intended to prepare a module to perform a division of the clock frequency to get a clock with a microsecond period.

- Launch the Quartus II and open the DE2_top project;
- Identify the signal clk_50 as well as the signals corresponding to the LEDs (LEDR- red LEDs; LEDG- green LEDs);
- Implement a counter in which the input is the clk_50 and connect the outputs to the LEDs.

1.2 7-segment decoder

Objective: To decode a number in binary so that it is shown in the 7-segment display using a decimal base. The number is input using switches from the board and the number should appear in decimal in the display. It is intended to prepare a module that decodes the binary counting of the clock for the visualization of the elapsed time.

- Launch the Quartus II and open the DE2_top project;
- Identify the signals from the switches (SW) as well as the signals corresponding to a 7-segment display;
- Implement the code that allows to make the decoding. The inputs will be the signals from the switches and the outputs, the segments of the display.

(Sugestion: recycle the code from the previous laboratory)

1.3 Micro-Stopwatch

Objective: Implement a stopwatch with microsecond resolution. The stopwatch should have as control lines a reset (from a pushbutton) and a start/stop signal from a switch. The elapsed time is to be shown in the 7-segment displays in the format

sm mm μμμ (s-seconds, m- thousandth of second, μ-micro seconds).

The 1MHz clock is built from the 50MHz clock using a counter (“sounds familiar”). The numbers are shown in the 7-segment displays using a decoder (“sounds familiar”). It is necessary to implement a counter for the elapsed time and the control of the stopwatch.

Reconfigure the stopwatch in such a way that the start/stop signal is given as an external input (GPIO). This external input will be a square wave from a signal generator with a certain time at “high” in order to check the proper working of the stopwatch with the required precision.

Support Material

DISCLAIMER:
THE FOLLOWING CONTENT DO NOT BIND THE AUTHOR AND MAY BE INCORRECT OR INCOMPLETE, EVEN INTENTIONALLY! THEY SHOULD BE UNDERSTOOD AS MOTIVATIONAL MATERIAL.

Asynchronous Counters

Let's implemente two asynchronous counters (the clock is not connected to all the output states. The clock of the "next" bit is the previous bit) of four bits each connected one to the other in such a way to obtain an 8 bit counter. Each counter will have 4 bits, hence we will use 4 flip-flops T. The T flip-flop is a primitive of the VerilogHDL that is instantiated as:

```
tff instance_name(IN,CLK,CLEAR,PRESET,OUT)
```

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

counter4asy counter40(LEDR[11:8], Cont[4]); //use module counterasy
counter4asy counter41(LEDR[15:12], LEDR[11]);

module counter4asy(q, clk);//module that implements a 4 bit asynchronous
counter

output [3:0] q;

input clk;
//The clock is negated as we want to change values in the falling edges to
get a count in the rising edge
tff tff0(.t(1), .clk(!clk),.clrn(1), .prn(1), .q(q[0]));
tff tff1(.t(1), .clk(!q[0]),.clrn(1), .prn(1), .q(q[1]));
tff tff2(.t(1), .clk(!q[1]),.clrn(1), .prn(1), .q(q[2]));
tff tff3(.t(1), .clk(!q[2]),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implement a counter and show the counts in the leds.
It can be seen that the clock of the following flip-flop comes from the switching of the previous flip-flop.

Synchronous counters

Let’s implemente two synchronous counters (the clock is common to all flip-flops of the counter) of four bits connected to each other to obtain na 8 bit counter. Each counter will have 4 bits, hence 4 flip-flops are used.

The counter have na enable and rco enabling to connect them.

The code is:

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

wire rco;//rco to enable to activate the second counter when the first one
reaches 15
counter4sinc counter42(LEDG[3:0], rco, Cont[4], 1);
counter4sinc counter43(LEDG[7:4], , Cont[4], rco);

module counter4sinc(q, rco, clk,enb );//module that implements a 4 bit
synchronous counter with enable and rco

output [3:0] q, rco=(q[0] && q[1] && q[2] && q[3])  ;

input clk, enb;

wire a[4:0];

tff tff0(.t(enb), .clk(clk),.clrn(1), .prn(1), .q(q[0]));
and and0(a[0], q[0], enb);
tff tff1(.t(a[0]), .clk(clk),.clrn(1), .prn(1), .q(q[1]));
and and1(a[1],q[1],enb);
and and2(a[2], a[1], a[0]);
tff tff2(.t(a[2]), .clk(clk),.clrn(1), .prn(1), .q(q[2]));
and and3(a[3],q[2],enb);
and and4(a[4], a[3], a[2]);
tff tff3(.t(a[1]), .clk(clk),.clrn(1), .prn(1), .q(q[3]));

endmodule
```

This code will implemente a counter and show the value in the LEDs.

Counter in Verilog – The easy way

```
reg    [7:0]      Cont;
always@(posedge CLOCK_50)
begin
      Cont  <=    Cont+1;
end

assign LEDR[7:4] = Cont[7:4];
```