Introduction to FPGA Programming

Laura Francesca Iacob, Sara Pieri, Sara Schippa
Laboratorio di Elettronica e Tecniche di Acquisizione Dati, professor Mirko Mariotti

Di seguito sono riportati i codici scritti in Vivado ed utilizzati per programmare la FPGA a disposizione nei diversi modi richiesti.

- 0_notModule

```
module notMod (
        input wire A,
        output wire B
        );
        assign B=!A;
endmodule
```

- 1_andModule

```
module andMod (
        input wire A,
        input wire B,
        output wire C
        );
        assign C=A&B;
endmodule
```

- 2_logicDoor

```
module logicDoor (
        input wire A,
        input wire B,
        input wire C,
        output wire D,
        output wire E
        );
        assign D=A&(B|C);
        assign E=B|C;
endmodule
```

- 3_blink_f0

```
module blink (
        input clk,
        output reg  [7:0] LED
        );
        reg counter;
        initial;
                counter=0;
        always @ (posedge clk)  begin
                LED[0] <= counter;
                counter <= counter +1;
        end
endmodule
```

- 4_blink_f1

```
module blink (
        input clk,
        output reg  [7:0] LED
        );
        reg [32:0] counter;

        initial;
                counter=0;

        always @ (posedge clk)  begin
                LED[0] <= counter[23];
                counter <= counter +1;
        end
endmodule
```

- 4_blink_f2

```
module blink (
        input clk,
        output reg  [7:0] LED
        );
        reg [32:0] counter;

        initial;
                counter=0;

        always @ (posedge clk)  begin
                LED[0] <= counter[31];
                counter <= counter +1;
        end
endmodule
```

- 5_switch

```
module LED_sw (
        input clk,
        input [1:0] sw,
        output reg  [7:0] LED
        );
        reg [32:0] counter;
        initial
                counter=0;
        always @ (posedge clk)  begin
                if (sw[0]==1'b1) begin
                        LED[0] <= counter[25];
                        counter <= counter +1;
        end
    end
endmodule
```

- 6_counter_binario

```
module counter_binario (
        input clk,
        output reg  [15:0] LED
        );
        reg [50:0]  counter;
        integer i;
        initial
                counter=0;
        always @ (posedge clk)  begin
                for (i=0; i<16; i=i+1)
                        LED[i] <= counter[23+i] ;
                        counter <= counter +1;
    end
endmodule
```

- 7_joystick_c

```
module joystick_c (
        input clk,
        input btnC
        output reg  [7:0] LED
        );
        integer old;
        initial
                LED[0]=1'b0;
        always @ (posedge clk)  begin
                old<=btnC;
                if(btnC!=old & btnC==1)
                    if(LED[0]==1'b0)
                        LED[0]<=1'b1;
                else if (LED[0]==1'b1)
                        LED[0]<=1'b0;
    end
endmodule
```

- 8_kitt

```
module kitt (
          input clk,
          input btnL,
          input btnR,
          output reg  [15:0] LED
         );
         reg oldR;
         reg oldL;
         reg [4:0] i;
         initial begin
                i=0;
                 LED[0]=1'b1;
    end
          always @ (posedge clk)  begin
                  oldR<=btnR;
                   oldL<=btnL;
                   if (oldR!=btnR & btnR==1 & i>0) begin
                        LED[i]<=1'b0;
                        LED[i-1]<=1'b1;
                        i<=i-1;
           end
                   else if (oldL!=btnL & btnL==1 & i<15) begin
                         LED[i]<=1'b0;
                        LED[i+1]<=1'b1;
                        i<=i+1;
           end
      end
endmodule
```