**Universidade Federal de São Carlos**

**Departamento de Computação**

## Disciplina:

Laboratório de Arquitetura e Organização de Computadores 1

# Relatório 4

Circuitos Reconfiguráveis: Flip-Flops D, contadores e máquinas de estado.

**Daniel Chaves Macedo** **RA: 280844**

**Pedro Gabriel Artiga** **RA: 351180**

**Felipe Fiori Campos Martins** **RA: 316660**

## São Carlos, 2011.

# Introdução

Neste relatório pretendemos demonstrar o funcionamento de estruturas, como flip-flops D, contadores e máquinas de estados, e como fazer uso das ferramentas já apresentadas na disciplina para implementar estes dispositivos.

# Objetivo

Implementar flip-flops D, contadores e máquinas de estados, fazendo uso da linguagem Verilog e utilizando as ferramentas do software Quartus II.

# Contador

**Crie um projeto como o nome counter.**

**Crie o arquivo verilog HDL.**

**Código:**

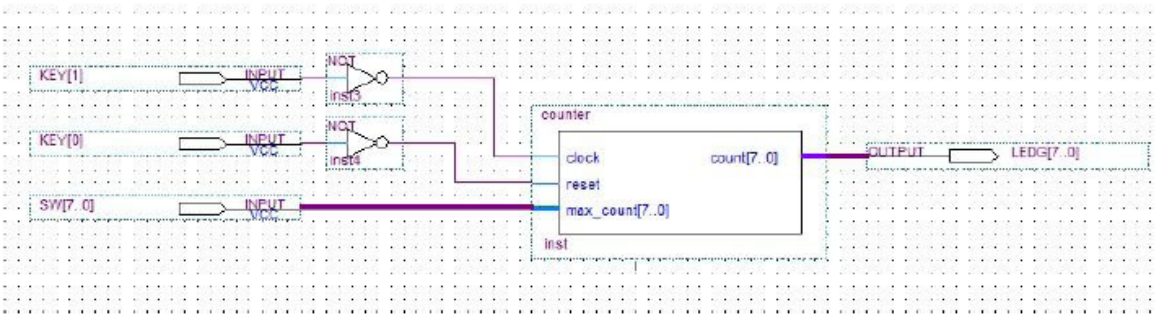
```
module counter(clock, reset,max_count,count);
        input clock;
        input reset;
        input [7:0] max_count;
        output [7:0] count;
        reg [7:0] count;

        always @(posedge clock or posedge reset)
        begin
                if (reset)
                        count = 0;
                else if (count < max_count)
                        count = count + 1;
                else
                        count = 0;
        end
endmodule
```

Salve como counter.
Compile.
File -> Create/Update -> Create Symbol Files for Current File.

**Criando diagram de blocos:**

File -> new -> Block diagram/Schematic File
Duplo clique no projeto -> project -> counter -> ok e depois na área do diagrama.
Salve como blococounter.
Adicione como inputs 2 entradas negadas, 1 barramento de 8 bits não negado e adicione como output um barramento de 8 bits.

**NOTA :Os Nomes devem ser mudados de acordo com o manual da placa FPGA:**



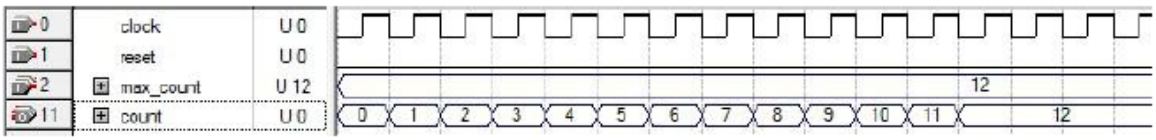

Assignments -> Import Assignments -> DE1_Default.qsf

**Para deixar seu bloco esquemático como top file :**
Files -> botão direito em blococounter.bdf -> Set as top level entity

Start Compilation para compilar.

**Waveform:**

File -> New -> Vector Waveform File.
File -> Save: blococounter.vwf
Insert Node or bus -> Node Finder.
Filter -> pins: unassigned -> List.
Botão >> -> OK.
Escolha os valores de entrada ->Start Simulation.



# Flip-flop D

**Crie um projeto como o nome DFFs.**

**Crie o arquivo verilog HDL.**

**Código:**

```
module DFFs(D, clock, reset, enable, Q1, Q2, Q3, Q4);
        input D;
        input clock;
        input reset;
        input enable;
        output Q1, Q2, Q3, Q4;
        reg Q1, Q2, Q3, Q4;

        always @(posedge clock)
                Q1 = D;
        always @(posedge clock)
                if (reset)
                        Q2 = 0;
                else
                        Q2 = D;
        always @(posedge clock or posedge reset)
                if (reset)
                        Q3 = 0;
                else
                        Q3 = D;
        always @(posedge clock or posedge reset)
                if (reset)
                        Q4= 0;
                else if (enable)
                        Q4 = D;
Endmodule
```

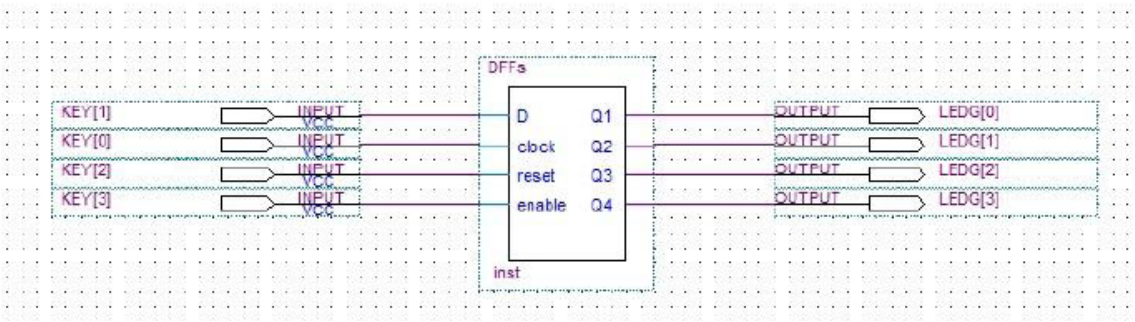
Salve como DFFs.
Compile.
File -> Create/Update -> Create Symbol Files for Current File.

**Criando diagram de blocos:**

File -> new -> Block diagram/Schematic File
Duplo clique no projeto -> project -> DFFs -> ok e depois na área do diagrama.
Salve como blocoDFFs.

Adicione 4 inputs e 4 outputs ao diagrama.

**NOTA :Os Nomes devem ser mudados de acordo com o manual da placa FPGA:**



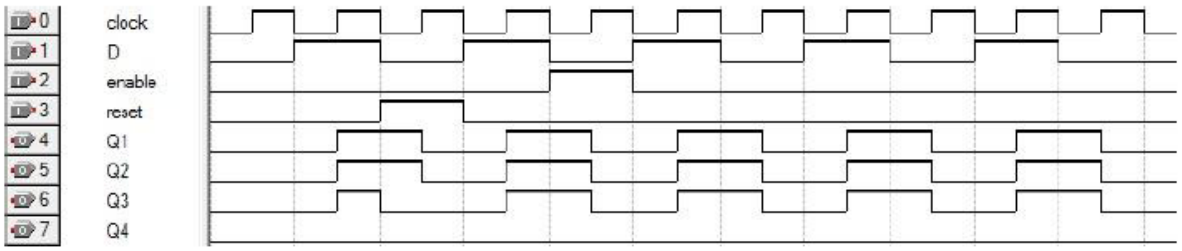

Assignments -> Import Assignments -> DE1_Default.qsf

**Para deixar seu bloco esquemático como top file :**
Files ->  botão direito em blocoDFFs.bdf -> Set as top level entity

Start Compilation para compilar.

**Waveform:**

File -> New -> Vector Waveform File.
File -> Save: blocoDFFs.vwf
Insert Node or bus -> Node Finder.
Filter -> pins: unassigned -> List.
Botão >> -> OK.
Escolha os valores de entrada ->Start Simulation.



# Máquina de estados de Moore

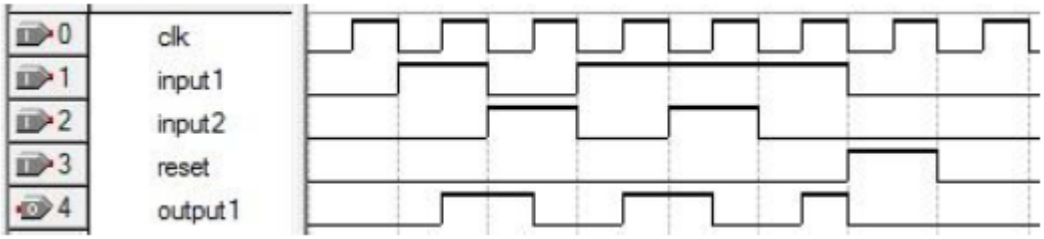
**Crie um projeto como o nome state_mach.**

**Crie o arquivo verilog HDL.**

**Código:**

```
module state_mach (clock, reset, input1, input2 ,output1);
        input clock, reset, input1, input2;
        output output1;
        reg output1;
        reg [1:0] state;
        parameter [1:0] state_A = 0, state_B = 1, state_C = 2;

        always@(posedge clock or posedge reset)
        begin
                if (reset)
                        state = state_A;
                else
                        case (state)
                                state_A: if (input1==0)
                                state = state_B;
                                        else
                                                state = state_C;
                                                state_B: state = state_C;
                                                state_C: if (input2)
                                                state = state_A;
                        endcase
        end

        always @(state)
        begin
                case (state)
                        state_A: output1 = 0;
                        state_B: output1 = 0;
                        state_C: output1 = 1;
                        default: output1 = 0;
                endcase
        end
endmodule
```

Como já demonstrado acima crie a **Waveform** e simule com as entradas desejadas.




# Máquina de estados de Mealy

**Crie um projeto como o nome mealy.**

**Crie o arquivo verilog HDL.**

**Código:**

```
module mealy(clock, input1, input2, output1);
        input clock, input1, output1, input2;
        output output1;
        reg output1;
        reg [1:0] estado, prox_estado;

        always @(posedge clock)
                if (output1)
                        estado = A;
                else
                        estado = prox_estado;

        always @(estado or input1)
                case(estado)
                        A:
                                if (input1)
                                        prox_estado = B; input2=0;
                                else
                                        prox_estado = A; input2=0;
                        B:
                                if (input1)
                                        prox_estado = C; input2=0;
                                else
                                        prox_estado = A; input2=1;
                        C:
                                if (input1)
                                        prox_estado = C; input2=0;
                                else
                                        prox_estado = D; input2=1;
                        D:
                                if (input1)
                                        prox_estado = A; input2=0;
                                else
                                        prox_estado = B; input2=1;
                endcase
endmodule
```

# Máquina de dois estados

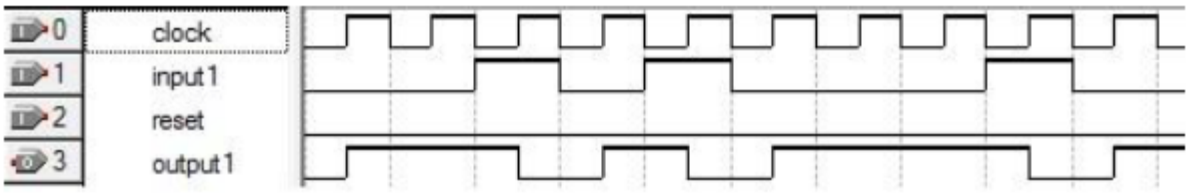
**Crie um projeto como o nome two_state.**

**Crie o arquivo verilog HDL.**

**Código:**

```
module dois_estados(clock, input1, output1, reset);
        input clock, reset, input1;
        reg [1:0] estado;
        output output1;
        reg output1;
        parameter [1:0] estadoA = 0, estadoB = 1;

        always @(posedge clock or posedge reset)
        begin
                if (reset)
                        estado = estadoA;
                else
                        case(estado)
                                estadoA:
                                        if (input1 == 0)
                                                estado = estadoA;
                                        else
                                                estado = estadoB;
                                estadoB:
                                        if (input1 == 0)
                                                estado = estadoB;
                                        else
                                                estado = estadoA;
                        endcase
        end

        always @(estado)
        begin
                case(estado)
                        estadoA: output1 = 0;
                        estadoB: output1 = 1;
                endcase
        end

endmodule
```

Como já demonstrado acima crie a **Waveform** e simule com as entradas desejadas.




# Conclusão

As ferramentas oeferecidas pelo Quartus II são capazes de realizar também o projeto de circutios combinátorios, facilitando ainda mais os nossos projetos, além disso ele facilitar arquitetar máquinas de estados e estudar seu funcionamento.