Identificador de Sequência

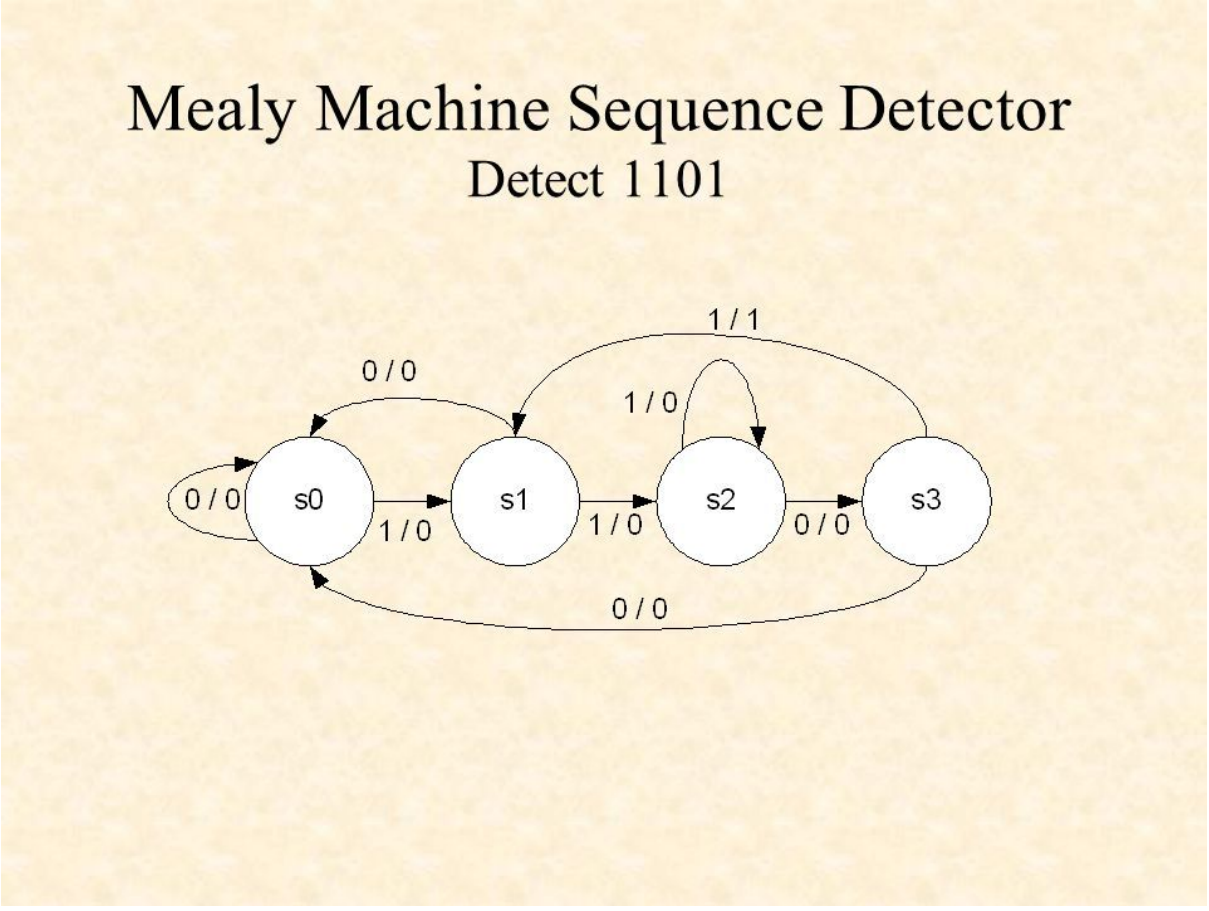
Máquina de Mealy

Para que possamos identificar uma sequência a partir de uma entrada serial, utilizei o conceito de Máquina de Mealy. Que é uma máquina de estado finito que produz um resultado (saída de dados) baseando-se no estado em que se encontra e na entrada de dados. Podemos ver alguns exemplos em verilog no endereço abaixo:

http://electrosofts.com/verilog/fsm.html

Detector 1101

A máquina que resolve o problema já se encontrava construída na internet da seguinte forma:



Baseado na máquina mostrada, a implementação ficou assim:

```
/*Para identificarmos a sequencia 1101, usaremos uma Máquina De Mealy
*/

module detector(
  input wire clock,
  input wire reset,
  input wire sequence,
  output reg flag);

//Estados
localparam A = 2'b00;
localparam B = 2'b01;
localparam C = 2'b10;
localparam D = 2'b11;

//Sinais
reg [1:0] presentState ;
reg [1:0] nextState ;

//Controle de estados
always @ (posedge clock , posedge reset)
begin
  if(reset) presentState <= A;
  else if (clock) presentState <= nextState;
end

always @ *
begin
  nextState = presentState;
  flag = 1'b0;

  case(presentState)
    A: if (sequence)   nextState = B;
    B:
      begin
        if (sequence)   nextState = C;
        else   nextState = A;
      end
    C: if (~sequence)  nextState = D;
    D:
      begin
        if(sequence)
        begin
          flag = 1'b1;
          nextState = B;
        end
        else    nextState = A;
      end
    default:
      begin
        flag = 1'b0;
        nextState = A;
      end
  endcase
end
endmodule
```

Se flag = 1, quer dizer que a sequência foi encontrada;

Para testar seu funcionamento, construí o seguinte testbench, este imprime a flag a cada ciclo de clock:

```
// Code your testbench here
// or browse Examples
`timescale 1 ns / 1 ns
module test;

  //Sinais
  reg clock, reset;
  reg sequence;
  wire flag; //Flag que nos dirá se a sequencia foi encontrada

  //Inicializando registradores em 0
  initial
  begin
    clock = 1'b0 ;
    sequence = 1'b0;
  end

  //Inicializando reset em 1 e com atraso de 30 para 0
  initial
  begin
    reset = 1'b1 ;
    #30 reset = 1'b0 ;
  end

  //Entrada serial dada a cada subida do clock
  initial
  begin
    @(posedge clock);
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);

  $finish;

  end


  //Inicializando a nossa máquina
  detector machine(
      .clock(clock),
      .reset(reset),
      .sequence(sequence),
      .flag(flag)
  );

  always #11 clock = ~clock;
  always #22 $display(flag);

endmodule
```

O circuito completo simulado e seu respectivo testbench está no seguinte endereço:

https://www.edaplayground.com/x/3YyQ

Utilize o Icarus Verilog 0.9.7. Se for de interesse, é possível mudar a entrada serial apenas alterando os valores do sequence.

Detector de Sequência Qualquer (4bits)

Foi utilizada a mesma ideia da implementação anterior, só que agora recebemos uma sequência qualquer de 4 bits para ser identificada. Com as alterações necessárias, o circuito ficou assim:

```
/*Para identificarmos a sequencia, usaremos uma Máquina De Mealy*/

module detector(
  input wire clock,
  input wire reset,
  input [3:0] in,
  input wire sequence,
  output reg flag);

//Estados
localparam A = 2'b00 ;
localparam B = 2'b01 ;
localparam C = 2'b10 ;
localparam D = 2'b11 ;

//Sinais
reg [1:0] presentState ;
reg [1:0] nextState ;

//Controle de estados
always @ (posedge clock , posedge reset )
begin
  if(reset)     presentState <= A;
  else if (clock)   presentState <= nextState;
end

always @ *
begin
  nextState = presentState;
  flag = 1'b0;

  case(presentState)
    A: if (sequence == in[3])   nextState = B ;
    B:
          begin
            if (sequence == in[2])   nextState = C ;
            else         nextState = A ;
          end
    C: if (sequence == in[1])  nextState = D ;
    D:
          begin
            if(sequence == in[0])
                begin
                  flag = 1'b1 ;
                  nextState = A;
                end
            else    nextState = A;
          end
      default:
              begin
                flag = 1'b0 ;
                nextState = A ;
              end
    endcase
end
endmodule
```

Se flag = 1, quer dizer que a sequência foi encontrada;

Para testar seu funcionamento, construí o seguinte testbench, este imprime a flag a cada ciclo de clock:

SV/Verilog Testbench

```
// Code your testbench here
// or browse Examples
`timescale 1 ns / 1 ns
module firstFSMTest ;

  //Sequencia a ser detectada
  reg [3:0] in = 4'b0010;

  //Sinais
  reg clock, reset;
  reg sequence;
  wire flag; //Flag que nos dirá se a sequencia foi encontrada

  //Inicializando registradores em 0
  initial
  begin
    clock = 1'b0 ;
    sequence = 1'b0;
  end

  //Inicializando reset em 1 e com atraso de 30 para 0
  initial
  begin
    reset = 1'b1 ;
    #30 reset = 1'b0 ;
  end

  //Entrada serial dada a cada subida do clock
  initial
  begin
    @(posedge clock);
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);     sequence <= 1'b0 ;
    @(posedge clock);     sequence <= 1'b1 ;
    @(posedge clock);

  $finish;

  end


  //Inicializando a nossa máquina
  detector machine (
     .clock(clock),
     .reset(reset),
     .in(in),
     .sequence(sequence),
      .flag(flag));

  always #11 clock = ~clock;
  always #22 $display(flag);

endmodule
```

O circuito completo simulado e seu respectivo testbench está no seguinte endereço:

https://www.edaplayground.com/x/6FsL

Utilize o Icarus Verilog 0.9.7. Se for de interesse, é possível mudar a entrada serial apenas alterando os valores do sequence, bem como a sequência, mudando o valor de “in”.