Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática – ICEI
Arquitetura de Computadores I

ARQ1 _ Aula_12
Tema: Introdução à linguagem Verilog e simulação em Logisim (memórias)

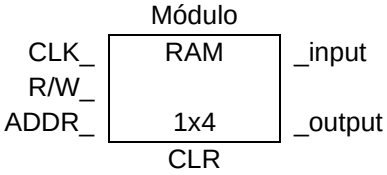
Orientação geral:

Apresentar todas as soluções em apenas um arquivo com formato texto (.txt).
Sugere-se usar como nome Guia_xx.txt, onde xx indicará o guia, exemplo Guia_01.txt.
Todos os arquivos deverão conter identificações iniciais com o nome e matrícula,
no caso de programas, usar comentários.
As implementações e testes dos exemplos em Verilog (**.v**) fornecidos como pontos de partida,
também fazem parte da atividade e deverão ter os códigos fontes entregues **separadamente**,
com o código fonte, a fim de que possam ser compilados e testados. Entregar os módulos de testes.
Sugere-se usar como nomes Guia_01yy.v, onde yy indicará a questão, exemplo Guia_0101.v
As saídas de resultados, opcionalmente, poderão ser copiadas ao final do código, em comentários.
Quaisquer outras anotações, observações ou comentários poderão ser colocadas em arquivo texto (README.txt) acompanhando a entrega.

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades
a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.
Os programas com funções desenvolvidas em C, Java ou Python (c, .java, py),
como os modelos usados para verificação automática de testes das respostas;
caso entregues, também deverão estar em arquivos **separados**, com o código fonte,
a fim de serem compilados e testados.
As execuções deverão, preferencialmente, serão testadas mediante uso de redirecionamento de entradas e saídas padrões, cujos dados/resultados deverão ser armazenados em arquivos textos.
Os resultados poderão ser anexados ao código, ao final, como comentários.
Planilhas, caso venham a ser solicitadas, deverão ser **programadas** e/ou usar funções nativas.
Serão suplementares e opcionais, e deverão ser entregues em formato texto, preferencialmente,
com colunas separadas por tabulações ou no formato (**.csv**), acompanhando a solução em texto.
Arquivos em formato (**.pdf**), fotos, cópias de tela ou soluções manuscritas também poderão ser aceitos como recursos suplementares para visualização, mas não servirão como substitutos e **não** terão validade para fins de avaliação.
Os *layouts* de circuitos deverão ser entregues no formato (**.circ**), identificados internamente.
Os *layouts* de diagramas deverão ser entregues no formato (**.jff**), identificados externamente.
Figuras exportadas pela ferramenta serão aceitas apenas como arquivos para visualização,
mas **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

Atividade: Circuitos sequenciais – Memórias
Todos os circuitos deverão ser simulados no Logisim.

01.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 1x4 (1 endereço para 4 bits)
usando flip-flops do tipo JK como registradores de dados.
Ver sugestão abaixo.



DICA:
Supor que a escrita ocorrerá na borda de subida do ***clock***,
enquanto a leitura poderá ocorrer a partir da borda de descida do mesmo.
Caso o endereço não seja selecionado, a saída padrão poderá ser zero ou indefinida (x).

02.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 1x8 (1 endereço para 8 bits cada)
usando duas memórias RAM 1x4.

03.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 2x8 (2 endereços para 8 bits cada)
usando memórias RAM 1x8.

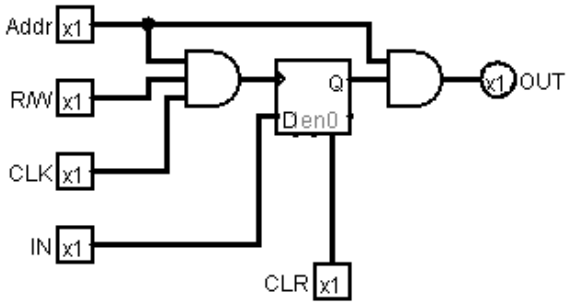
04.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 4x8 (4 endereços para 8 bits)
usando memórias RAM 2x8.

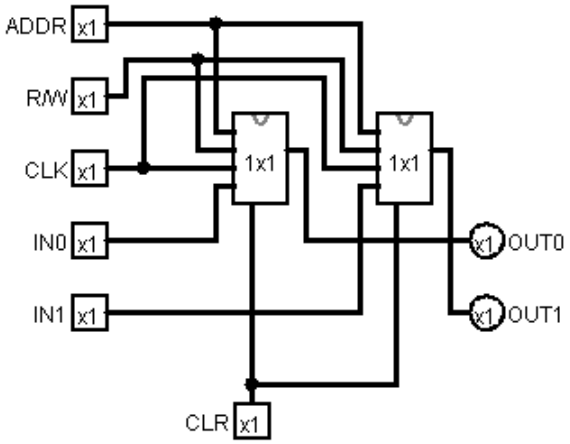
05.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 8x8 (8 endereços para 8 bits)
usando memórias RAM 4x8 do modelo acima (04).

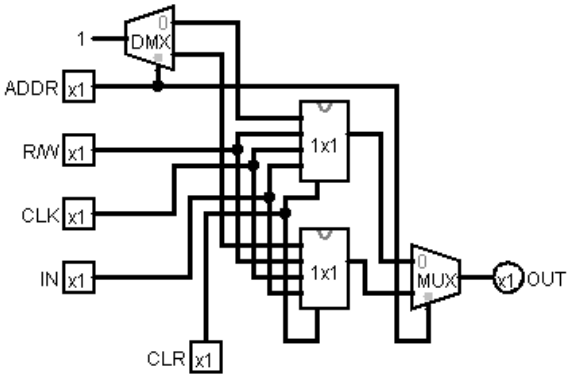
Extras

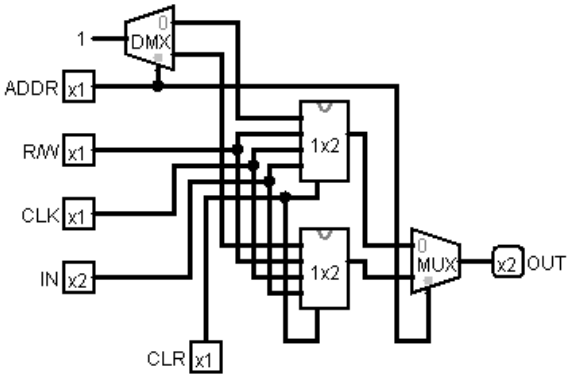
06.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 4x16 (4 endereços para 16 bits)
usando memórias RAM 4x8.

07.) Projetar e descrever em Logisim e Verilog por portas um módulo
para implementar uma memória RAM 8x16 (8 endereços para 16 bits)
usando memórias RAM 1x16.


Addr
x1
R/W
x1
CLK
x1
IN
x1
Q
D en0
CLR
x1
x1
OUT


ADDR
x1
R/W
x1
CLK
x1
IN0
x1
IN1
x1
CLR
x1
1x1
1x1
x1
OUT0
x1
OUT1


1
0
DMX
ADDR
x1
R/W
x1
CLK
x1
IN
x1
CLR
x1
1x1
1x1
0
MUX
x1
OUT


1
0
DMX
ADDR
x1
R/W
x1
CLK
x1
IN
x2
CLR
x1
1x2
1x2
0
MUX
x2
OUT

```
module dff ( output q, output qnot,
                input   d, input clk,
                input   preset, input clear );
reg q, qnot;

always @( posedge clk )
begin
  if ( clear )      begin q <= 0; qnot <=   1; end
  else
    if ( preset )  begin q <= 1; qnot <=   0; end
    else
                   begin  q <= d; qnot <= ~d; end
end

endmodule // dff

module jkff ( output q, output qnot,
            input  j, input k,
            input clk, input preset, input clear );

reg    q, qnot;

always @( posedge clk or
                posedge preset or
                posedge clear )
begin
  if ( clear )      begin q <= 0; qnot <= 1; end
  else
    if ( preset )  begin q <= 1; qnot <= 0; end
    else
      if ( j & ~k ) begin q <= 1; qnot <= 0; end
      else
       if ( ~j & k ) begin q <= 0; qnot <= 1; end
       else
        if ( j & k )
              begin q <= ~q; qnot <= ~qnot; end
end

endmodule // jkff
```

```
module tff ( output q, output qnot,
                input    t, input    clk,
                input    preset, input clear );

reg q, qnot;

always @( posedge clk or ~preset or ~clear)
begin
 if ( ~clear )
  begin    q <= 0;            qnot <= 1;   end
 else
  if ( ~preset )
   begin   q <= 1;            qnot <= 0;   end
  else
   begin
    if ( t ) begin  q <= ~q; qnot <= ~qnot; end
   end
end

endmodule // tff

module srff ( output q, output qnot,
                  input    s, input  r, input clk,
                  input preset, input clear );
reg q, qnot;

always @( posedge clk )
begin
  if ( clear )        begin q <= 0; qnot <= 1; end
  else
    if ( preset )    begin q <= 1; qnot <= 0; end
    else
      if ( s & ~r )   begin q <= 1; qnot <= 0; end
      else
        if ( ~s & r ) begin q <= 0; qnot <= 1; end
        else
          if ( s & r )
           begin   q <= 0; qnot <= 0;   end // arbitrary
end

endmodule // srff
```