UFPR – Bacharelado em Ciência da Computação
ci1068 – Circuitos Lógicos – 2019-1

Primeira Prova

1. Esta questão tem dois itens:

(a) Projete um circuito que tem como entrada um vetor de quatro bits $B = \langle b_3, b_2, b_1, b_0 \rangle$, e duas saídas: a saída V é 1 sempre que os quatro bits de B são zero, e a saída $N = \langle n_1, n_0 \rangle$ é o número (índice) da posição do bit mais à esquerda (mais significativo) que não é zero. O número N indica qual é a posição do 'primeiro' bit em 1. [10 pontos]

(b) Use dois dos circuitos do item (a) para projetar um circuito com uma entrada de 8 bits $C = \langle c_7, c_6, \ldots, c_2, c_1, c_0 \rangle$, e duas saídas: a saída V é 1 sempre que os oito bits de B são zero, e a saída $M = \langle m_2, m_1, m_0 \rangle$ é o número (índice) da posição do bit mais à esquerda (mais significativo) que não é zero. [10 pontos]

2. Considerando que o tempo de propagação através do circuito de um somador completo ($a + b + vem$) é de 5 unidades de tempo, projete um somador que soma quatro números de oito bits cada. Seu somador deve ter o menor tempo de propagação possível.

Dados A, B, C, D, todos representados em 8 bits, seu circuito deve produzir

$$S = A + B + C + D\,.$$

Ignore *overflow*. [5 pontos]

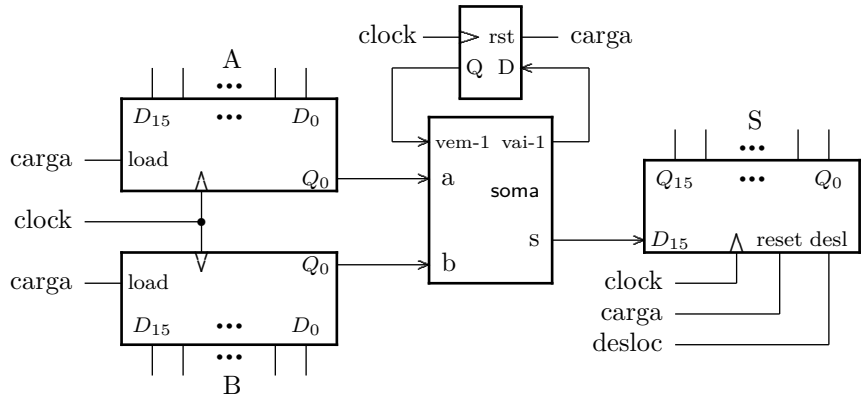
3. Efetue as cinco operações abaixo. Todos os números estão representados em cinco bits, em complemento de dois. [5 pontos]

(a) $01010 - 11011$
(b) $01010 + 11011$
(c) $01010 \oplus 11011$
(d) $01010 \ll 3$
(e) $11010 \gg 2$ deslocamento aritmético

Espaço em branco proposital.

Segunda Prova

4. Projete a máquina de estados que controla a operação do somador serial indicado na figura abaixo. A operação de soma é iniciada quando o sinal inicia é ativado pelo circuito externo. Quando a operação é completada, a máquina de estados deve ativar o sinal de pronto, e esperar até que inicia seja ativado novamente. [10 pontos]



5. Traduza para *assembly* do Mico XII o programa abaixo. Seu programa deve usar as convenções de programação. [15 pontos]

```
function fat(n: integer);
   i,j: integer;
begin
   j := 1;
   if (n > 1) then
   begin
      i := 1;
      while (i <= n) do
      begin
         j := j * i;
         i := i + 1;
      end;
   end;
   fat := j;
end;

{ no corpo do programa principal }
...
show( fat(5) ); { use a instrução SHOW }
show( fat(6) );
...
```

Espaço em branco proposital.

Exame Final

6. Esta questão tem dois itens:

(a) Projete um circuito que tem como entrada um vetor de quatro bits $B = \langle b_3, b_2, b_1, b_0 \rangle$, e duas saídas: a saída V é 1 sempre que os quatro bits de B são zero, e a saída $N = \langle n_1, n_0 \rangle$ é o número (índice) da posição do bit mais à esquerda (mais significativo) que não é zero. O número N indica qual é a posição do 'primeiro' bit em 1. [20 pontos]

(b) Use dois dos circuitos do item (a) para projetar um circuito com uma entrada de 8 bits $C = \langle c_7, c_6, \ldots, c_2, c_1, c_0 \rangle$, e duas saídas: a saída V é 1 sempre que os oito bits de B são zero, e a saída $M = \langle m_2, m_1, m_0 \rangle$ é o número (índice) da posição do bit mais à esquerda (mais significativo) que não é zero. [20 pontos]

7. Projete um circuito sequencial síncrono com uma entrada de controle T e um contador de 4 bits. Quando T=0, o contador conta de dois em dois, na sequência dos números pares. Quando T=1, o contador conta na sequência dos múltiplos de três $(0, 3, 6, 9, 12, 15, 0, \ldots)$. Quando T muda, o circuito deve mudar a sequência de contagem no próximo ciclo.

Sua resposta deve conter a Tabela de Próximo Estado, um diagrama de blocos para o circuito completo e a implementação completa para o circuito do bit 2 do estado do contador. Use um multiplexador pequeno, se possível. [30 pontos]

8. Traduza para *assembly* do Mico XII o programa abaixo. Seu programa deve usar as convenções de programação. [30 pontos]

```
{ no programa principal }
...
P array [0..2047] of integer;
Q array [0..63]   of integer;
...

function reduz(var GDE array [0..2047] of integer,
               var PEQ array [0..63]   of integer,
               n: integer) : integer;
var i,j: integer;
begin
   i := 1;
   s := 0;
   while (i < n) do
   begin
      j := i AND 63;    { (i AND (64-1)) = (i MOD 64) }
      s := s + PEQ[j] + GDE[i];
      i := i * 2;
   end;
   reduz := s;
end;
...
{ no corpo do programa principal }
...
show( reduz(P, Q, 2000) ); { use a instrução SHOW }
...
```