Série Nro. 7 – MEF Síncronas II

1Q: Obter as tabelas de estado minimizadas das seguintes tabelas de transições de estados (8 estados) modelo Moore de duas entradas (com código I1, I2, I3, e I4) e duas saídas (O1 e O2)

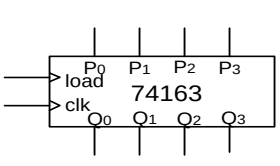
a) Especificada Incompletamente

	I1	**I2**	**I3**	**I4**	**O1**	**O2**
A	A	B	-	C	0	0
B	E	B	D	-	0	1
C	A	-	F	C	1	0
D	-	G	D	H	1	1
E	E	B	-	C	0	0
F	-	B	F	C	0	1
G	E	G	D	-	1	0
H	A	-	D	H	1	1

b) Especificada Completamente

	I1	**I2**	**I3**	**I4**	**O1**	**O2**
A	A	A	B	C	0	0
B	B	C	D	E	0	1
C	F	G	E	D	1	0
D	E	F	G	H	1	1
E	D	C	B	A	0	0
F	A	A	B	B	0	1
G	C	D	C	D	1	0
H	E	H	H	E	1	1

2Q: Usando somente um contador 74163, Mux's e portas, **projete** um **contador** binário de quatro bits com deslocamento bidirecional, onde a sua **tabela de operações** está descrita na figura 2.



clk	Dir	Des	Q_1 Q_2 Q_3 Q_4
0	x	x	Q_1 Q_2 Q_3 Q_4
1	x	x	Q_1 Q_2 Q_3 Q_4
↑	x	0	$(Q_1\ Q_2\ Q_3\ Q_4)_{N+1}$
↑	1	1	Es_1 Q_1 Q_2 Q_3
↑	0	1	Q_2 Q_3 Q_4 Es_2

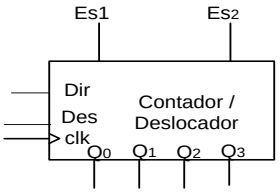
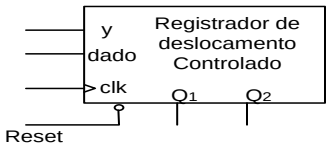


Figura 2 – Contador com deslocamento bidirecional.

3Q: Usando somente FF's JK e portas, **projete** um **registrador** de deslocamento controlado de dois bits, onde a sua **tabela de operações** está descrita na figura 3.



Reset	Dado	Y	Clk	Q_1	Q_2
0	x	x	x	0	0
1	0	0	↑	Q_1	Q_2
1	0	1	↑	0	Q_1
1	1	0	↑	1	Q_1
1	1	1	↑	Q_2	1

Figura 3 – Registrador de deslocamento controlado.

4Q: Usando a **técnica de síntese** de máquinas seqüenciais síncronas e mostrando estas etapas (diagrama de estados, tabela de estados reduzida, equações de excitação e desenho lógico), sintetize um **registrador** de deslocamento de dois bits com as entradas (*Dir*, *Dado*), onde a sua **tabela de operações** está descrito na figura 4. Use FF's JK e portas.

Dir	Dado	Clk	Q1	Q2
x	x	0	Q1	Q2
x	x	1	Q1	Q2
0	0	↑	0	Q1
0	1	↑	1	Q1
1	0	↑	Q2	0
1	1	↑	Q2	1

Figura 4 – Tabela de operações.

5Q: Usando FF's de sua preferência e portas, **projete** um **contador pseudo-síncrono** com seqüência programável, onde o seu **diagrama de estados** está descrito na figura 5.

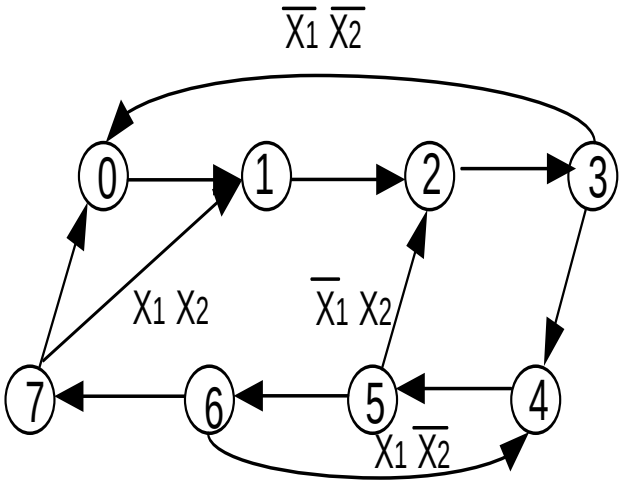


Figura 5 – Diagrama de estados.

6Q: Projete o bloco do **circuito pseudo-síncrono** mostrado na figura 6, para que este funcione segundo o seu diagrama de estados. Obs: Ignore possíveis *glitches* nas saídas.

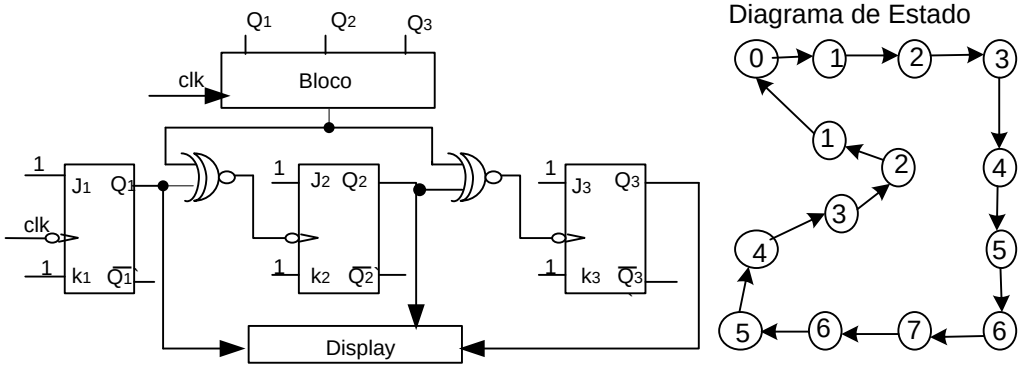
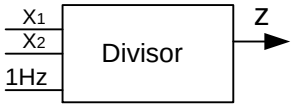


Figura 6 – Circuito pseudo-síncrono e o seu diagrama de estados.

7Q: Usando FF's de sua preferência e portas, projete um **divisor de freqüência** programável com ciclo de trabalho **simétrico** (50% alto e 50% baixo), descrito na figura 7.



X_1 X_2	Dividido por
0 0	1
0 1	2
1 0	3
1 1	4

Figura 7 – Tabela de operações do divisor programável.

8Q: Projetar um circuito digital síncrono que tem a função de **incrementador programável**. Este circuito processa segundo a tabela de **operações** descrita na figura 8. No término do **número de clocks** segundo S_1 e S_2, temos o novo valor dos Q's. Use funções MSI, FFs e portas. Por exemplo, quando S_1=S_2=1 e depois de quatro clocks temos um incremento Q←Q+1. Quando Q=1111→0000 é (próximo incremento).

CLK	S_1	S_2	Q_1 Q_2 Q_3 Q_4
↑	0	0	inibe
(2)↑	0	1	$Q_{N+1} \leftarrow Q_N + 1$
(3)↑	1	0	$Q_{N+1} \leftarrow Q_N + 1$
(4)↑	1	1	$Q_{N+1} \leftarrow Q_N + 1$

Figura 8 – Tabela de operações.

9Q: Usando funções MSI, FF's e portas, projete uma **fechadura digital** (ver figura 9 – esquema) que se a seqüência **50**, **100** e **200** for feita à tranca se abre (saída Z_3=1) caso contrário à tranca não se abre (saída Z_3=0). Duas saídas Z_1 e Z_2 mostram se a seqüência está correta. Quando Z_1=1 e Z_2=0 a **seqüência está errada**; quando Z_1=0 e Z_2=1 a **seqüência está correta**; quando Z1=Z2=0 **inicio da seqüência** deve ser introduzida; quando Z_1=Z_2=1 é **fim da seqüência** e está correta. A entrada X de 8 bits é usada para gerar a seqüência (entrada BCD). A **variável inicio**=0, a fechadura está trancada; para inicio=1, começa a leitura do código (X). A variável *inicio* é síncrona.

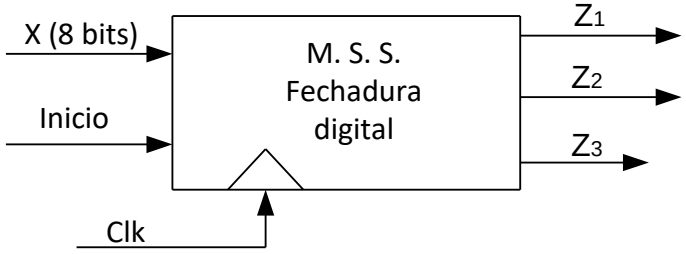


Figura 9 – Esquema da fechadura digital.

10Q: Usando funções MSI (menos contador e registrador), FFs de sua preferência e portas projetar um **contador síncrono** crescente programável de 4 bits, que realiza a tabela de operações descrita na figura 10. A variável síncrona *inicio*, inicializa a mudança de operação (inicio=1 inicializa a operação).

Clk	S	Q_0 Q_1 Q_2 Q_3
↑	0	modulo variável com inicio zero
↑	1	módulo 8 com início variável

Figura 10 – Tabela de operações.

Obs: Ilustrando o módulo 8 com início variável: Por exemplo, com o inicio=10, então temos: 10→11→12→13→14→15→0→1→10.....

Ilustrando o módulo variável com inicio zero: Por exemplo, com módulo=7, então temos: 0→1→2→3→4→5→6→0.....

11Q: Usando funções MSI, FF's D e portas projetar um registrador **que realizada a** tabela de operações descrita na figura 11**.**

CLK	S_1	S_2	Q_{N+1}
↑	0	0	**Q_{N+1} ←Dado (4 bits)**
↑	0	1	**Q_{N+1} ←3*Q_N**
↑	1	0	**Q_{N+1} ←4*Q_N**
↑	1	1	**Q_{N+1} ←5*Q_N**

Figura 11 – Tabela de operações.

12Q: Projetar um **Contador/Deslocador** de um bit (**rede iterativa seqüencial**), sendo que este pode ser usado para trabalhar em cascata, isto é, pode ser fazer contador/deslocador de **N bits**. Este circuito realiza as **operações:**

a) Se S=0 →Contador reversível pseudo-síncrono.
b) Se S=1 →Registrador de deslocamento: esq→dir; entrada e saída serial; saída paralela.

Obs: Use FF JK e portas

13Q: Projetar um **sistema de comunicação paralelo-serial**, usando funções MSI, FF's e portas (esquema – figura 13). O sistema recebe uma informação de 7 bits, quando a variável **pedido** tem a transição 0➔1. Há um processo de serialização (deslocamento: 1 bit por clock) da informação de 7 bits, e no oitavo bit é acrescentado o bit de paridade par (**BP**). O bit BP=1 se o número de 1's na informação for impar, caso contrário BP=0. Quando encerrar a transmissão serial, então temos na variável **Recebeu** o valor 1. Para um novo processamento necessitamos que a variável pedido vá de 1➔0 e posteriormente 0➔1, e a variável Recebeu inicialmente tem o valor zero.
Exemplo 1010111 1(bit de paridade); 0101110 0(bit de paridade)

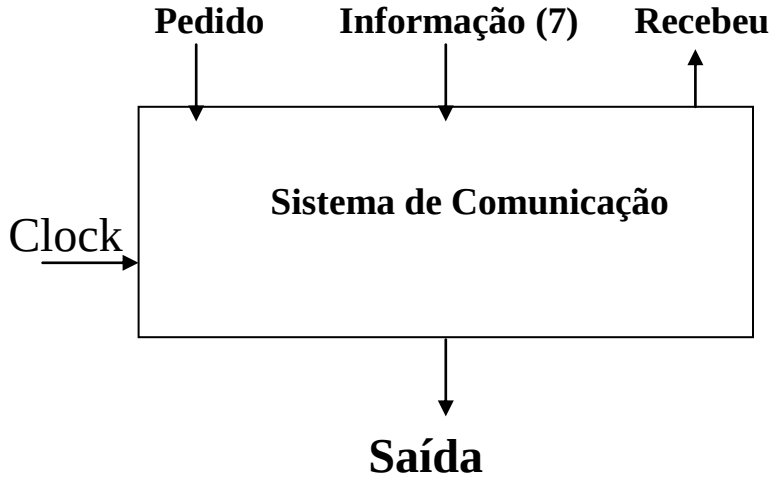


Figura 13 – Esquema de comunicação.

14Q: Faça as especificações do Data-path e do controlador, usando funções MSI, FF's e portas, do problema que calcula o **fatorial** de N (8 bits). O algoritmo é composto por duas subrotinas, descritas na figura 14.
Dados:

```
Program Fatorial
Read(N)
I:=1; Fat:=1;
Repeat
  Fat:=Fat*I;
  I:=I+1;
Until I>N
End.
```

```
Program Multiplicação
 Read(Fat,I)
  Z:=0; U:=I;
   Repeat
      Z:=Z+Fat;
      U:=U-1;
    Until U=0
    Fat:=Z;
End.
```

Figura 14 – Pseudo-código do algoritmo fatorial.

15Q: Usando funções MSI, FF's e portas projetar um **circuito digital** que lê serialmente um **protocolo** de tamanho de 8 bits. A variável de entrada **Pedido** determina a chegada do protocolo (Pedido=0➔1). Para cada protocolo que o circuito recebe e posteriormente faz o tratamento, a variável de saída **Pronto** recebe o valor 1. No inicio do recebimento do protocolo, a variável pronto recebe o valor zero. Enquanto que a variável pronto for zero o conteúdo do registrador de saída não tem significado (lixo). As figuras 15a e 15b mostram respectivamente o modelo do protocolo e o tipo de tratamento que há com as informações. O tratamento das informações é realizado em paralelo. Decompor o problema em data-path e controlador.

Operação

1	2	3 Info1 5	6 Info2 8

Figura 15a - Protocolo

OP(1)	OP(2)	Tratamento da Informação
0	0	Info1 + Info2
0	1	Info1
1	0	Info2
1	1	Maior (Info1,Info2)

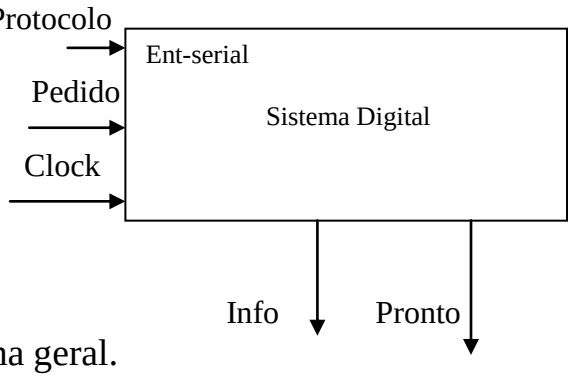


Figura 15b – Tabela de operações e esquema geral.

16Q: O sistema digital síncrono realiza a **Divisão inteira** usando apenas as operações primitivas de soma e de subtração, para números naturais de 8 bits (A e B). O sistema fornece o **Quociente e o Resto**, onde **A** é o dividendo e **B** é o divisor. Pede-se:

a) Descreva o algoritmo em uma **pseudo-linguagem** .

b) Descreva ao nível de RTL (unidades funcionais) o **Data-Path** do algoritmo de (a).

c) Descreva o **Diagrama de Estados** do algoritmo de (a).