UFRPE

Projeto de Circuitos sequenciais

Introdução

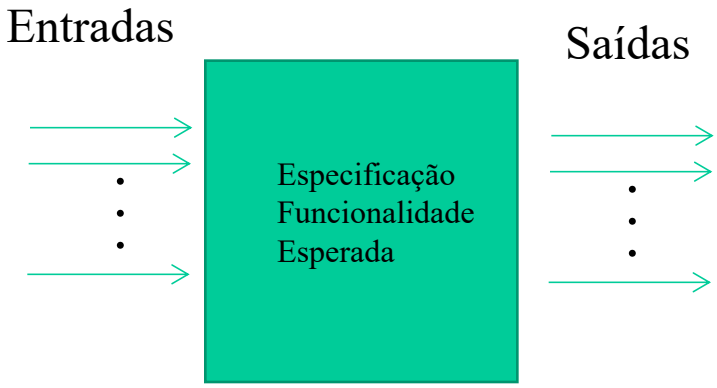
- O processo de criação de sistemas digitais complexos passa, obrigatoriamente, pela **etapa de modelagem**, etapa na qual o funcionamento do sistema pode ser visto em um nível de **abstração** tal que pode ser **avaliado independente de como ele será de fato implementado**

Introdução

- Para fins de análise, do ponto de vista da sua implementação, podemos dizer que um sistema digital é constituído por uma **estrutura hierárquica, com diversos circuitos digitais mais simples, tanto sequenciais quanto combinacionais, interconectados**, cada um dos quais tendo sido projetado e implementado para um fim específico.

Introdução

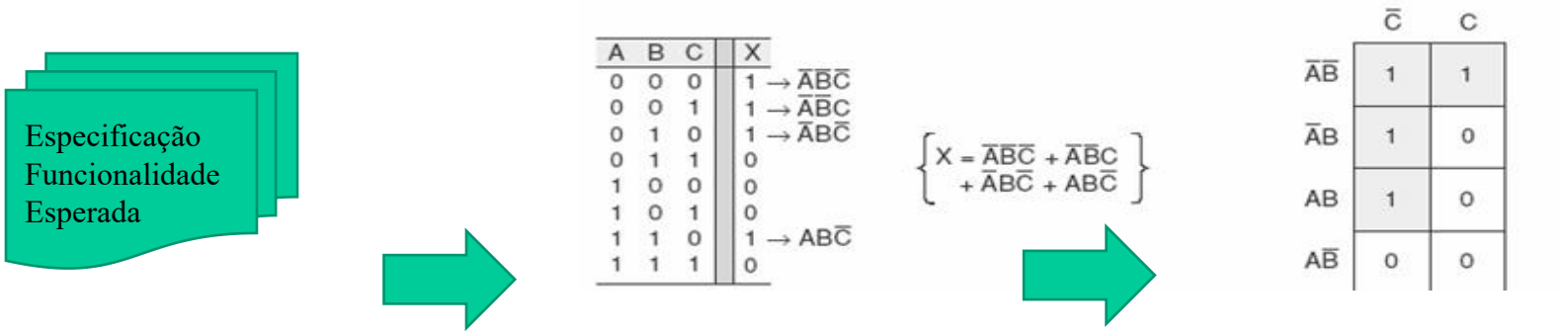
- Quando se pensa na modelagem de um circuito digitai é natural que a primeira opção a ser considerada seja a **Tabela Verdade.**



A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Introdução

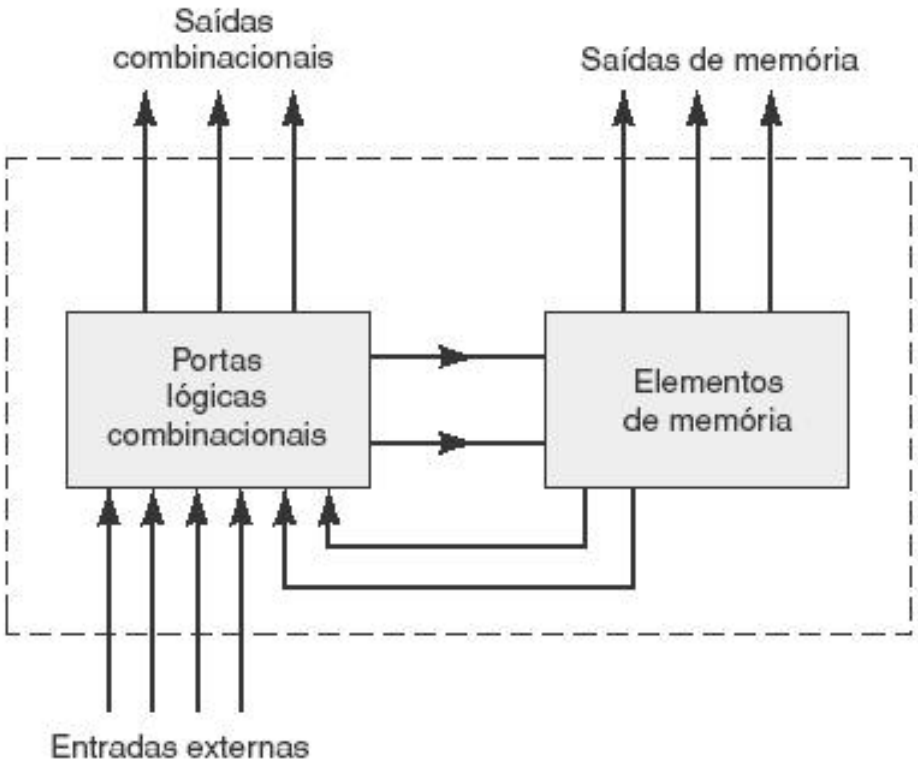
- De fato, em se tratado de um circuito combinacional , fazer a modelagem por meio de uma tabela verdade é algo extremamente simples e prático
- E, o mais interessante, nesta abordagem após a modelagem ser concluída segue-se um o fluxo de projeto único, independente da funcionalidade esperada do circuito que está sendo projetando



Introdução

- Entretanto, por mais bem elaborada que possa ser, uma tabela verdade não é suficiente para modelar de maneira conveniente a resposta esperada de um circuito lógico sequencial, isso porque, diferentemente de um **circuito combinacional,** no qual a resposta do circuito depende exclusivamente da **combinação dos sinais da entrada,** nos **circuitos sequenciais** a resposta depende tanto da **combinação dos sinais da entrada quanto da sequencia de sinais já aplicados** anteriormente.

Circuito lógico Sequêncial



Circuito lógico Sequêncial

UFRPE

- Nos circuitos sequenciais a resposta depende tanto das entradas externas quanto do que está armazenado no elemento de memória,
  - Com isso, a sua resposta pode ser diferente para um mesmo conjunto de entrada a partir da sequencia de entradas que tenha ocorrido antes.

- Nestes caso o mais indicado é utilizar-se de uma outra ferramenta de modelagem denominada de **Máquinas de Estados Finitos** ou apenas **Máquina de Estados**.

Máquina de Estados

- Uma **Máquina de Estados Finita** (**FSM** - do inglês ***Finite State Machine***) é um modelo matemático usado para representar programas de computadores ou circuitos lógicos.
- O conceito é concebido como uma máquina abstrata que deve sempre estar em um de um número finito de estados. A máquina está em apenas um estado por vez, este estado é chamado de estado atual.

Máquina de Estados

- Um estado armazena informações sobre o passado, isto é, ele reflete as mudanças ocorridas desde a inícialização do sistema até o momento presente.
- Um evento é um estímulo que causa uma mudança de estado e pode ser descrito como uma condição que precisa ser realizada para que a transição entre estados ocorra.
- Uma ação é uma atividade que deve ser realizada quando determinada condição (Estado/Evento) é atendida.

Máquina de Estados

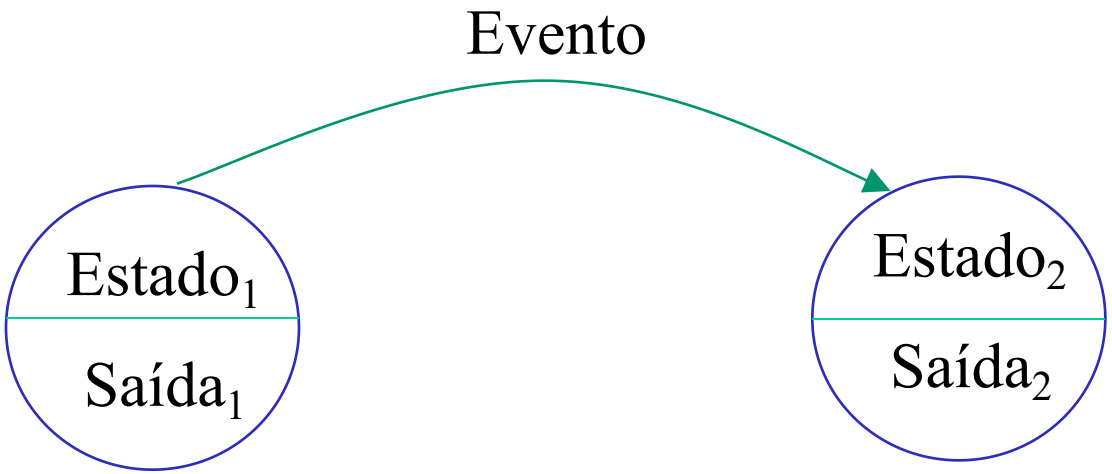
- De uma maneira mais simplista, uma máquina de estados pode ser vista como um diagrama gráfico que representa os diferentes estados que um sistema poderá assumir ao longo do tempo.
- Neste diagrama também são representados os eventos que levam o sistema de um estado a outro e as respostas geradas pelo sistema em função do estado em que se encontra ou em fução da transição de um estado para outro.

Tipos de Máquinas de Estados

- Existem basicamente dois tipos de máquinas de estados denominadas
  - **Máquinas de Moore**
  - **Máquinas de Mealy**

Máquina de Moore

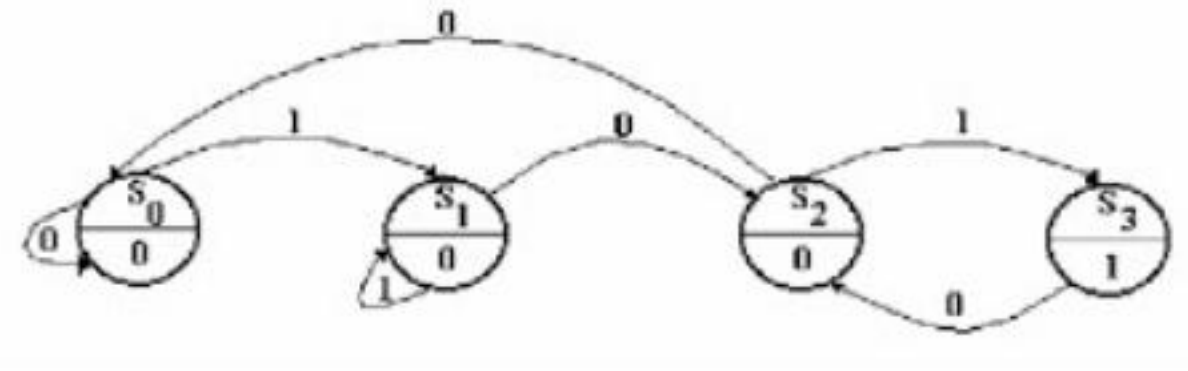
- A saída/ação da máquina depende exclusivamente do estado do sistema*
- A saída/ação ocorre no estado



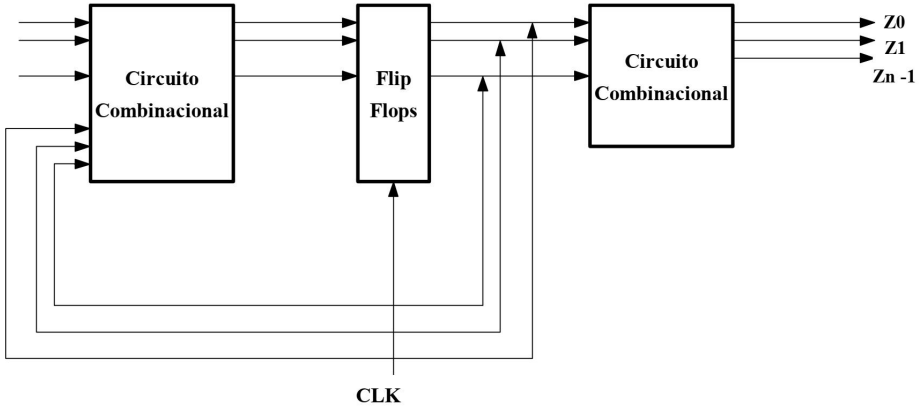
*valor armazenado nos Flip-flops (saídas Q dos FF)

Máquina de Moore

Modelagem

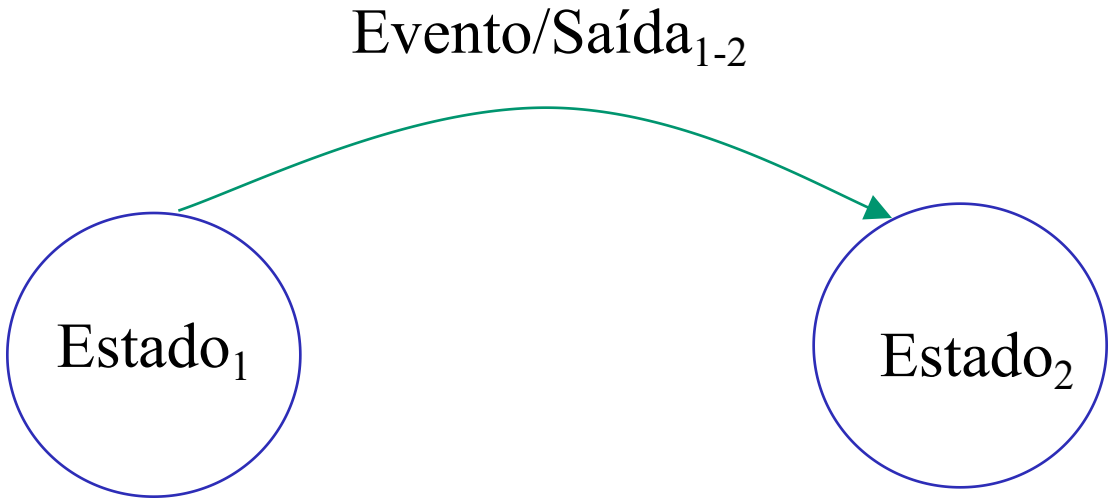


Implementação



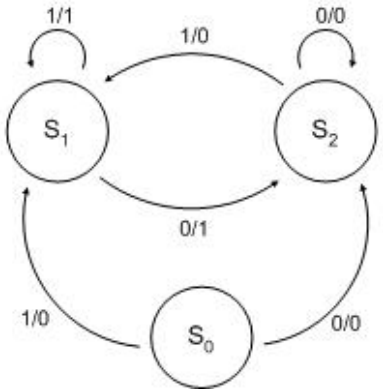
Máquina de Mealy

- A saída/ação depende tanto do estado do sistema quanto dos eventos
- A saída/ação ocorre na transição entre os estados

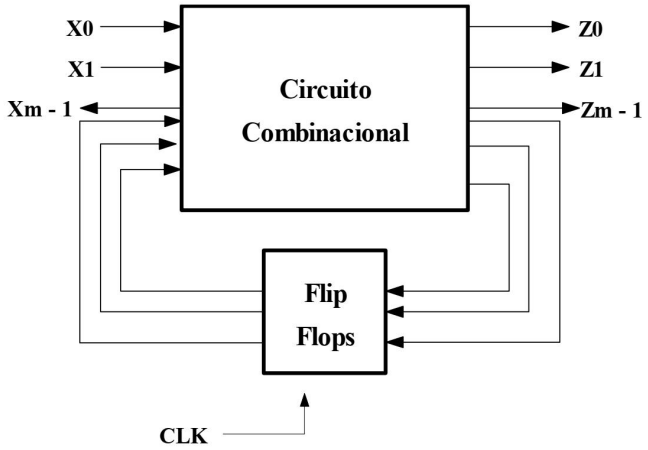


Máquina de Mealy

Modelagem



Implementação



Aplicações das FSM

- As máquinas de Moore são indicadas para modelar sistemas nos quais a resposta (a saída) depende exclusivamente do estado do sistema, aqueles onde os eventos servem apenas para levar a máquina de um estado para outro

- Exemplo:
  - Contadores,
  - Geradores de sequência.

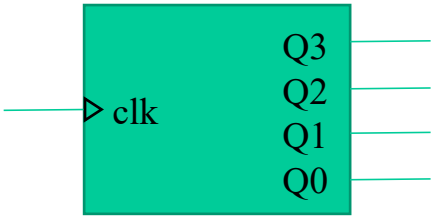
Aplicações das FSM

- As máquinas de Mealy são indicadas para modelar sistemas nos quais a resposta (a saída) depende tanto do estado do sistema quanto dos eventos, ou seja, o evento serve tanto para para levar a máquina de um estado para outro quanto para gerar a sinalização de saída/ação, por este motivo a ação é indicada na transição e não nos estados.
- Exemplo:
  - Autômatos finitos,
  - Circuitos de comunicação,
  - Decodificadores de protocolo

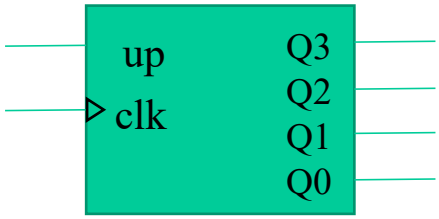
Exemplos típicos de projetos com Máquinas de Estados Moore

Contadores

- São circuitos que contam transiçoes de ciclos de clock. Podem ser contadores simples, que contam apenas em um dos dois modos, ou crescente (Up) ou descrescente (Down), ou contadores cresente e descrescente (Up/Down), que podem contar tanto num modo quanto no outro. Neste último caso, além da entrada de clock e da saida com a contagem, terá ainda uma entrada para indicar se a contagem será crescente ou decrescente.



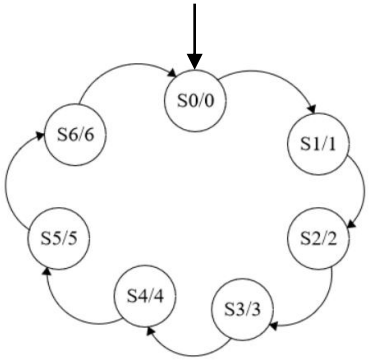
Contador Simples



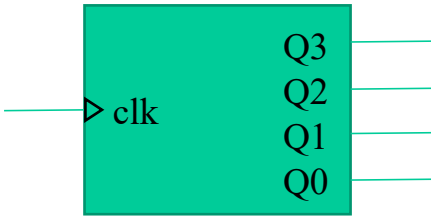
Contador Up/Down

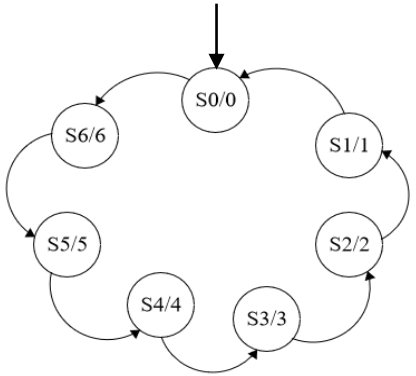
Exemplo de modelagem de contador simples

- Modelagem de um contador simples módulo 7
  - A modelagem consiste simplesmente em associar a cada estado uma contagem e interligar estes estados entre sí com transições vazias, de acordo com a funcionalidade esperada. As transições ocorrem espontâneamente a cada ciclo de clock.



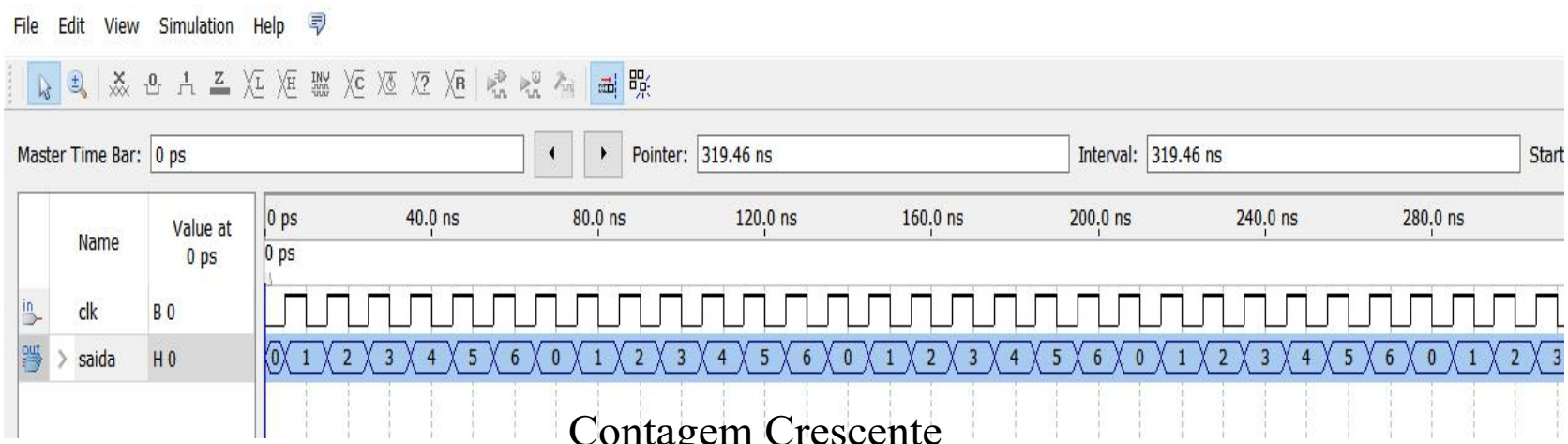
Crescente/Up



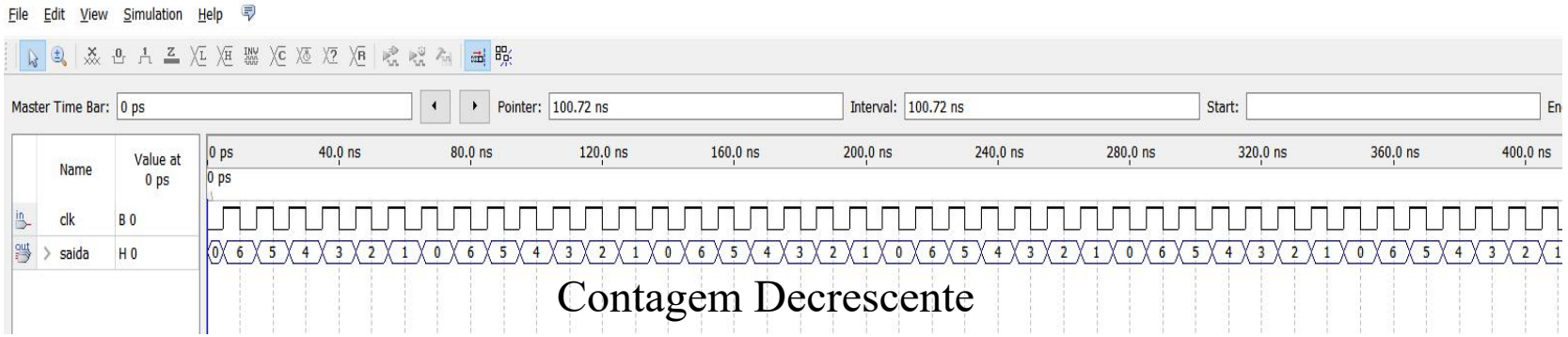


Decrecente/Down

Funcionamento dos contadores simples



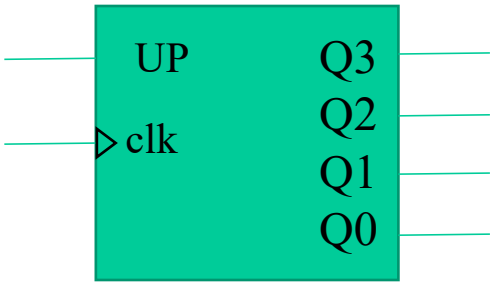
Contagem Crescente



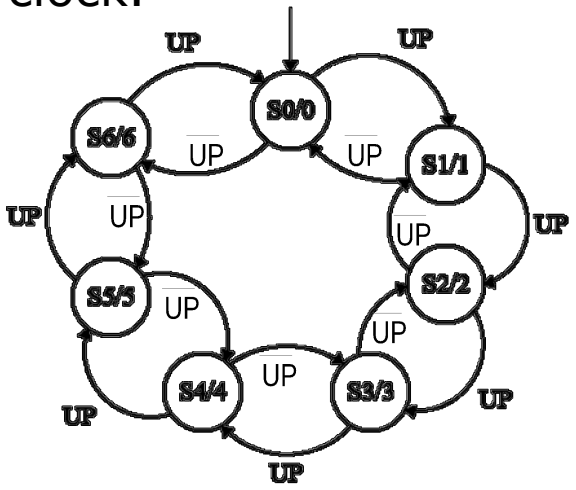
Contagem Decrescente

Exemplo de modelagem de contador crescente/decrescente

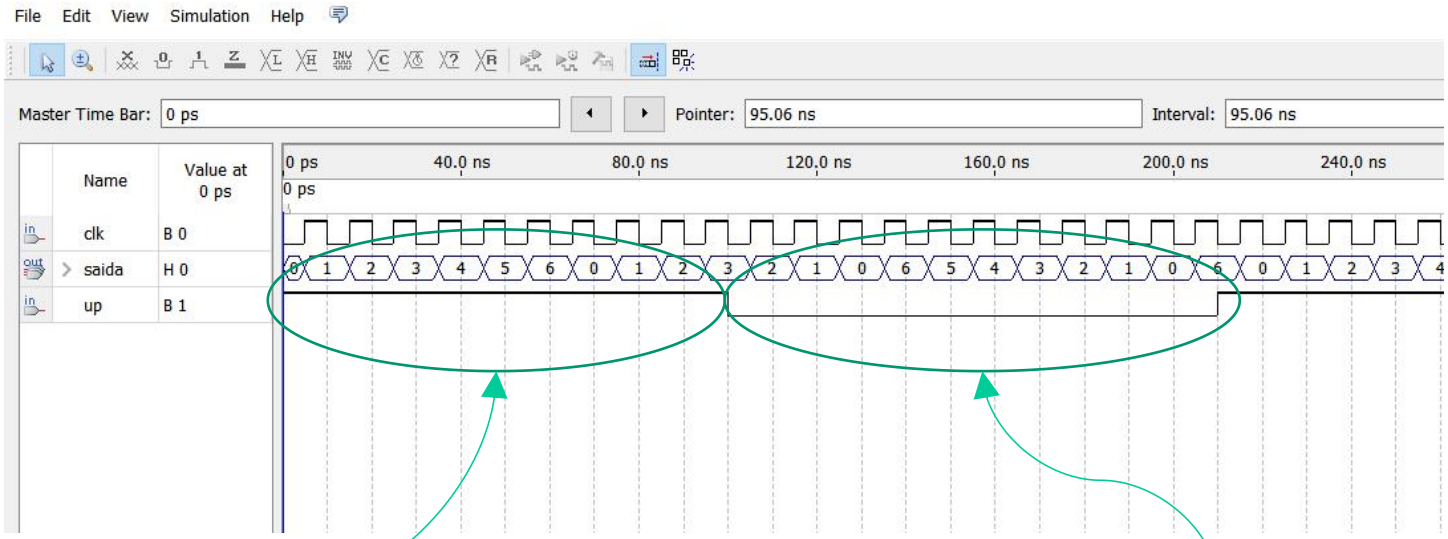
- Modelagem de um contador crescente/decrescente módulo 7
  - Em tudo se assemelha a modelagem dos contadores simples, diferindo apenas que a orientação das transições espontâneas entre os estados vai depender do estado da entrada UP no momento da transição do sinal de clock.



Contador Up/Down



Funcionamento de um contador crescente/decrescente



up=1 ➡ Contagem Crescente

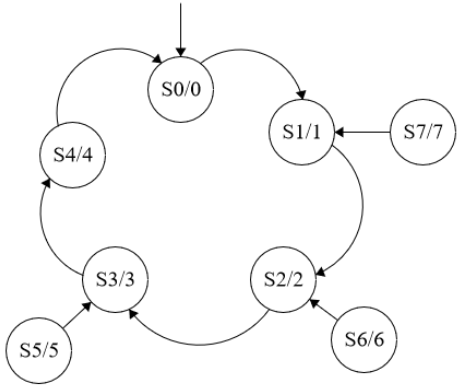
up=0 ➡ Contagem Decrescente

Sistemas com estados não alcançáveis

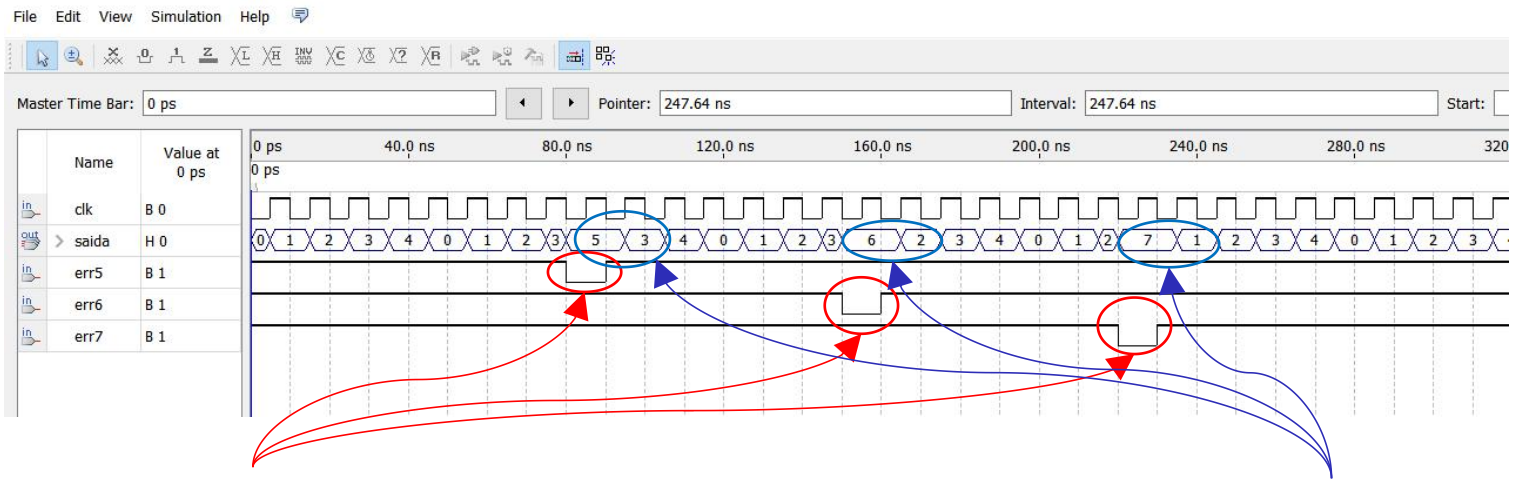
- Existem situações nas quais, como ocorreu nos exemplos anteriores, nem todos os estados possíveis do sistema são alcançáveis, ou seja, nem todos os estados possuem transições que liguem de um outro estado para eles.
  - lembre-se que se forem utilizados n bits (Flip-Flops) para representar os estados de um sistema sempre serão gerados 2^n estados
- Nestes casos ou os estados não alcançáveis podem ser considerados como irrelevantes, podendo ser utilizados para otimizar o circuito, ou inserem-se no sistema transições de recuperação as quais levam destes estados não alcançáveis para um outros estados alcançáveis, nesse caso o sistema é dito como sendo auto-recuperável

Exemplos de contador autorrecuperável

- Projete um contador síncrono, módulo 5, **autorrecuperável** dos estados 5, 6 e 7 respectivamente para os estados 3, 2 e 1.
  - Este projeto é em tudo semelhante ao projeto dos outros contadores, com a única diferença de que os estados não alcançáveis devem aparecer na modelagem para indicar para qual estado o sistema deve ser levado nos casos em que, devido a uma falha, acabar parando em um destes estados indesejáveis.



Funcionamento do contadore autorrecuperável



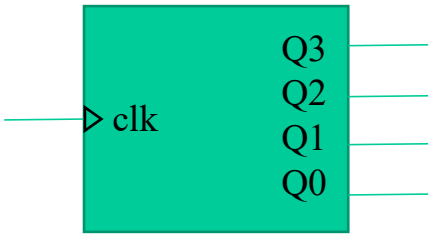
Forçando situações de erro, indo para estados não alcançáveis

Recuperando de situações de erro,

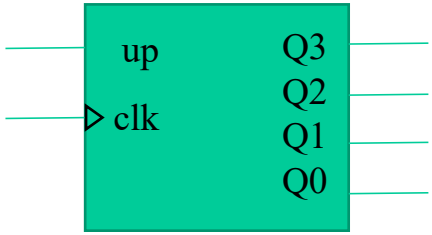
Exemplo de modelagem de gerador de sequência

Geradores de Sequência

- São circuitos que, em vez de contar transiçoes de ciclos de clock, geram um sequencia de valores pré-definidos, um a cada ciclo de clock. A exemplo dos contadores, estes circuitos podem operar tanto nos modos crescente, descrescente ou cresente e descrescente.



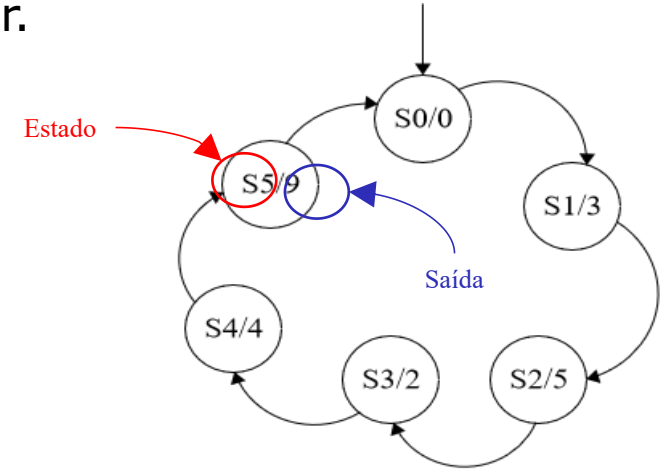
Gerador Simples



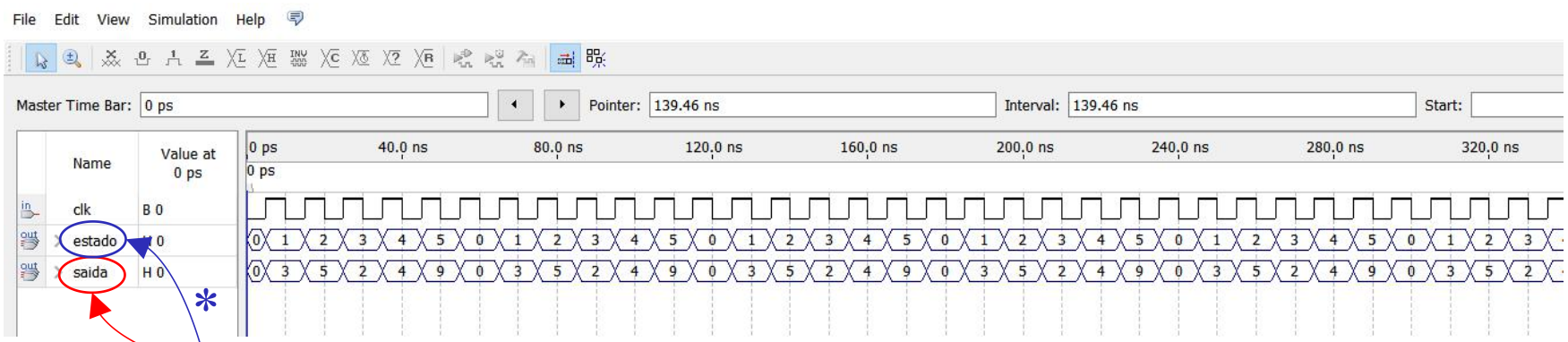
Gerador Crescente e Decrescente

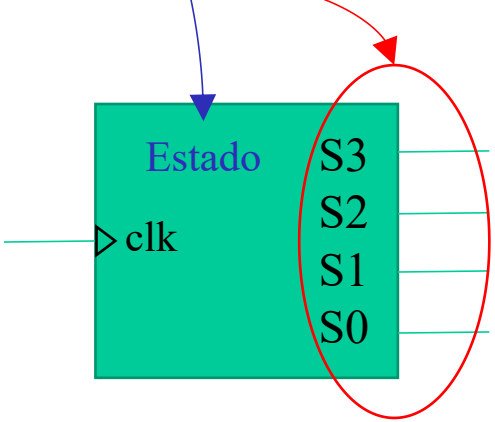
Exemplo de gerador de sequência

- Projete um gerador de sequecia que gere ciclicamente os seguintes valores: 0, 3, 5, 2, 4, 9
  - Este projeto é em tudo semelhante ao projeto de um contador simples, com a diferença que os valores a serem atribuídos à saida a cada ciclo de clock não são mais fruto de uma contagem simples crescente ou decrescente, mas sim resultado dos valores determinados pela especificação do gerador.



Funcionamento do contadore autorrecuperável



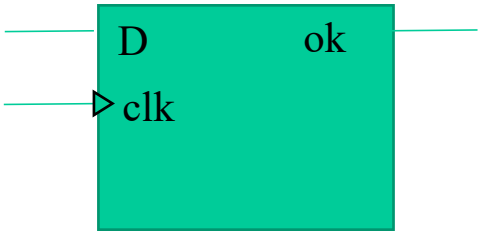


*A informação de estado é **interna ao circuito**, normalmente não é visível externamente

Exemplos típicos de projetos com Máquinas de Estados Mealy

Autômato Finito

- Na nossa bordagem são circuitos que reconhecem uma sequencia binária, sinalizando quando a sequencia é concluída.
- Diferentemente dos contadores e geradores de sequencia, os autômatos além do sinal de clock possuem apenas um bit de entrada, por onde recebem a sequencia binária, e uma saída, por onde sinalizam que a sequencia ffoi reconhecida.



Tipos de autômatos finitos

- Nós trabalhamos basicamente com três tipos de autômatos
  - Aqueles que reconhecem a sequencia um aúnica vez, ou seja, que reconhecem a sequencia **sem repetição**
  - Aqueles que reconhecem quando a sequencia se repete de maneira completa, sem que esteja intercalada com partes de uma sequencia já reconhecida, ou seja, que reconhecem a sequencia **com repetição e sem intercalação**
  - Aqueles que reconhecem quando a sequencia se repete mesmo que de maneira intercalada com partes de uma sequencia já reconhecida, ou seja, que reconhecem a sequencia **com repetição e com intercalação**

Tipos de autômatos finitos

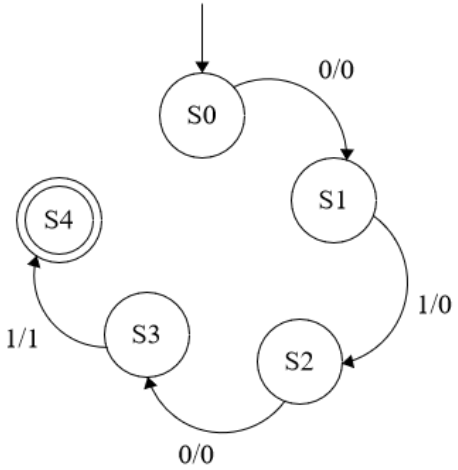
- O que difere os três tipos de autômatos é apenas para onde vai a última transição de estados, aquela na qual a sequência é reconhecida
  - Nos autômatos que reconhecem a sequência **sem repetição** esta transição vai para um **estado de aceitação** o qual não tem trasição para mais nenhum outro estado
  - Nos autômatos que reconhecem a sequência **com repetição sem intercalação** esta transição faz o autômato voltar para o seu **estado inicial**
  - Nos autômatos que reconhecem a sequência **com repetição e com intercalação** esta transição vai para o **estado no qual o mesmo bit que conclui a sequencia possa ser aproveitado para dar seguimento ao reconhecimento de uma nova sequencia intercalada**

Modelagem dos autômatos finitos

- Normalmente a melhor maneira de modelar um autômato é percorrer o chamado "***caminho feliz***", que nada mais é do que percorrer todas as transições que levam a aceitação da sequencia esperada. Observe que em cada estado, a cada ciclo de clock, o autômato pode receber um dado que pode ser zero ou um. Desta forma, percorrer o caminho feliz é reconhecer em cada estado apenas a sequencia de bits que completa a sequencia.
- Uma vez feito isso, deve-se voltar a vizitar todos os estados a fim de completa-los com as transições resultantes de dados que levam a falha na detecção da sequencia, aquelas onde o valor do bit lido não correspondeu ao valor esperado.

Exemplo de modelagem de autômato finito

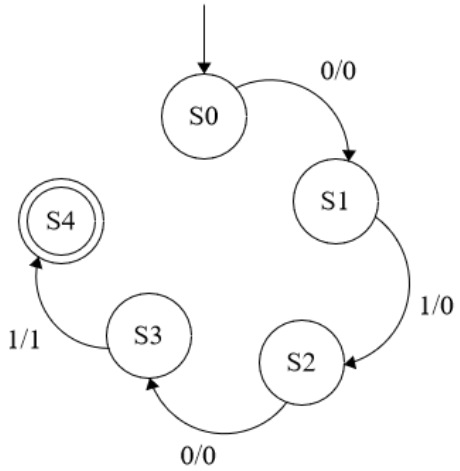
- Projete um autômato que reconheça a sequencia 0101 sem repetição



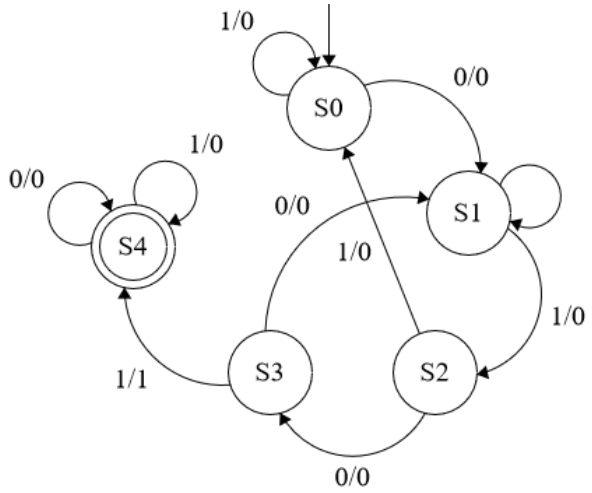
Caminho Feliz

Exemplo de modelagem de autômato finito

- Projete um autômato que reconheça a sequencia 0101 sem repetição

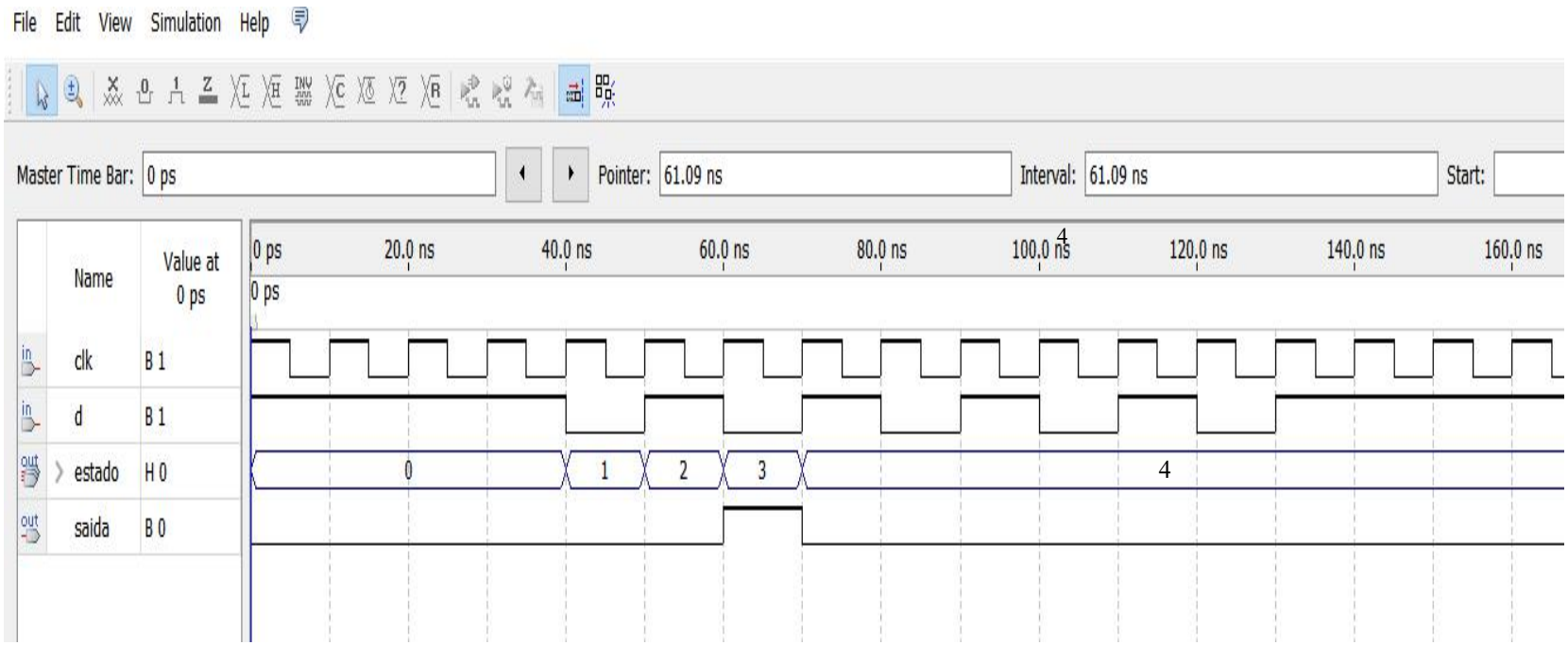


Caminho Feliz



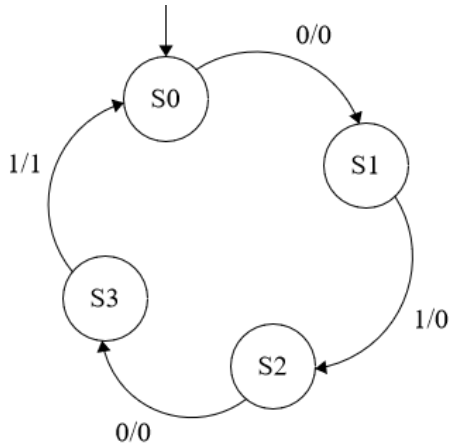
Modelagem Completa

Funcionamento do autômato sem repetição

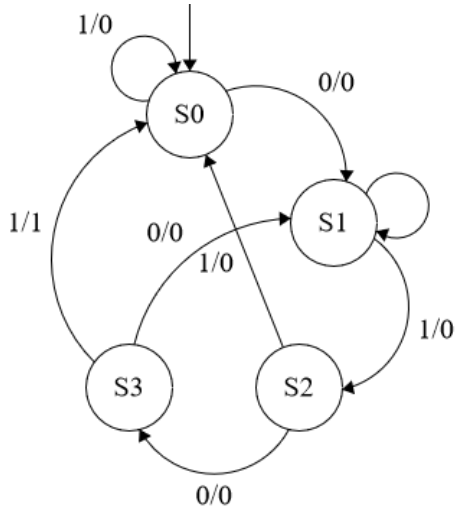


Exemplo de modelagem de autômato finito

- Projete um autômato que reconheça a sequencia 0101 com repetição, sem intercalação

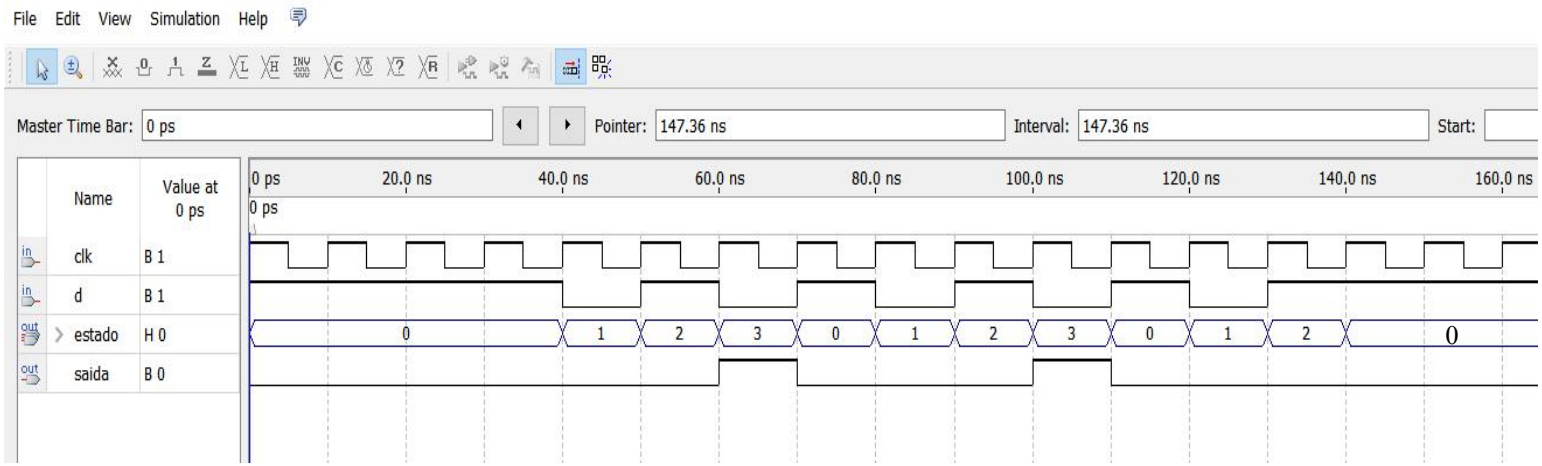


Caminho Feliz



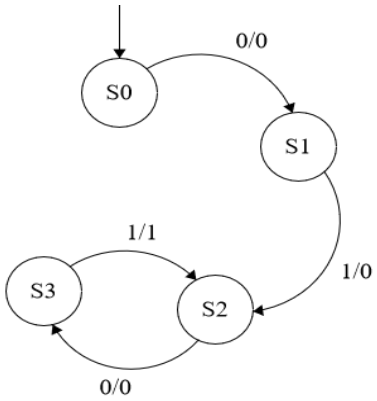
Modelagem Completa

Funcionamento do autômato com repetição sem intercalação

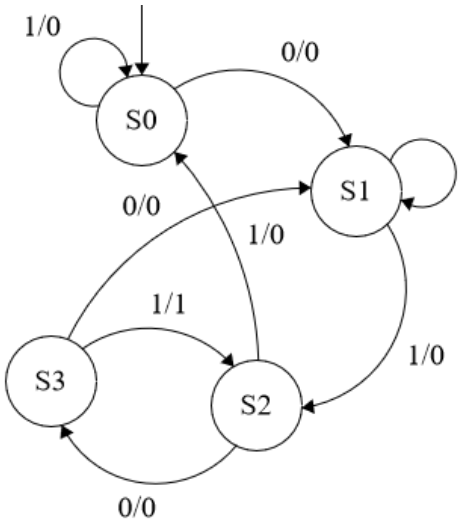


Exemplo de modelagem de autômato finito

- Projete um autômato que reconheça a sequencia 0101 com repetição, com intercalação

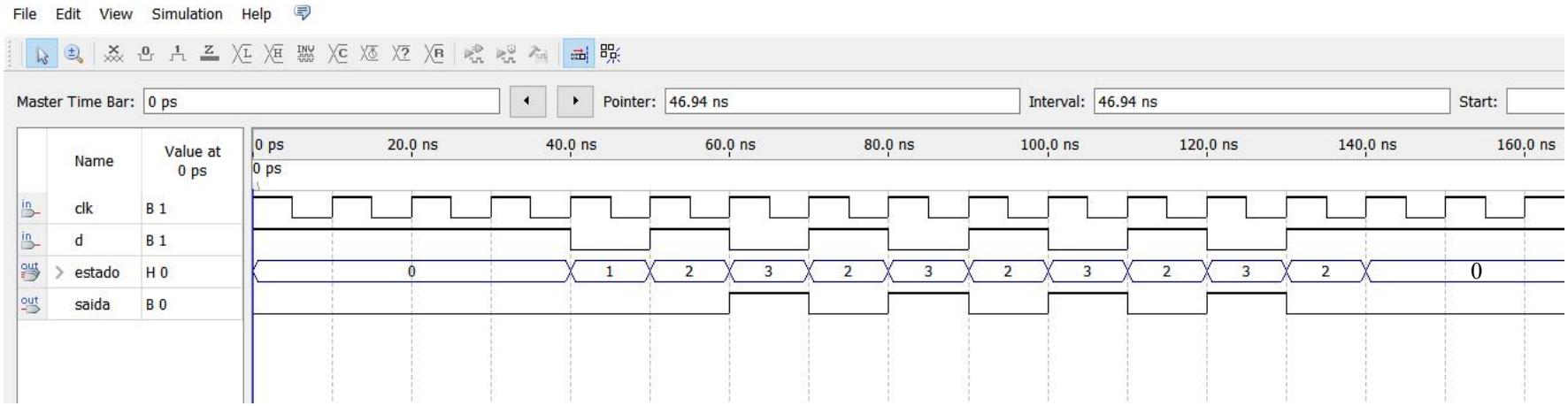


Caminho Feliz



Modelagem Completa

Funcionamento do autômato com repetição com intercalação



Projeto de circuitos sequenciais síncronos

1. **Modelagem em Alto-Nível**

   Transcrever a funcionalidade do sistema no modelo de Máquina de estados escolhido

   I. Percorrer o caminho feliz - registrando todas as transições entre os estados que satisfazem a funcionalidade esperada e indicando a saída/resposta em cada estado ou transição

   II. Escolher o padrão de representação para cada estado

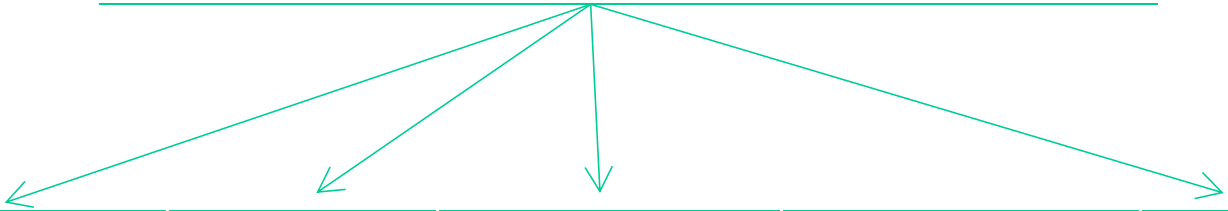
   III. Completar a modelagem com as transições faltantes, aquelas que levam a condições de falha

Projeto de circuitos sequenciais síncronos

2. Transcrição do conteúdo da máquina de estados para uma tabela de transição de estados do sistema
3. Elaboração dos Circuitos de Ativação dos FFs e dos circuito de saída a partir das informações contidas na tabela de transição de estados do sistema

Tabela de Transição de Estados

Valores copiados da Máquina de Estados



Entradas (Eventos)	Estado Atual Q_t	Próximo Estado Q_{t+1}	Conf. Ent. Flip-Flops	Saídas (Resposta)

Valores preenchidos a partir das tabelas de transição do Flip-Flop adotado, com base na mudança de estado exigido em cada bit de dado armazenado

Tabela de Transição de Estados

- A tabela de transição de estados nada mais é que do que uma tabela com uma transcrição ordenada das informações já contidas na máquina de estados.
- Cada linha desta tabela representa o que ocorre em cada estado do sistema quando um evento (valor de entrada) acontece, registrando qual será o próximo estado e o valor que será atribuído à saída.

Tabela de Transição de Estados

- Para completar o preenchimento da tabela de transição, além das informações que são copiadas da máquina de estados, é necessário preencher a coluna que conterá as informações de configuração de cada flip-flop para que no próximo estado ele armazene o bit necessário para reprentar convenientemente o estado do sistema.

Tabelas de transição dos Flip-Flops

Flip-Flop J-K

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Flip-Flop T

Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Flip-Flop D

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

Tabela de Transição de Estados

- Uma vez concluído o preenchimento da tabela de transição de estados do sistema, passa-se a etapa de elaboração dos circuitos de ativação dos Flip-Flops.
- Os circuitos de ativação dos Flip-Flops são responsáveis por configurar as entradas dos Flip-Flops de forma que estes possam armazenar corretamente os bits de informação que levam o sistema de um estado a outro, conforme modelado na máquina de estados do sistema.

Projeto dos circuitos de Ativação dos flip-flops

Entradas (Eventos)	Estado Atual Q_t	Próximo Estado Q_{t+1}	Conf. Ent. Flip-Flops	Saídas (Resposta)
$X_n \dots X_0$	$Q_n \dots Q_0$		$E_n \dots E_0$	$X_n \dots X_0$

Entradas Karnaugh
(Índices Linha-Coluna)

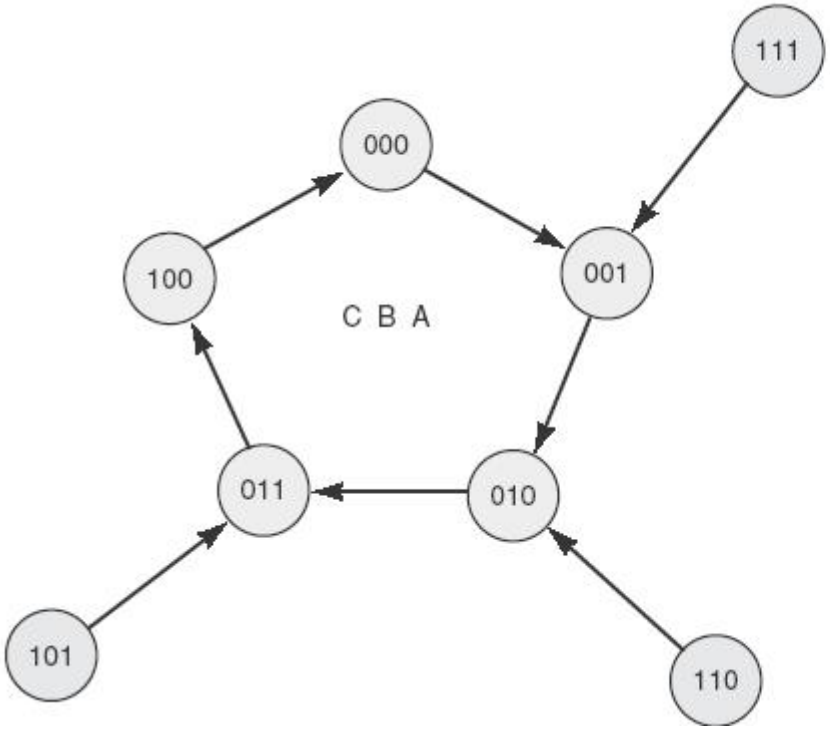
Mapa de Karnaugh

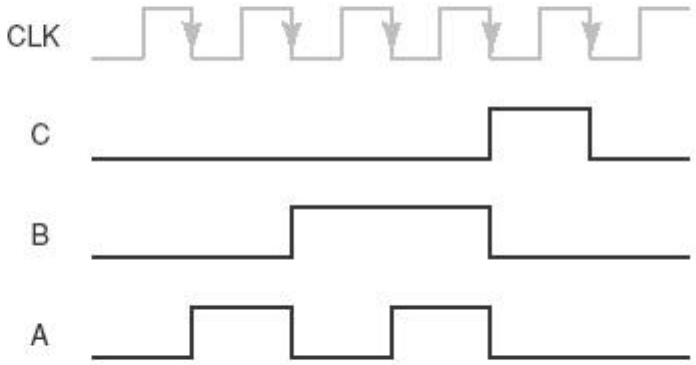
Conteúdo do Karnaugh
Um Karnaugh para cada elemento

Exemplo1- Projeto de um contador Síncrono Módulo 5

- Projete um contador síncrono, módulo 5, auto recuperável dos estados 5, 6 e 7 respectivamente para os estados 3, 2 e 1.

Exemplo1- Projeto de um contador Síncrono Módulo 5





Organizando as tarefas...

1. Definir máquina de estados
2. Definir o padrão de representação dos estados
    1. Preencher os campos estados atual e próximo estado da Tabela de próximo estado(t->t_{+1}) com os valores definidos na máquina de estados
3. Elaborar os circuitos de ativação dos FFs utilizados
    1. Preencher as colunas de configuração dos FF de acordo com o FF que estiver utilizando
    2. Implementar um mapa de Karnaugh para cada entrada de cada Flip-Flop

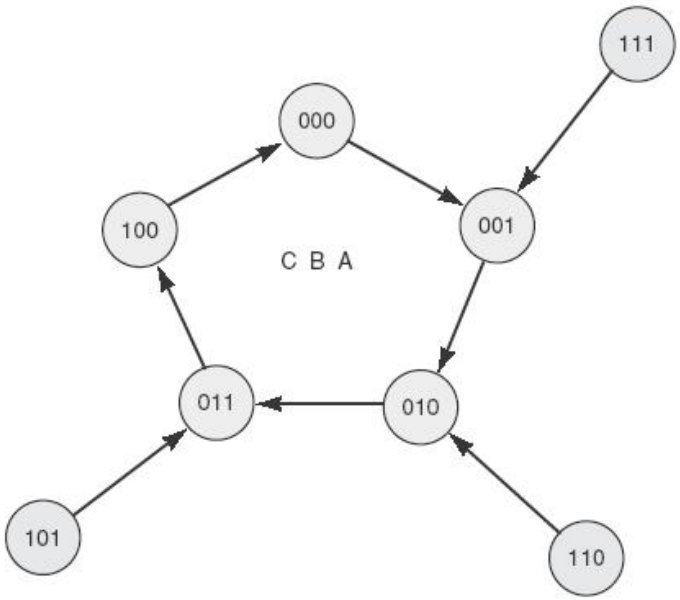
Organizando as tarefas...

UFRPE

3. Circuitos de ativação dos FFs utilizados
    1. Preencher os campos das entradas de controle dos FFs na Tabela de próximo estado com valores que levem os FFs dos seus estados atuais aos próximos estados desejados
    2. Com base nos valores do campo estado atual e das entradas de controle, projetar os circuitos combinacionais que permitam configurar os FFs para assumirem os estados definidos no campo próximo estado

Implementando um exemplo...

Tomando como base o contador síncrono módulo 5 auto-recuperável



Máquina de estados

Atual	Entradas de Controle		Prox.
CBA	....	....	CBA_{+1}
000			001
001			010
010			011
011			100
100			000
101			011
110			010
111			001

Tab. de T. de Prox. Est.

Implementação com FF-D
Tabela de Próx. Estado

Atual	Entradas de Controle			Prox.
CBA	D_C	D_B	D_A	CBA_{+1}
000	0	0	1	001
001	0	1	0	010
010	0	1	1	011
011	1	0	0	100
100	0	0	0	000
101	0	1	1	011
110	0	1	0	010
111	0	0	1	001

Implementação com FF-T
Tabela de Próx. Estado

Atual	Entradas de Controle			Prox.
CBA	T_C	T_B	T_A	CBA_{+1}
000	0	0	1	001
001	0	1	1	010
010	0	0	1	011
011	1	1	1	100
100	1	0	0	000
101	1	1	0	011
110	1	0	0	010
111	1	1	0	001

Implementação com FF-JK
Tabela de Próx. Estado

Atual	Entradas de Controle						Prox.
CBA	J_C	K_C	J_B	K_B	J_A	K_A	CBA_{+1}
000	0	X	0	X	1	X	001
001	0	X	1	X	X	1	010
010	0	X	X	0	1	X	011
011	1	X	X	1	X	1	100
100	X	1	0	X	0	X	000
101	X	1	1	X	X	0	011
110	X	1	X	0	0	X	010
111	X	1	X	1	X	0	001

Geração do circuito de ativação dos FFs

- Analizando a tabela de próximo estado, projetar o circuito combinacional para ativaçao de cada um dos FFs utilizados
- Deve ser projetado um circuito combinacional para cada uma das entradas síncronas, de cada um dos FFs. Ou seja, tendo como base todas as possíveis combinações previstas para o estado atual, associadas a todas as possíveis combinações dos valores das demais entradas do circuito, projetar os circuitos combinacionais que tenham como resposta os valores que levam os FFs aos estados futuros desejados

Geração do circuito de ativação dos FFs

- Este projeto segue o padrão já anteriormente adotado para circuitos combinacionais:
  - tabela verdade, equações booleanas e simplificação algébrica ou,
  - tabela verdade e simplificação pelo método do Diagrama de Karnaugth
- Por ser mais prático, vamos adotar o método do mapa de Karnaugh para proceder a simplificação dos circuitos

simplificação pelo método do Diagrama de Karnaugth

Exemplo: Equação do circuito de geração do sinal J_C

	$\overline{A}$	A
$\overline{C}\ \overline{B}$	0	0
$\overline{C}\ B$	0	1
$C\ B$	X	X
$C\ \overline{B}$	X	X

$J_C = A.B$

Circuito Final com FF-JK

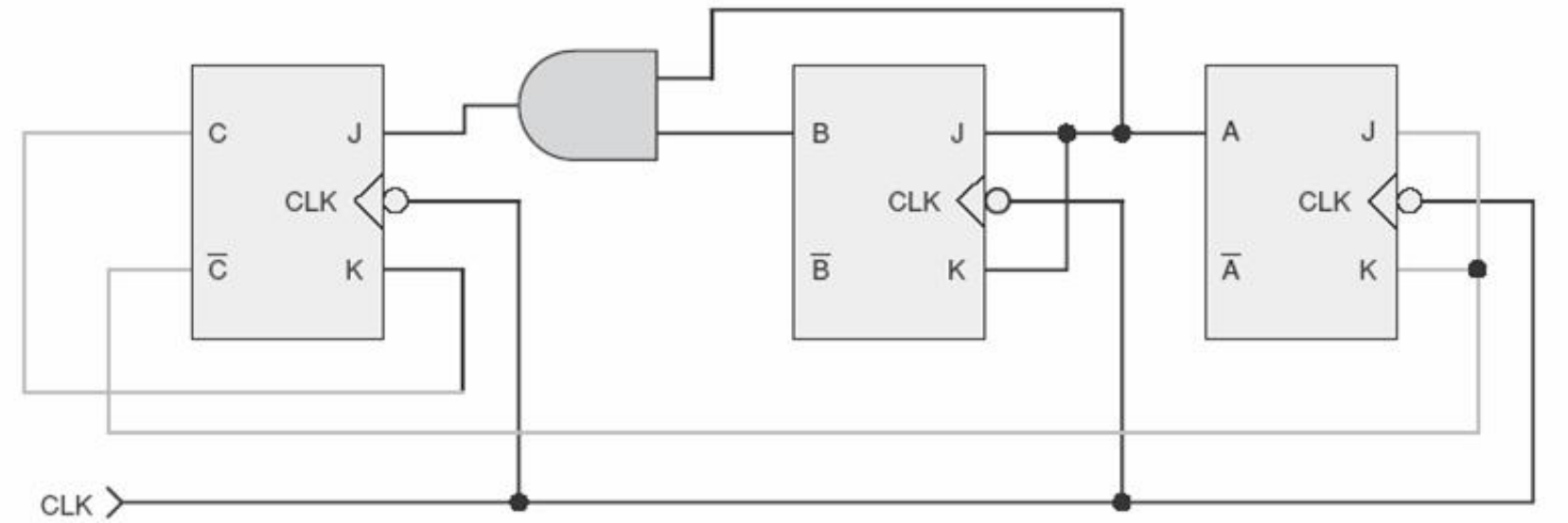


FIGURA 7.23
Contador síncrono com diferentes entradas de controle.