



FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 9

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

Material desenvolvido com apoio de arquivos de apresentação do livro de Frank Vahid

9.1 RTL Design: Introdução

9.2 RTL Design: Elementos do Datapath

Nesta aula: introdução ao "RTL Design" e elementos do datapath.

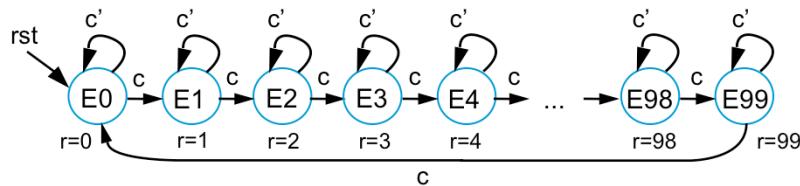
9.1 RTL Design: Introdução

- Até agora: **sistemas digitais cada vez + complexos**
 - Porta lógicas, multiplex., decod., somadores, controladores (FSMs)
- Nas próximas aulas: **como associar esses sistemas para fazer circuitos ainda mais complexos**

Até agora o que vimos foram apenas sistemas digitais isolados cada vez mais complexos, nas próximas aulas vamos aprender como associar todos esses sistemas para fazer circuitos ainda mais complexos.

9.1 RTL Design: Introdução

- **Exemplo:** Contador de 100 com reset **rst**, pause **c** e saída de contagem **r**:
 - Se fazemos usando a abordagem apresentada



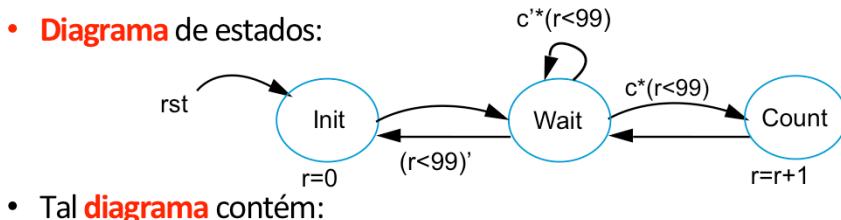
Vamos ter de montar uma FSM de 100 estados!

Este primeiro exemplo mostra um circuito mais elaborado do que os vistos anteriormente: um contador de 100 com reset, pause e saída de contagem. Se tentássemos construí-lo com a mesma abordagem vista em aulas anteriores, teríamos que montar uma FSM de 100 estados, codificar cada um desses estados e todo o resto do processo, algo inviável, ou, pelo menos, pouco prático.

9.1 RTL Design: Introdução

- **Exemplo:** Contador de 100 com reset **rst** e pause **c**:

- **Diagrama** de estados:



- Tal **diagrama** contém:

- Variáveis com **múltiplos bits** (**r**)
 - Operações de “**alto nível**”:
 - Soma de valores com múltiplos bits (**r**)
 - Atribuição (registro) de valores (**r=0**, **r=r+1**)
 - Comparação de grandezas (**r<99**)

- Tem-se então um **diagrama de estados de alto nível**

Mas podemos combinar os sistemas digitais vistos até aqui para obter uma outra máquina de estados muito mais simples e que realiza a mesma função.

Note que neste diagrama temos a variável **r**, de múltiplos bits, e operações de alto nível. Assim, temos um diagrama de estados de alto nível.

9.1 RTL Design: Introdução

- **RTL Design** (Projeto RTL) envolve converter **diagrama de estados de alto nível** em **círculo digital**
- Para tal, dois tipos de **entradas/saídas** são definidos:
 - Entradas/saídas de **controle**: tipicamente de **um bit** (ou poucos bits) representando **evento** ou **estado** do sistema
 - Entradas/saídas de **dados**: geralmente **múltiplos bits** que representam alguma **informação**

O projeto RTL envolve converter o diagrama de estados de alto nível em um circuito digital. Para fazermos isso, separamos as entradas e saídas em dois tipos: de entrada e de saída.

As entradas/saídas de controle representam um evento ou estado do sistema. Normalmente possuem um ou poucos bits.

Já as entradas/saídas de dados representam alguma informação. Normalmente possuem múltiplos bits.

9.1 RTL Design: Introdução

- **RTL Design** (Projeto RTL) envolve converter **diagrama de estados de alto nível** em **circuito digital**
- Dois tipos de **entradas/saídas** são definidos:
 - Entradas/saídas de **controle**: tipicamente de **um bit** (ou poucos bits) representando **evento** ou **estado** do sistema
 - Entradas/saídas de **dados**: geralmente **múltiplos bits** que representam alguma **informação/entidade**
- São definidos também dois blocos:
 - **Bloco de controle**: trata de entradas/saídas de controle
 - **Bloco operacional (datapath)**: trata de entr./saídas de dados

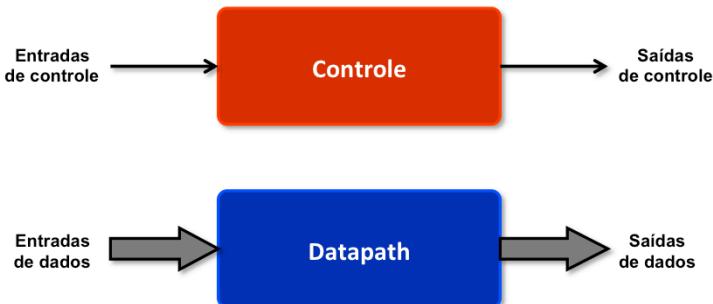
Além disso, são definidos dois blocos: de controle e operacional (ou datapath).

O bloco de controle trata de todas as entradas e saídas de controle.

O datapath trata das entradas e saídas de dados.

9.1 RTL Design: Introdução

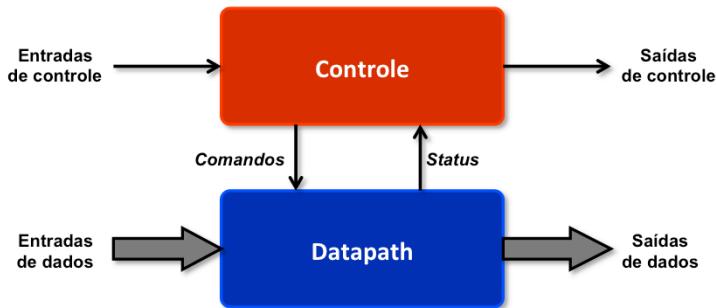
- **RTL Design** (Projeto RTL) envolve converter **diagrama de estados de alto nível** em **círculo digital**
- Com isso, temos a seguinte arquitetura geral:



Com isso, temos a seguinte arquitetura geral incompleta.

9.1 RTL Design: Introdução

- **RTL Design** (Projeto RTL) envolve converter **diagrama de estados de alto nível** em **círculo digital**
- Com isso, temos a seguinte arquitetura geral:



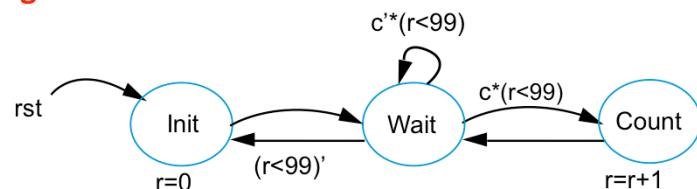
- **Bloco de controle** e **datapath** conversam entre si a partir de sinais de **comando** e **status**

Para completar esta arquitetura geral temos os "comandos" e os "status", através dos quais o bloco de controle e o datapath conversam entre si.

9.1 RTL Design: Introdução

- Para o **contador de 100**:

- **Diagrama de estados de alto nível:**



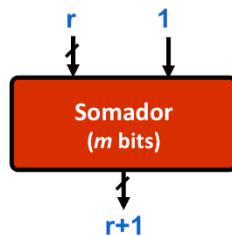
- Operações de “**alto nível**”:

- Contador crescente (**r+1**)
 - Atribuição (registro) de valores (**r=0, r=r+1**)
 - Comparação de grandezas (**r<99**)

As operações de alto nível para o nosso exemplo são as listadas no slide.

9.1 RTL Design: Introdução

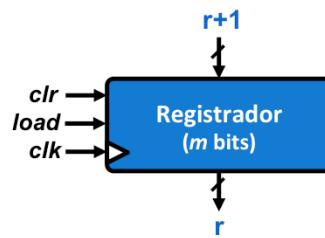
- No **contador de 100**:
 - Soma incremental (**r+1**)
 - Somador da **Aula 4** (veremos outra forma de fazer uma soma incremental na próxima aula)



Para fazer a soma incremental utilizaremos um somador, visto na aula 4.

9.1 RTL Design: Introdução

- No **contador de 100**:
 - Atribuição (registro) de valores ($r=0, r=r+1$)
 - Registrador com FFs Tipo D com *load* e *clear* das **Aulas 7 e Lab**

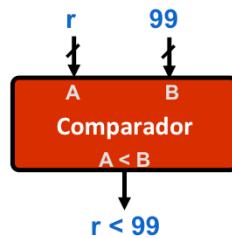


$$\begin{aligned} \text{load} = 1 &\rightarrow r = r+1 \\ \text{clr} = 1 &\rightarrow r = 0 \end{aligned}$$

Para o registro de valores de r utilizaremos um registrador com flip-flops do tipo D, load e clear, visto na aula 7. O load é que possibilitará $r=r+1$ e o clear $r=0$.

9.1 RTL Design: Introdução

- No **contador de 100**:
 - Comparação de grandezas (**r < 99**)
 - Comparador, o qual pode ser projetado usando **Aula 3**

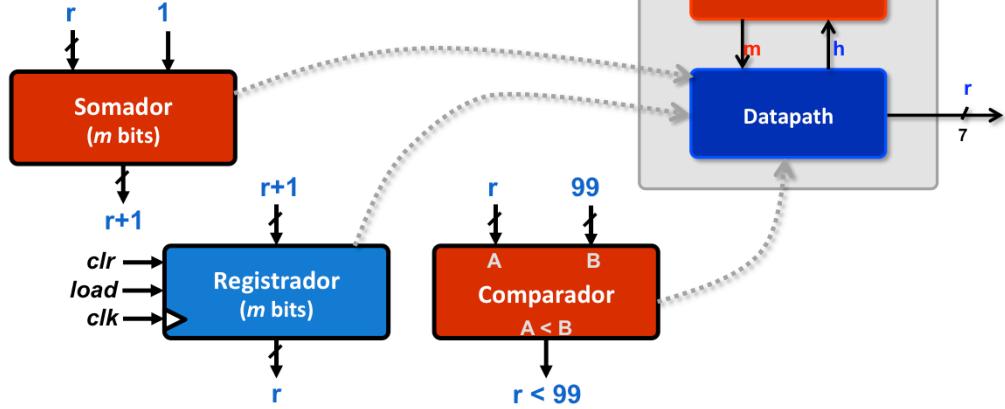


E, finalmente, para a comparação de grandezas utilizaremos um comparador, que pode ser construído usando a aula 3.

9.1 RTL Design: Introdução

- No **contador de 100**:

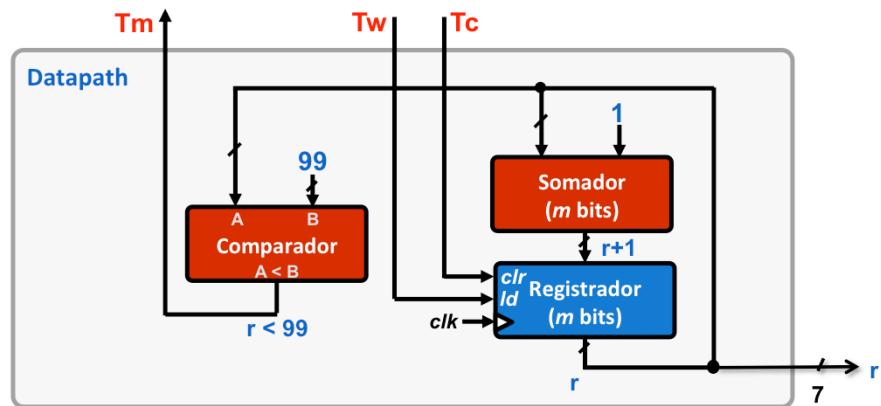
- Associando os elementos que fazem operações de alto nível, obtém-se o **datapath**



Associando esses elementos que fazem as operações de alto nível, obtemos o datapath.

9.1 RTL Design: Introdução

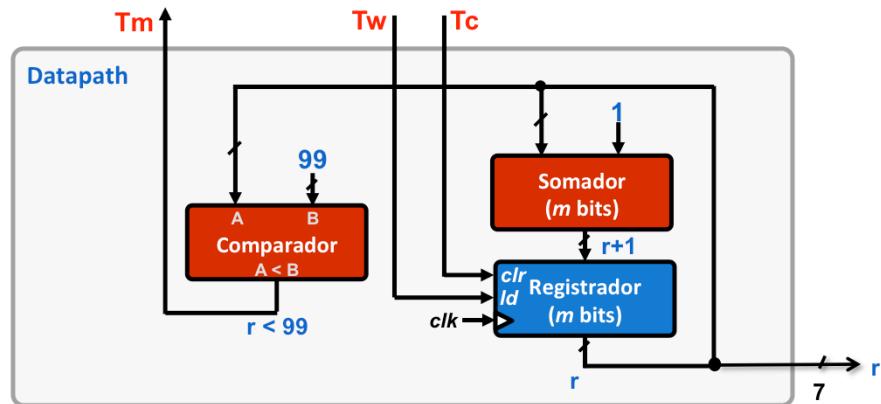
- No **contador de 100**:
 - Associando os elementos que fazem operações de alto nível, obtém-se o **datapath**



Tais são as conexões entre eles. Veja que **Tm** é um sinal de "status" e **Tw** e **Tc** são ambos "comandos", enquanto **r** é uma saída de dados. Não há entrada de dados.

9.1 RTL Design: Introdução

- No **contador de 100**:

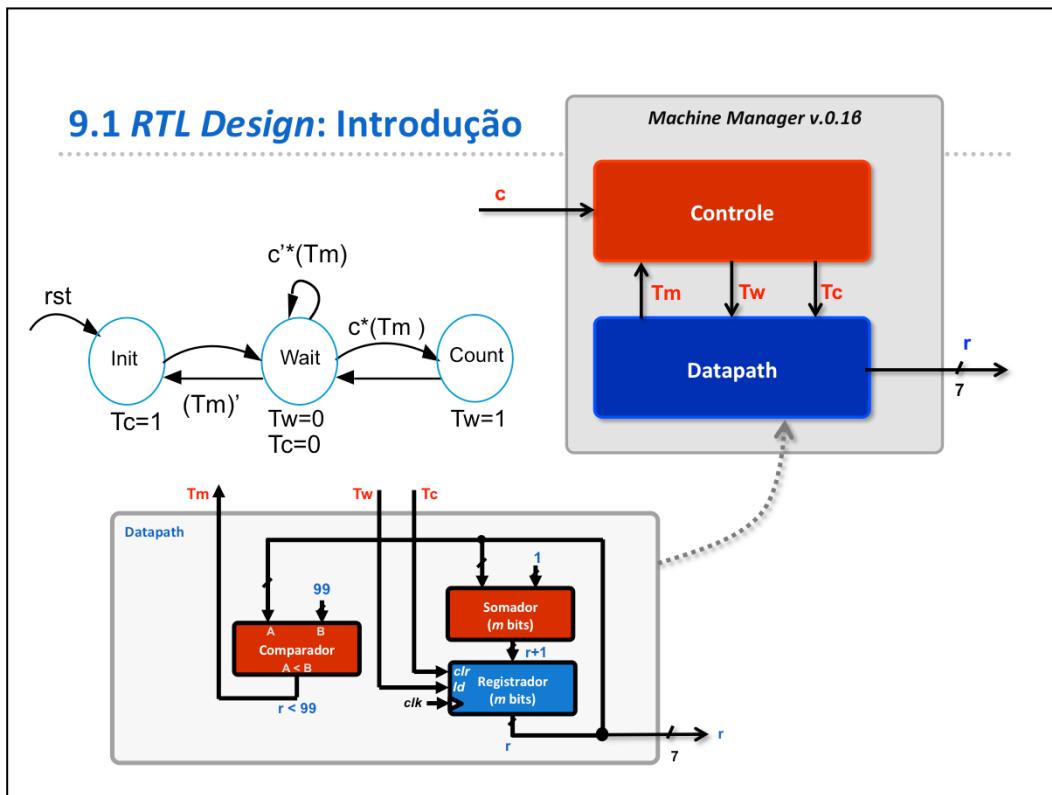


Tm: resultado de $r < 99$

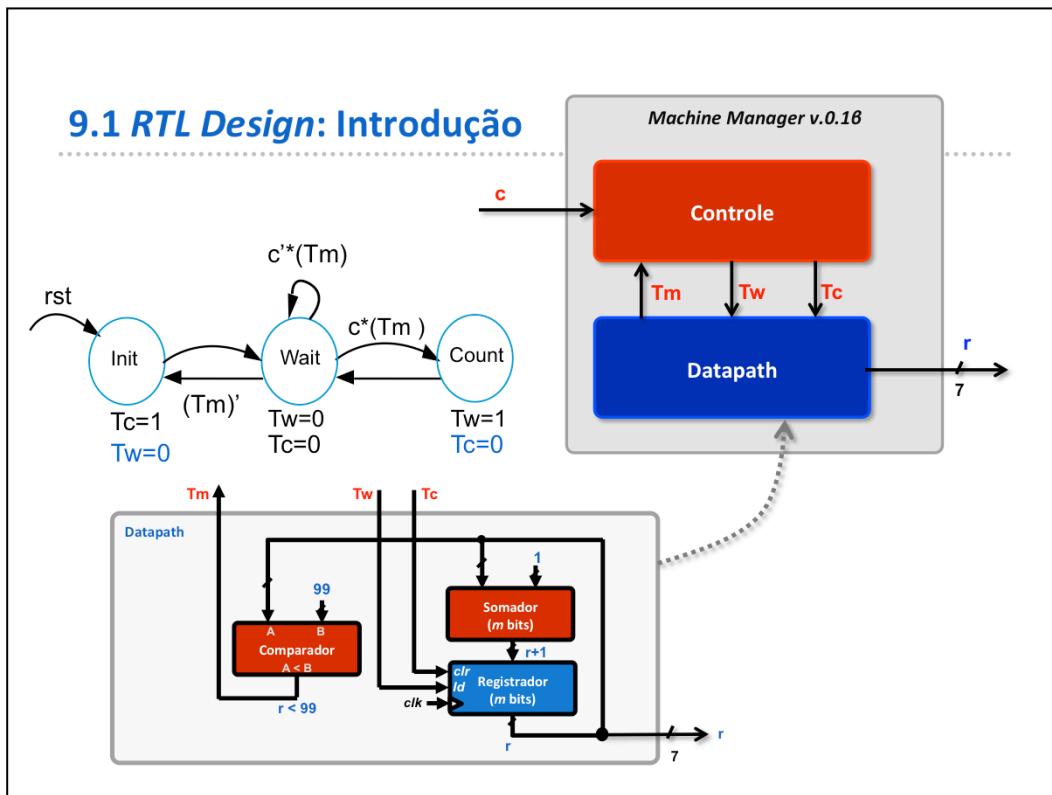
Tw = 1 $\rightarrow r = r+1$

Tc = 1 $\rightarrow r = 0$

Veja que quando $Tw=1$, load =1 e, consequentemente, $r=r+1$. Já quando $Tc=1$, clear=1 e, assim, $r=0$. Tm será 1 enquanto $r<99$, sinalizando que a contagem deve continuar.



Assim, podemos tirar as operações de alto nível do diagrama de estados e transformá-lo em um que poderá compor o bloco de controle. Veremos como fazer tal boco na próxima aula.



Observe que em "Init" $Tw=0$ e em "Count" $Tc=0$.

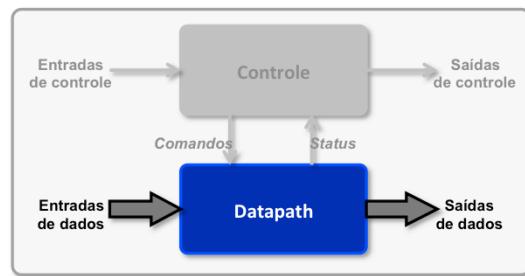
9.1 RTL Design: Introdução

9.2 RTL Design: Elementos do Datapath

Por ora, vamos ver mais alguns elementos que podem compor um datapath.

9.2 RTL Design: Elementos do Datapath

- Exemplos de **elementos do bloco operacional**, ou seja, elementos utilizados para construir um **datapath**:
 - Registradores
 - Registradores de deslocamento
 - Comparadores
 - Somadores
 - Contadores
 - Multiplicadores
 - Subtratores
 - ALUs (unidades lógico-aritméticas)

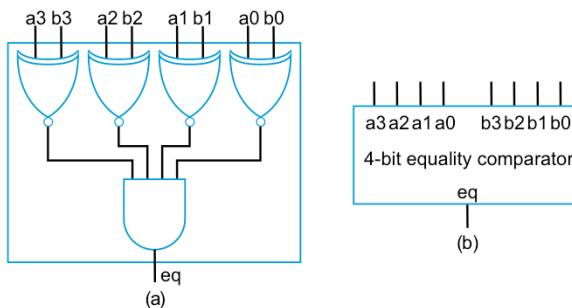


Entre eles estão: registradores, registradores de deslocamento, comparadores, somadores, contadores, multiplicadores, subtratores e ALUs.

9.2 RTL Design: Elementos do Datapath

Comparadores

- **Comparador de igualdade** de N bits:
 - Saída é igual a 1 se dois números de N bits são iguais
 - Ou seja, saída é igual se todos os bits são iguais
- Como saída de **XNOR** é 1 se os 2 bits de entrada **são iguais**, comparador de 4 bits pode ser obtido fazendo:



Comecemos olhando para os comparadores, mais especificamente um compadore de igualdade de N bits. Para comparar dois números basta apenas compará-los bit a bit. Para tanto utilizamos portas XNOR bit a bit, já que a sua saída é 1 se os dois bits na sua entrada forem iguais (veja a tabela verdade), e, em seguida, uma porta AND, produzindo 1 apenas se todos os bits comparados forem iguais.

9.2 RTL Design: Elementos do Datapath

Comparadores

- **Comparador de magnitude** de N bits:
 - Indica se $A > B$, $A = B$ ou $A < B$, sendo A e B dois números de N bits
 - Como projetar?

Dentro dos comparadores temos também comparadores de magnitude de N bits, uma versão estendida do comparador de igualdade.

9.2 RTL Design: Elementos do Datapath

Comparadores

- Comparador de magnitude de N bits:
 - Indica se $A > B$, $A = B$ ou $A < B$, sendo A e B dois números de N bits
 - Como projetar?
 - Basta considerar como a comparação é feita “manualmente”:

$A=1011 \quad B=1001$

1011 1001 Igual

1011 1001 Igual

1011 1001 Maior

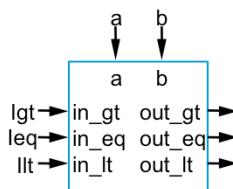
Então A > B

Para verificar se A é maior ou menor do que B , basta proceder da mesma forma que fazemos manualmente. Veja o exemplo no slide.

9.2 RTL Design: Elementos do Datapath

Comparadores

- Comparador de magnitude de N bits:
 - Indica se $A > B$, $A = B$ ou $A < B$, sendo A e B dois números de N bits
 - Portanto, basta projetar um bloco:
 - $out_{gt} = in_{gt} + (in_{eq} * a * b')$ [greater than]
 - $out_{lt} = in_{lt} + (in_{eq} * a' * b)$ [less than]
 - $out_{eq} = in_{eq} * (a XNOR b)$ [equals]



Considerando isso, basta projetar um bloco que seja responsável pela comparação entre dois bits, um de cada número, isso de acordo com as funções apresentadas no slide.

Atenção:

out_{gt} = output greater than

in_{gt} = input greater than

out_{lt} = output less than

in_{lt} = input less than

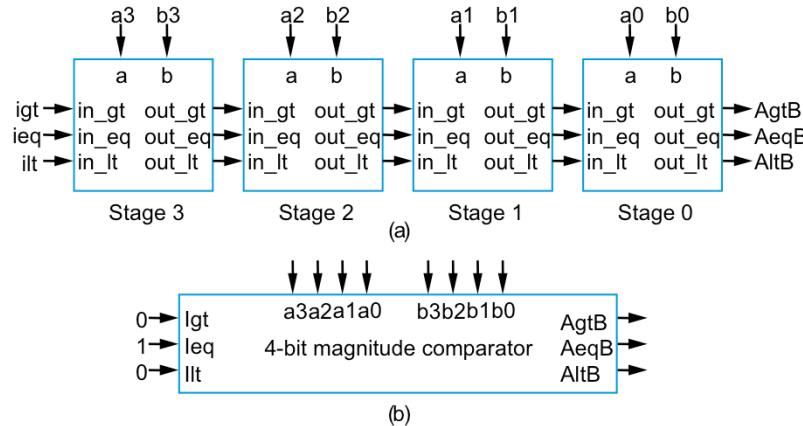
out_{eq} = output equal

in_{eq} = input equal

9.2 RTL Design: Elementos do Datapath

Comparadores

- **Comparador de magnitude** de N bits:
 - Indica se $A > B$, $A = B$ ou $A < B$, sendo A e B dois números de N bits
 - Então:

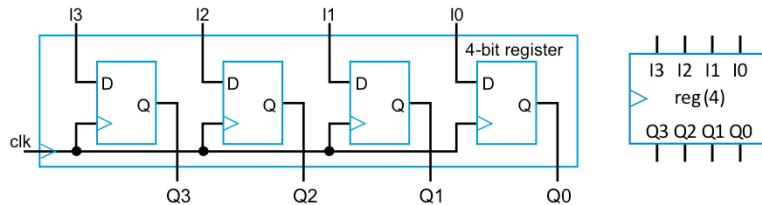


Por fim, ligamos tais blocos de acordo com o necessário (a) e assim temos o comparador de magnitude (b). Veja que $lgt=0$, $leq=1$ e $llt=0$ por conta das funções definidas para os blocos.

9.2 RTL Design: Elementos do Datapath

Registradores

- **Registrador básico:** armazena um certo número de bits



- Esse registrador é gravado **em todo ciclo de clock**
 - Como modificá-lo para que ele seja gravado apenas em determinados ciclos indicados por um entrada **load**?

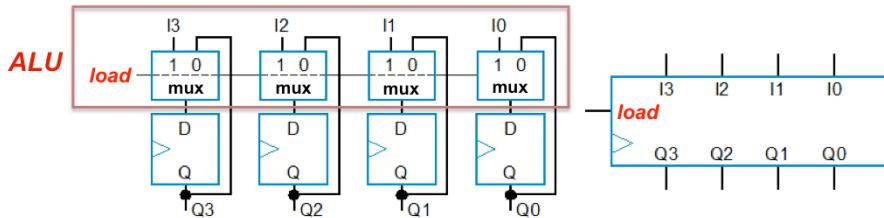
Partindo agora para os registradores, temos neste slide uma estrutura básica de um registrador, que é formado por um agrupamento de flip-flops (normalmente do tipo D) que possuem um sinal de clock em comum. A quantidade de flip-flops é determinada pela quantidade de bits a serem armazenados neste elemento de memória.

Este registrador é gravado em todo ciclo de clock, mas como modificá-lo para que ele seja gravado apenas em determinados ciclos indicados por uma entrada load?

9.2 RTL Design: Elementos do Datapath

Registradores

- **Registrador com carga paralela:**
 - Armazena bits somente quando **load = 1**.
 - Obtido adicionando um **mux** em cada entrada do **reg. básico**

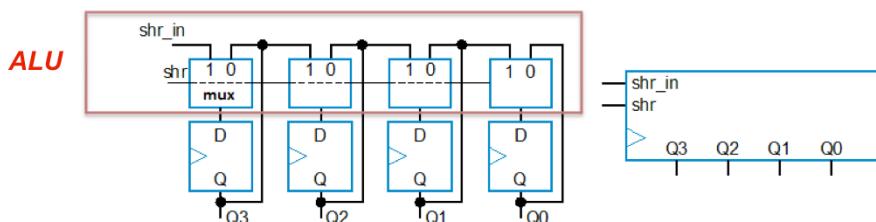


Para isso precisamos adicionar um multiplexador 2:1 em cada entrada do registrador básico, para o qual a entrada de seleção é o load. Este é um registrador com carga paralela.

9.2 RTL Design: Elementos do Datapath

Registradores de Deslocamento

- Registrador de deslocamento à direita (shr) que faz deslocamento de **shr_in** somente se entrada **shr = 1**:



Realizando algumas modificações no último registrador podemos obter um registrador de deslocamento. Este é um registrador de deslocamento à direita, o qual faz o deslocamento de **shr_in** somente se **shr=1**.

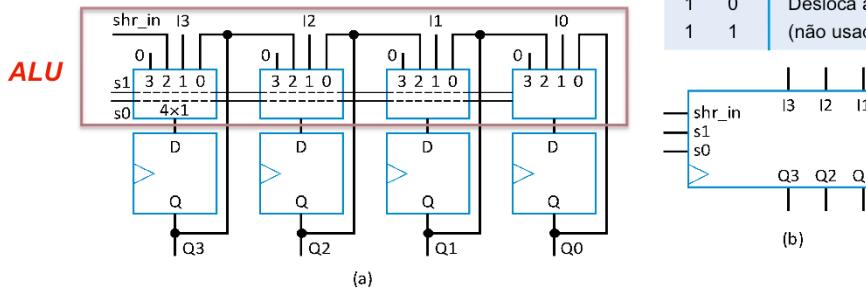
9.2 RTL Design: Elementos do Datapath

Registradores de Deslocamento

- Registrador multifunção:
 - Muitos registradores tem múltiplas funções
 - Podem ser facilmente projetados usando muxes

Funções:

• Exemplo:



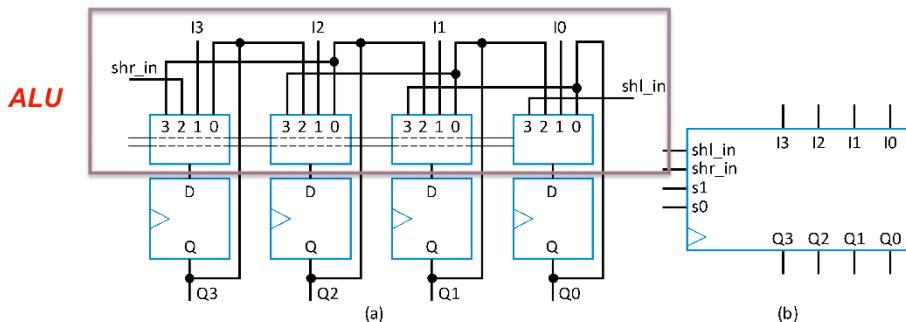
Há também os registradores multifunção. O registrador deste slide faz a carga paralela de acordo com um sinal de load e desloca à direita. Veja que para construí-lo apenas combinamos o registrador com carga paralela e o registrador de deslocamento à direita com a ajuda de um multiplexador 4:1.

9.2 RTL Design: Elementos do Datapath

Registradores de Deslocamento

- Outro Exemplo:

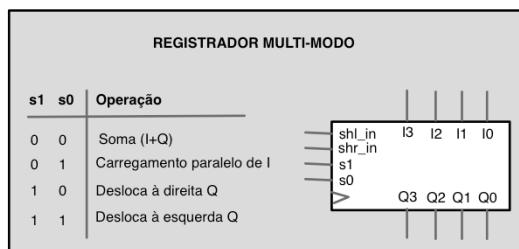
s1	s0	Operação
0	0	Mantém valores atuais
0	1	Carga paralela
1	0	Desloca à direita
1	1	Desloca à esquerda



Para adicionar o deslocamento à esquerda podemos aproveitar o mesmo multiplexador, fazendo ligações entre os registradores contrárias às da função de deslocamento à direita.

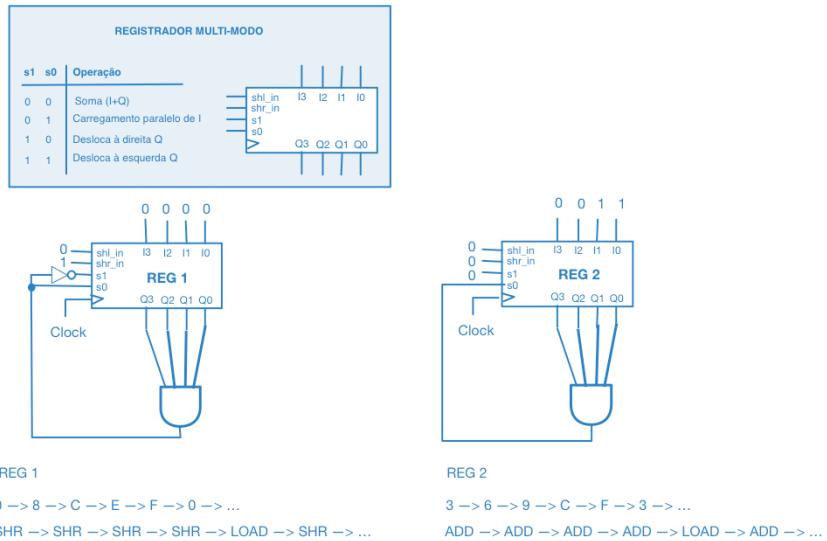
PROBLEMAS

Problema 9.5. Pretende-se implementar um circuito que implemente o padrão de contagem em hexadecimal $03 \rightarrow 86 \rightarrow C9 \rightarrow EC \rightarrow FF \rightarrow 03 \rightarrow 86 \dots$, correspondente a um sinal de 8 bits (b_0 a b_7). Este padrão corresponde à contagem de 5 estados, que se repetem ao longo do tempo. Utilizando o mínimo de lógica combinatória adicional, ligue dois registros Multi-modo apresentado na Figura de modo a implementar o circuito gerador deste padrão.



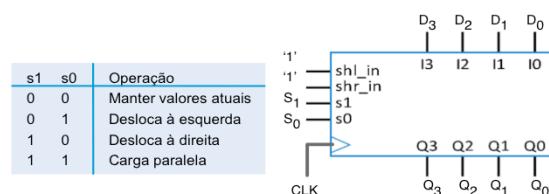
PROBLEMAS

Solução:

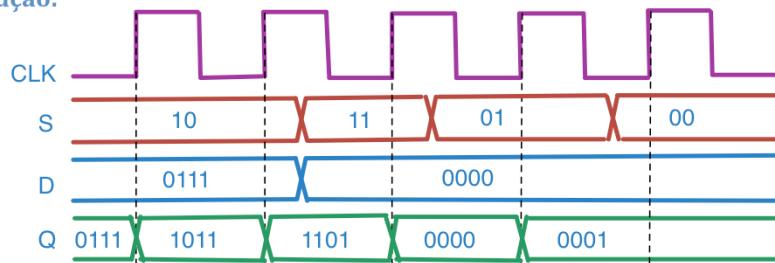


PROBLEMAS

Problema 9.6. Obtenha a forma de onda para a saída Q de 4-bits do registrador Multi-modo apresentado.



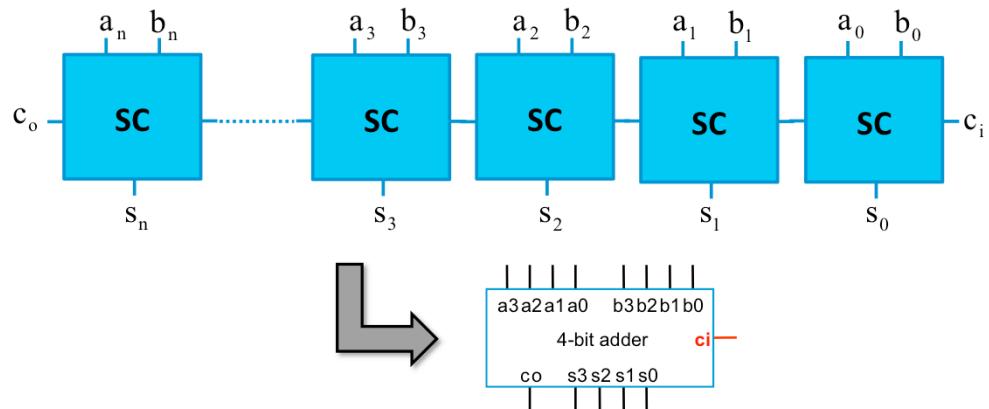
Solução:



9.2 RTL Design: Elementos do Datapath

Somadores

- Vistos em aulas anteriores:

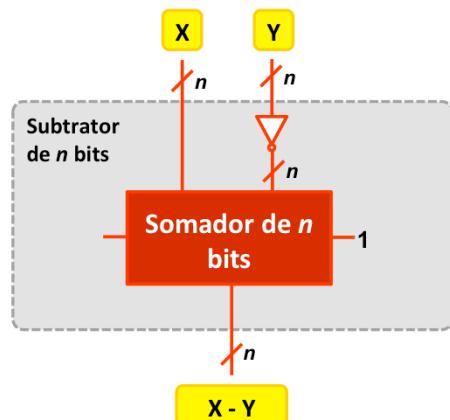


Os somadores são os mesmos vistos na aula 4.

9.2 RTL Design: Elementos do Datapath

Subtratores e Somadores/Subtratores

- Vistos em aulas anteriores:

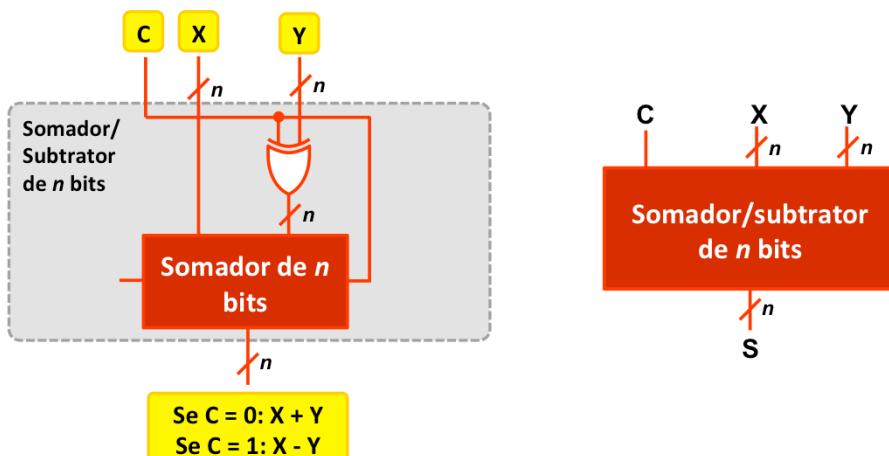


Bem como o subtrator.

9.2 RTL Design: Elementos do Datapath

Subtratores e Somadores/Subtratores

- Vistos em aulas anteriores:



E como o somador/subtrator.

9.2 RTL Design: Elementos do Datapath

Deslocadores

- Operação de **deslocamento** (ex. deslocar 0011 para esquerda resulta em 0110) é útil para:
 - Manipular bits
 - Converter dados seriais para paralelos (geralmente usando registradores de deslocamento)
 - Fazer **divisões** ou **multiplicações** por potências de 2
 - Deslocar n vezes para esquerda = multiplicação por 2^n
 - Deslocar n vezes para direita = dividir por 2^n

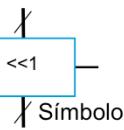
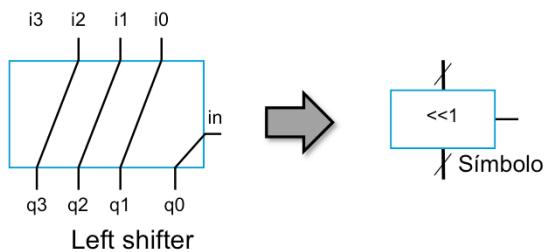
Vejamos então os deslocadores.

A operação de deslocamento, apesar de simples, é muito útil para a manipulação de bits, conversão de dados seriais em paralelos e para fazer divisões ou multiplicações por potências de 2.

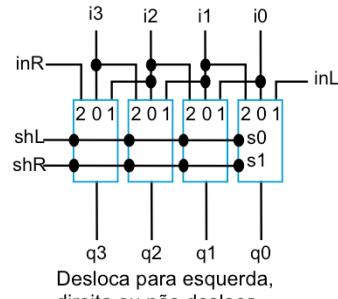
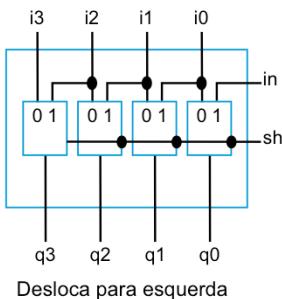
9.2 RTL Design: Elementos do Datapath

Deslocadores

- Circuito pode ser muito simples:



- Outros:



O circuito pode tanto cumprir somente a função de deslocar, como também possuir outras funções (não deslocar e deslocamento bilateral). Isso é feito com a ajuda de multiplexadores, de forma semelhante aos registradores multifunção.

9.2 RTL Design: Elementos do Datapath

Deslocadores

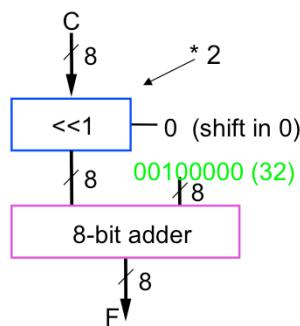
- **Exemplo:** conversor aproximado de Celsius para Fahrenheit
 - $F = C * 9/5 + 32$
 - De forma aproximada: $F = C*2 + 32$

Para ver do que são capazes os deslocadores, vejamos alguns exemplos. Neste primeiro exemplo temos um conversor aproximado de Celsius para Fahrenheit. Aproximando a fórmula temos que $F=C*2 + 32$.

9.2 RTL Design: Elementos do Datapath

Deslocadores

- **Exemplo:** conversor aproximado de Celsius para Fahrenheit
 - $F = C * 9/5 + 32$
 - De forma aproximada: $F = C * 2 + 32$
 - Usando deslocamento à esquerda: $F = (C \ll 1) + 32$



Para fazer a multiplicação por 2, utilizamos um deslocador à esquerda que desloca a entrada 1 bit, atribuindo 0 ao bit menos significativo.

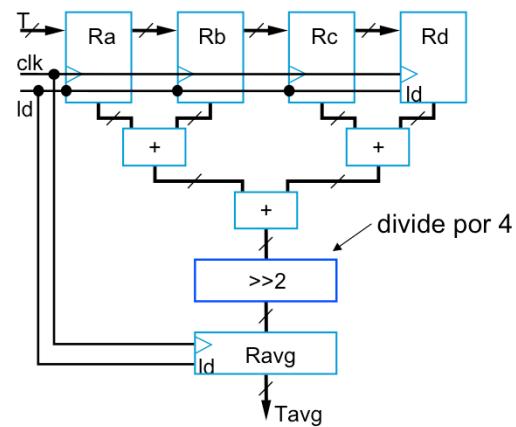
Em seguida, basta adicionarmos um somador de 32 e temos $F = C * 2 + 32$.

9.2 RTL Design: Elementos do Datapath

Deslocadores

- Outro exemplo: média de temperaturas

- Quatro registradores armazenam um histórico de temperaturas
- Deseja-se calcular a média das últimas 4 leituras
- Estratégia: somar e dividir por 4



Neste outro exemplo temos uma calculadora de média para as últimas quatro temperaturas lidas. No topo, quatro registradores guardam as temperaturas, gravando um novo dado toda vez que Id é pressionado. Em seguida, temos uma cascata de somadores somando todos estes valores. Logo depois, um deslocador de 2 bits à direita divide a soma por 4. O registrador final guarda o resultado obtido.

9.2 RTL Design: Elementos do Datapath

Contadores

- Contador ascendente de N bits construído usando registrador:
 - 0000, 0001, 0010, ..., 1111
 - Saída **tc** indica que contagem chegou ao máximo
 - Entrada **cnt** habilita contagem

A seguir veremos os contadores.

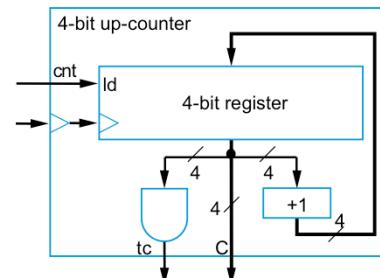
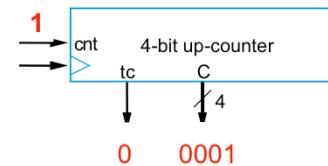
Os contadores possuem muitas variações, mas a forma mais básica é a de um contador ascendente de N bits com uma saída (tc), que indica que a contagem chegou ao máximo, e uma entrada (cnt), que habilita a contagem.

9.2 RTL Design: Elementos do Datapath

Contadores

- Contador ascendente de **N** bits construído usando registrador:
 - 0000, 0001, 0010, ..., 1111
 - Saída **tc** indica que contagem chegou ao máximo
 - Entrada **cnt** habilita contagem

- Incrementador: 

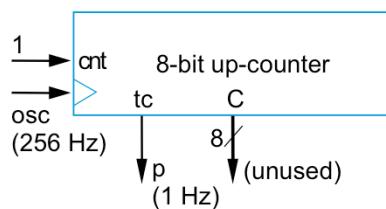


Neste slide temos a visão interna do contador, formado por um registrador, um somador e portas AND.

9.2 RTL Design: Elementos do Datapath

Contadores

- Contador pode ser usado para **gerar pulso de 1 Hz** a partir de um **clock de 256 Hz**
 - Basta usar um contador de 8 bits com indicador de máximo:

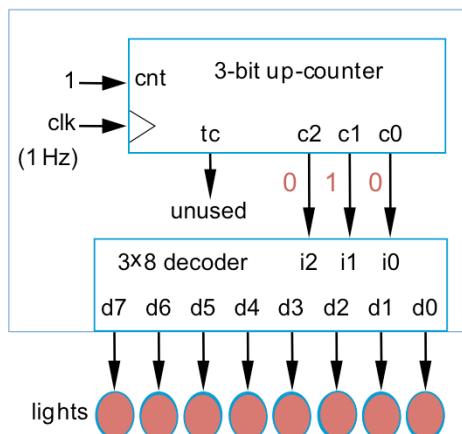


Os contadores são muito úteis como divisores de frequência, ou seja, componentes que transformam uma frequência de clock mais alta em outra mais baixa. Neste slide temos um divisor de frequência que transforma um clock de 256 Hz em outro de 1 Hz. Para construí-lo basta usar um contador de 8 bits (que vai de 0 a 255), onde o novo clock será a saída p, vindia de tc.

9.2 RTL Design: Elementos do Datapath

Contadores

- Sequenciador de lâmpadas:

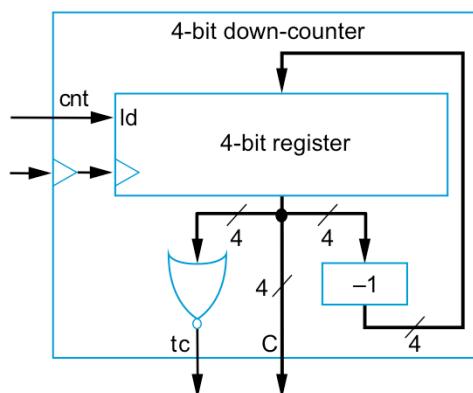


Aqui temos um contador associado a um decodificador (ver aula 5) para fazer um sequenciador de lâmpadas, de modo que a cada ciclo de clock de 1 Hz uma lâmpada é acesa de cada vez.

9.2 RTL Design: Elementos do Datapath

Contadores

- Contador **decrescente (descendente)** de 4 bits:

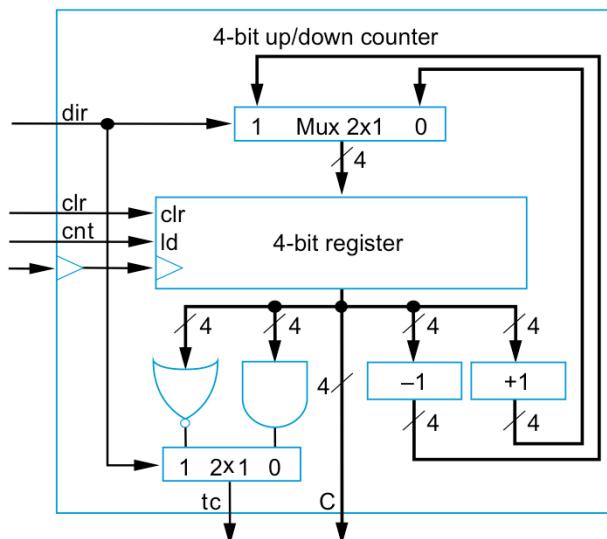


Existem também os contadores descendentes, que contam com um registrador, um subtrator e portas NOR.

9.2 RTL Design: Elementos do Datapath

Contadores

- Contador **ascendente/descendente** de 4 bits:

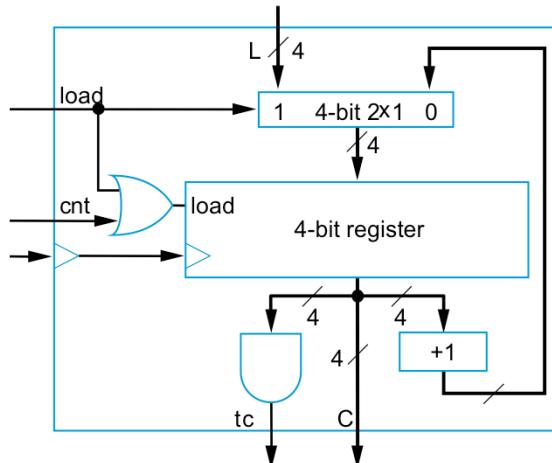


E uma abordagem mista, o contador ascendente/descendente, que se utiliza de multiplexadores 2:1 comandados por uma entrada em comum (dir).

9.2 RTL Design: Elementos do Datapath

Contadores

- **Contador com carga paralela:**
 - É possível gravar valor inicial para contagem:

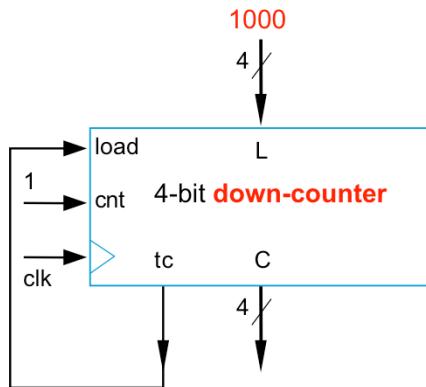


Outra versão é o contador com carga paralela, no qual podemos gravar um valor inicial para a contagem. Note o multiplexador e a porta OR no registrador. A versão descendente deste contador é análoga.

9.2 RTL Design: Elementos do Datapath

Contadores

- **Contador com carga paralela:**
 - Interessante para criar pulsos com frequências múltiplas da frequência de **clock**
 - Gerador de um pulso a cada 9 ciclos de clock:

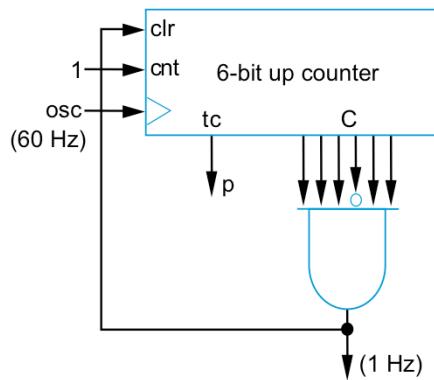


Este tipo de contador é interessante para criar pulsos com frequências múltiplas da frequência de clock original. Veja que toda vez que o limite de contagem é atingido, em "0000", "1000" é carregado paralelamente para o contador, pois tc está ligado a load.

9.2 RTL Design: Elementos do Datapath

Contadores

- Gerador de pulsos de 1 Hz a partir de clock de 60 Hz:



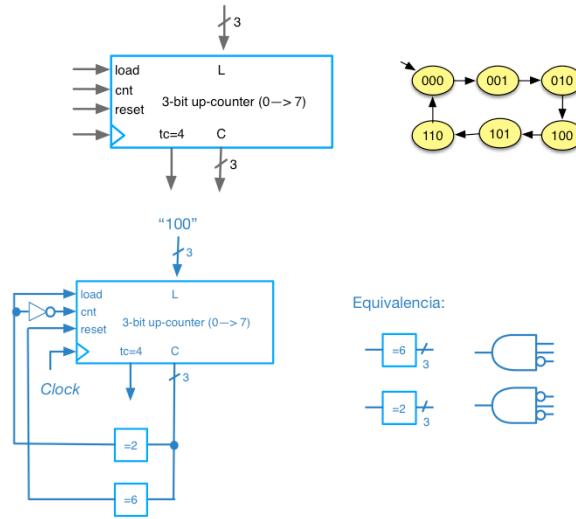
Em caso de o pulso de clock original não ser um múltiplo de dois (como neste exemplo), podemos associar o contador a portas lógicas. Note que o pulso gerado está ligado a clr, recomeçando a contagem.

Tente gerar um pulso de clock de 2 Hz a partir de um clock de 60 Hz na abordagem deste slide. Você verá que o resultado é semelhante ao do slide anterior, mas com um circuito mais simples, apesar de menos genérico.

PROBLEMAS

Problema 9.7. Projete um circuito que fornece a sequência apresentada usando o contador dado e portas lógicas de duas entradas.

Solução:





FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 9

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

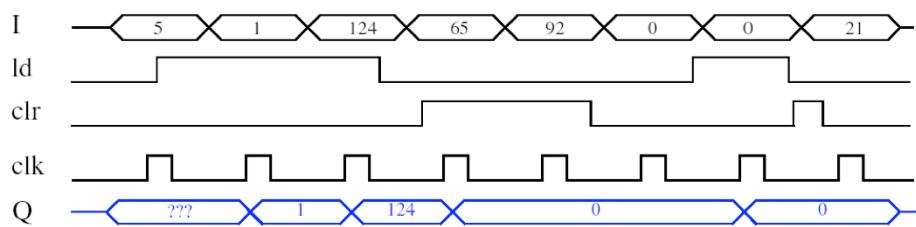
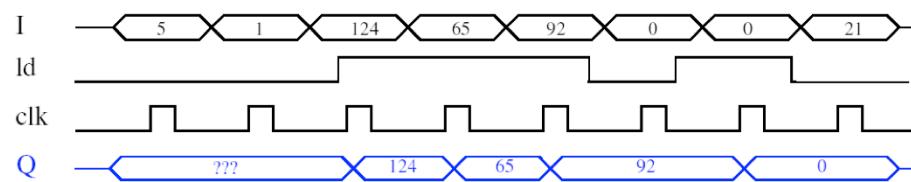
Material desenvolvido com apoio de arquivos de apresentação do livro de Frank Vahid

Exercícios

- Exercícios do Livro do **Frank Vahid**
 - **4.1, 4.2, 4.6, 4.15, 4.17, 4.18, 4.19, 4.28, 4.31, 4.39, 4.40 (ignorar item c), 4.44.**
- **A versão digital do livro do Frank Vahid está disponível no site da BU**
 - Acesse: <http://www.bu.ufsc.br/design/LivrosEletronicos1.html>
 - Clique no link para “Minha Biblioteca”
http://150.162.1.90/pergamon/biblioteca_s/php/login_usu.php?flag=minhabiblioteca_redirect.php
 - Use sua senha da Biblioteca da UFSC para acessar a “Minha Biblioteca”
 - Procure pelo livro do Vahid...

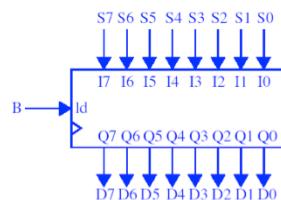
Exercícios

- Resposta do 4.1:



Exercícios

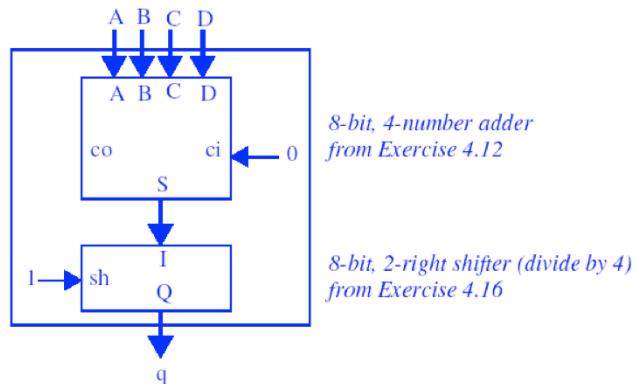
- Resposta do 4.6: **basta usar um registrador com carga paralela**



An 8-bit carry-ripple adder features 8 full-adders. Internally, a full adder's co output is valid after 2 gate delays, and a full adder's s output is valid after 1 gate delay. Thus, each 8-bit carry-ripple adder requires $8 \text{ FAs} * 2 \text{ gate delays/FA} = 16 \text{ gate delays}$. With 3 8-bit carry-ripple adders chained together, the longest delay of adding three 8-bit numbers is $16 * 3 = 48$ time units.

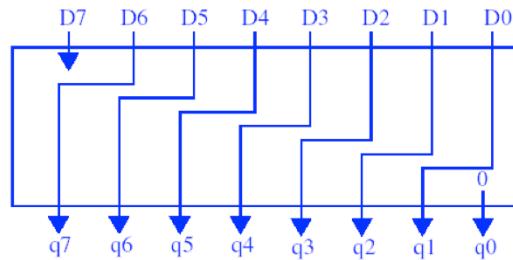
Exercícios

- Resposta do 4.17:



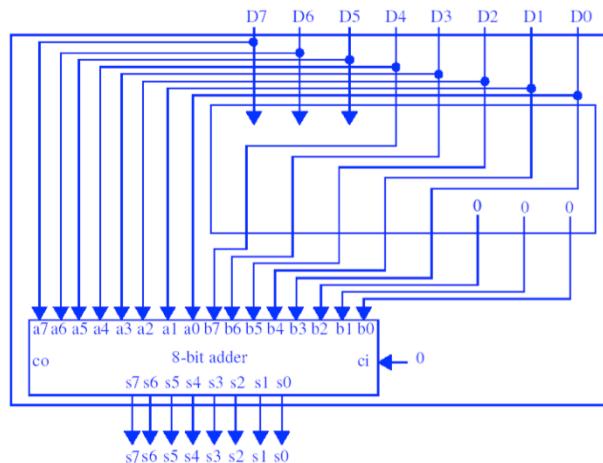
Exercícios

- Resposta do 4.18:



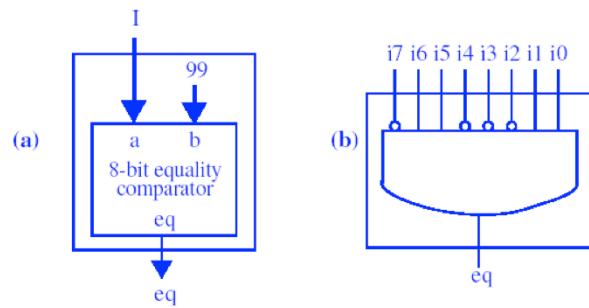
Exercícios

- Resposta do 4.19:



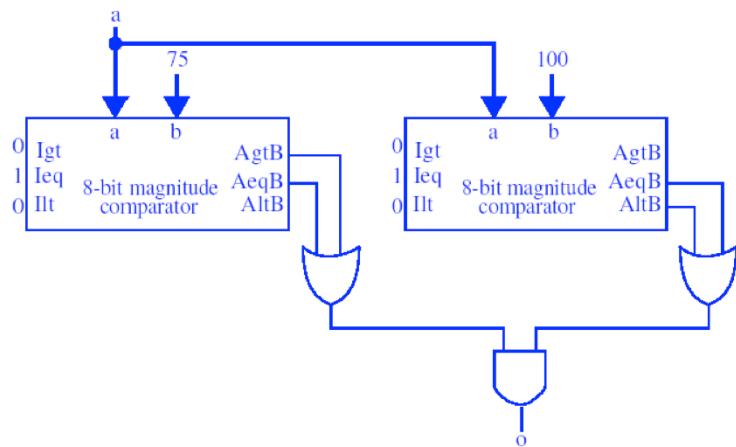
Exercícios

- Resposta do 4.28:



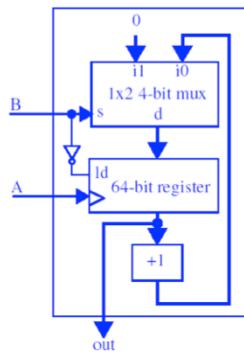
Exercícios

- Resposta do 4.31:



Exercícios

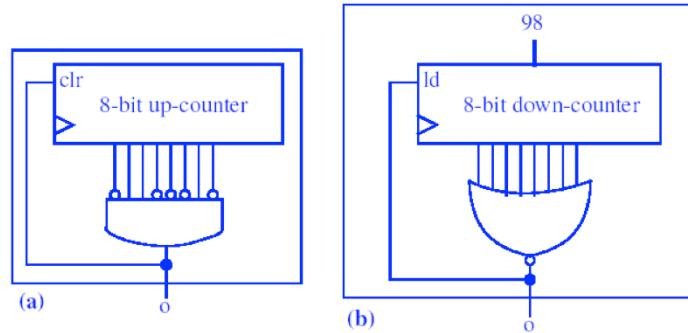
- Resposta do 4.39:



$$2^{64}/15000 = 1,229,782,938,247,303 \text{ days}$$

Exercícios

- Resposta do 4.40:



Exercícios

- Resposta do 4.44: Uma forma de fazer sem o uso de multiplicadores é a seguinte
 - Primeiramente calcular **C*32** (deslocando **C** 5 vezes)
 - Em seguida, calcular **C*2** (deslocando **C** 1 vez)
 - Calcular **C*32-C*2** (com um subtrator), o que resulta em **C*30**
 - Deslocar **C*30** 4 vezes para obter **C*30/16**
 - Finalmente, somar **C*30/16** com 32 para obter **F=C*30/16 + 32**
 - Então, basta usar diversos somadores, subtratores e deslocadores para fazer as operações anteriores
 - É possível fazer de outras formas...