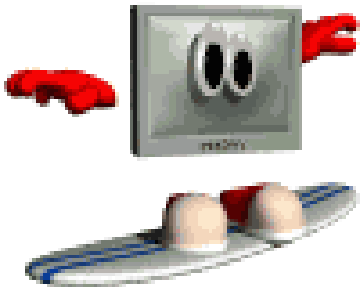


Síntese de Fluxo de Dados de um Circuito Digital

ou Projeto RTL

PCS3722: Organização e Arquitetura de Computadores II
Profa. Dra. Cíntia Borges Margi
cintia@usp.br

Bibliografia

- "Principles of Digital Design", Daniel Gajski, Prentice Hall, 1997.
  - Capítulo 8: Registrer-Transfer Design
- “Embedded System Design: Modeling, Synthesis, Verification”, D. D. Gajski, S. Abdi, A. Gerstlauer, and G. Schirner,, Springer, ISBN 978-1-4419-0503-1, July 2009.
  - Capítulo 6: Hardware Synthesis

Agenda

- Introdução
- Minimização de Registradores
- Minimização de Unidades Funcionais
- Minimização de Vias
- Agrupamento de Registradores
- Revisão

Dados de entrada para a síntese

- A síntese do FD de um módulo digital começa com uma descrição do algoritmo desse módulo.
- O primeiro passo nessa etapa de projeto é transformar a descrição de algoritmo em uma descrição em nível de registradores.

Descrição em nível de registradores

- Fazemos os seguintes mapeamentos:
  - **Variáveis** usadas no algoritmo são transformadas em **registradores físicos** que deverão existir no circuito final;
  - **Operações** definidas pelo algoritmo são transformadas em **unidades funcionais** do **circuito;**
  - **Transferências de dados** são realizadas por **elementos de interconexão** do circuito.

Otimizações na síntese

- 3 tipos de otimizações no projeto do FD:
  - **Minimização de Registradores**: agrupamento de variáveis ou **compartilhamento de registradores**;
  - **Minimização de Unidade Funcionais**: agrupamento de operadores ou com**partilhamento de unidades funcionais**;
  - **Minimização de Interconexões**: agrupamento de interligações ou com**partilhamento de Vias** (“Buses” ou “Barramentos”).

Compartilhamento de Registradores

- Em um algoritmo, em geral, as variáveis não precisam estar disponíveis (“vivas”) o tempo todo, em todos os estados do módulo.
- Variáveis que não estiverem “vivas” ao mesmo tempo podem residir em mesmo registrador.
- A variável está “viva” em determinado estado quando é referenciada (usada) nesse estado.
- Mesmo em caso de variáveis “vivas”, pode ser que elas não sejam utilizadas ao mesmo tempo, e assim podem compartilhar um mesmo banco de registradores.

Compartilhamento de Unidades Funcionais

- **Operadores funcionais** que não são utilizados ao mesmo tempo (ou seja, no mesmo estado), viabilizam a existência de unidades multifuncionais capazes de realizar múltiplas funções em momentos diferentes.
- Por outro lado, a mesma operação pode ser executada mais de uma vez num mesmo estado. Nestes casos, diversas unidades funcionais do mesmo tipo devem existir para que o circuito opere corretamente.
- Num caso pode-se obter redução de custo, e no outro melhoria de desempenho.

Compartilhamento de interconexões

- Atribuições de valores nos algoritmos originam transferências de dados no FD, que implicam na existência de caminhos (ou seja, interconexões).
- Uma mesma interconexão pode suportar diversas transferências de dados.
- Diversas transferências podem ser realizadas num mesmo estado.
- Transferências que não são executadas ao mesmo tempo, e que especificam o mesmo destino, podem compartilhar a mesma interconexão.
- Compartilhar interconexões origina utilização de vias multiponto (*Buses*) → reduz número de conexões.

Minimização de Registradores

Vamos ilustrar o processo de síntese por meio de um exemplo!

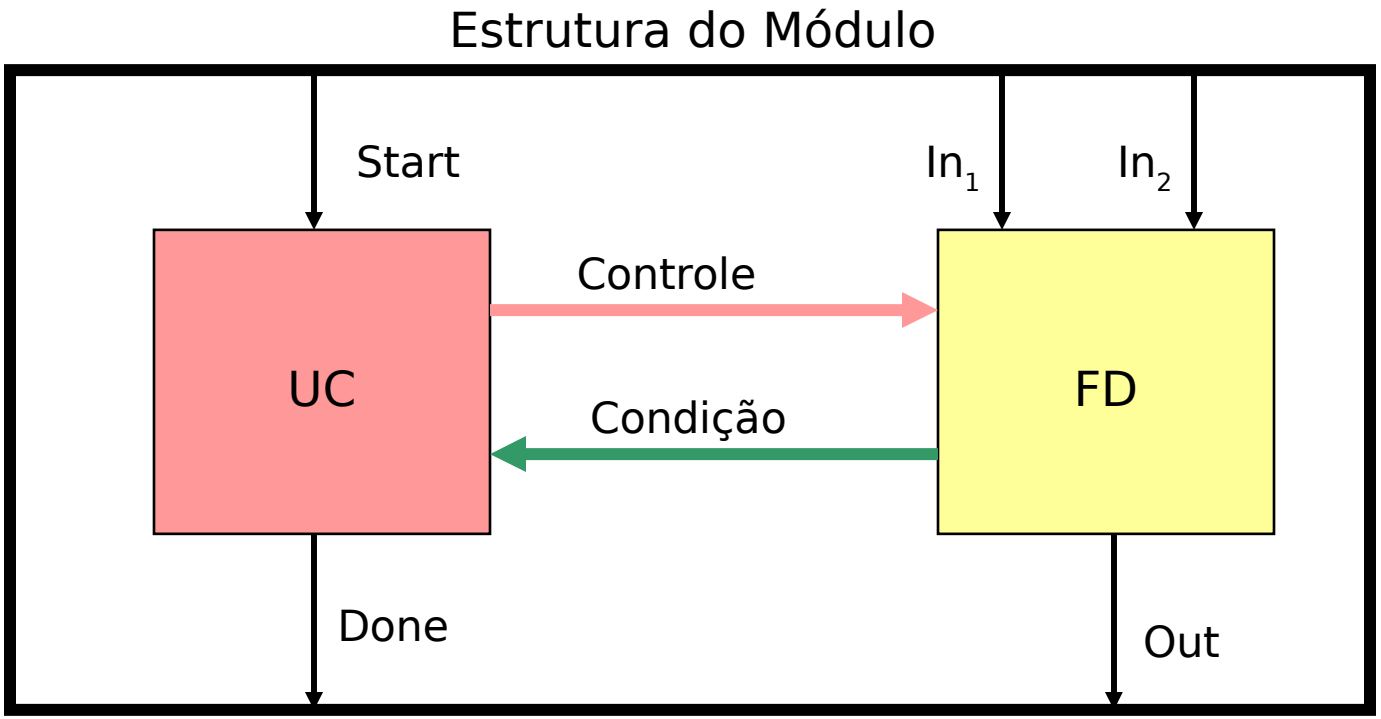
Exemplo de Projeto

- Módulo que calcula a raiz quadrada (valor aproximado) da soma do quadrado de dois números inteiros (hipotenusa de um triângulo retângulo):

$$\sqrt{a^2 + b^2} \approx \max((0.875x + 0.5y), x)$$

onde x = max(|a|,|b|) e y = min(|a|,|b|), e |a| e |b| representam o valor absoluto de a e b respectivamente.

Raiz quadrada da soma de inteiros

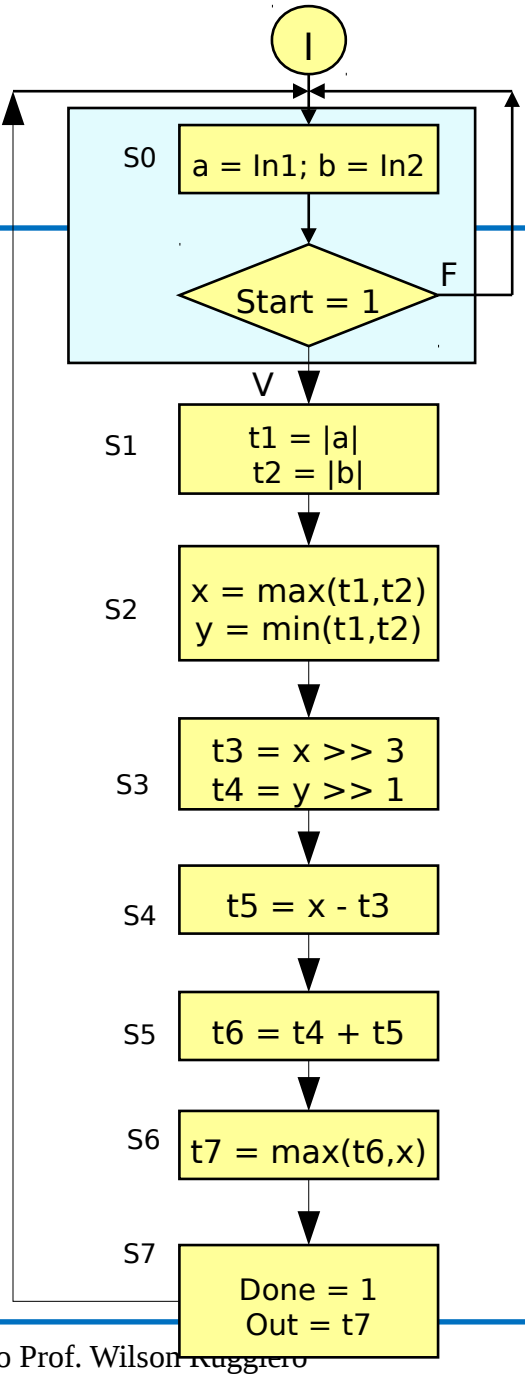


Sinais de Entrada: In_1, In_2 e Start

Sinais de Saída:Out, Done

Diagrama ASM do Módulo

- Possui 8 estados.
- Testa a única entrada (Start).
- Recebe os valores de entrada nos sinais In1 e In2.
- Aciona um sinal de controle de saída (Done) e posiciona o resultado na saída Out.



Utilização de recursos

- Para se determinar os recursos que são necessários à implementação deste diagrama ASM deve-se construir duas tabelas:
  - Tabela de uso de variáveis: utilizada no processo de minimização do número de registradores necessários;
  - Tabela de uso de operadores: utilizada no processo de composição de unidades multifuncionais.

Tabela de uso de Variáveis

- Cada linha representa uma variável e cada *coluna* representa um *estado* do diagrama ASM.
- Para cada variável deve-se marcar os estado em que ela esta “viva”.
  - Uma variável é considerada “viva” desde o estado seguinte àquele em que ela recebeu um novo valor até o último estado em que ela é utilizada.
- O número máximo de variáveis “vivas” num único estado define o número mínimo de registradores necessários para a síntese do diagrama ASM do circuito.

Tabela de uso de Variáveis para exemplo

- Observando o diagrama ASM, vamos construir a tabela de uso de variáveis.

  1) Identifique os estados em que são realizadas atribuições às variáveis.

  2) Identifique os estados em que as variáveis são referenciadas, ou seja, estão vivas.

  3) O número máximo de variáveis vivas em um estado determina o número mínimo de registradores no Fluxo de Dados.

Atribuições de Variáveis

Variável / Estado	S0	S1	S2	S3	S4	S5	S6	S7
a	x							
b	x							
t1		x						
t2		x						
x			x					
y			x					
t3				x				
t4				x				
t5					x			
t6						x		
t7							x	

Tabela de Uso de Variáveis

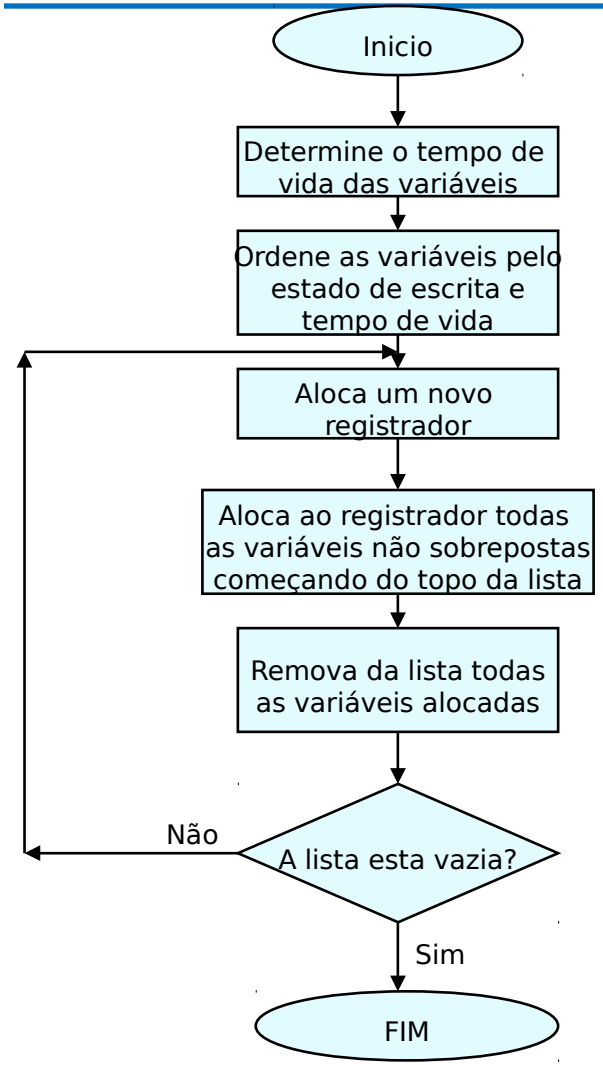
Variável / Estado	S0	S1	S2	S3	S4	S5	S6	S7
a		x						
b		x						
t1			x					
t2			x					
x				x	x	x	x	
y				x				
t3					x			
t4					x	x		
t5						x		
t6							x	
t7								x
Número de Variáveis Vivas	0	2	2	2	3	3	2	1

Variável está viva no estado em que é referenciada!

Mapeamento: Variáveis <> Registradores

- Baseado na Tabela de uso de variáveis, conclui-se que as variáveis devem ser agrupadas em 3 conjuntos onde elas não estejam vivas ao mesmo tempo.
- Mas, como determinar o mapeamento ótimo?
  - Left-Edge Algorithm
  - Graph-Partitioning Algorithm

Left-Edge Algorithm

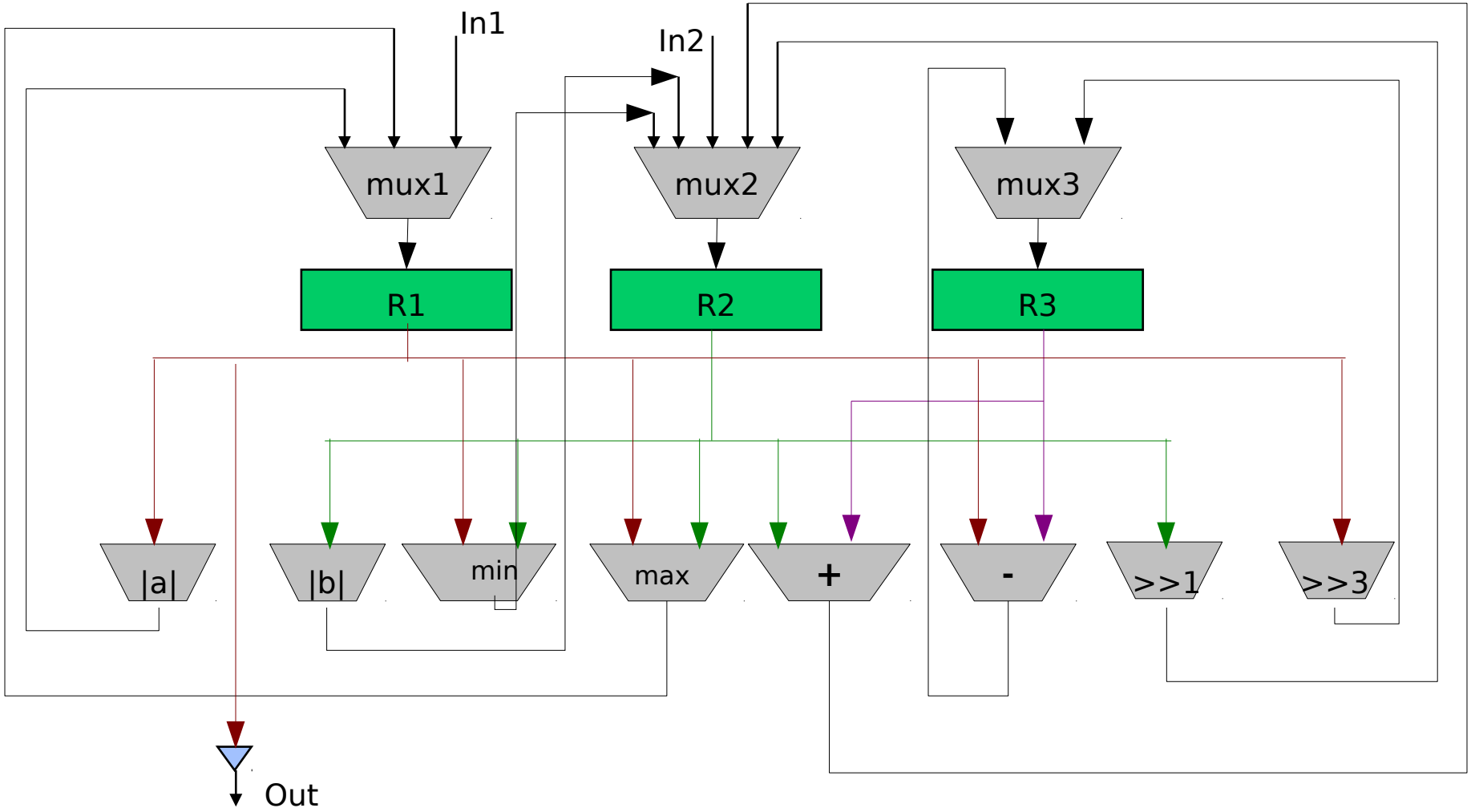


Tempo de Vida das Variáveis			
Variável	**Tempo de vida**	**Conflitos**	**Registrador**
x	**4**	**y,t3,t4,t5,t6**	**R1**
t4	**2**	**x,t3,t5**	**R2**
a	**1**	**b**	**R1**
b	**1**	**a**	**R2**
t1	**1**	**t2**	**R1**
t2	**1**	**t1**	**R2**
y	**1**	**x**	**R2**
t3	**1**	**x,t4**	**R3**
t5	**1**	**x,t4**	**R3**
t6	**1**	**x**	**R2**
t7	**1**	**--**	**R1**

Alocação de registradores

- A alocação resultante do algoritmo Left-Edge é a seguinte:
  - R1 = {a, t1, x, t7}
  - R2 = {b, t2, y, t4, t6}
  - R3 = {t3, t5}
- O diagrama em síntese necessita de três registradores e a alocação das variáveis do diagrama a esses registradores está definida nos três conjuntos acima.

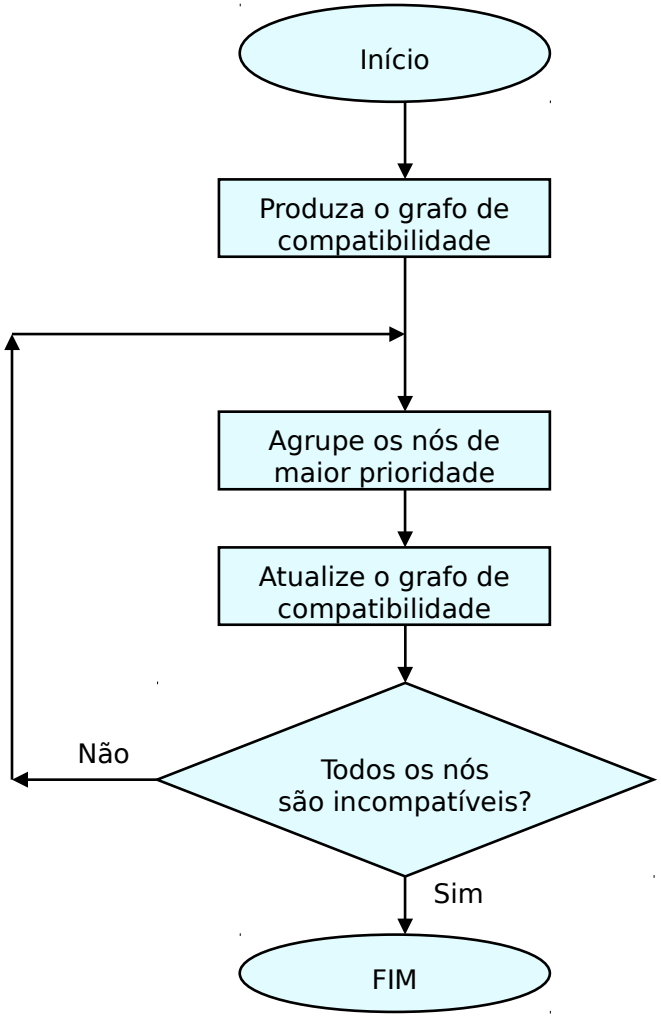
FD após minimização com Left-Edge



Mais minimização

- No processo de agrupar registradores dependendo de quais são agrupados pode-se minimizar também as entradas de Multiplexadores.
- Duas variáveis devem ser preferencialmente alocadas a um mesmo registrador se elas não possuem os seus tempos de vida sobrepostos e servem como origem ou destino para uma mesma unidade funcional.
- Esta observação dá origem a um novo algoritmo de alocação de variáveis a registradores: “Graph-Partitioning-Algorithm”.

Graph-Partitioning Algorithm



Este algoritmo necessita de um grafo auxiliar chamado de **grafo de compatibilidade**.

Grafo de compatibilidade

- Consiste de nós representando variáveis e ramos entre nós, representando uma relação de compatibilidade ou incompatibilidade entre as variáveis.
- Ramo de incompatibilidade (linha pontilhada) relaciona variáveis com tempo de vida sobrepostos.
- Um ramo de prioridade ou de compatibilidade (linha cheia) indica variáveis sem sobreposição de tempo de vida e que ambas servem como operando de entrada para uma mesma unidade funcional ou ambas servem como operando de saída para diferentes unidades funcionais.

Grafo de Compatibilidade

- Associado aos ramos de prioridade existe um peso que representa o número de entradas de multiplexadores que podem ser economizadas se essas variáveis forem alocadas a um mesmo registrador.
- O peso de prioridade possui o formato S/D onde:
  - S: representa o número de unidades funcionais diferentes que utilizam ambas variáveis como operando esquerdo ou direto (entrada);
  - D: representa o número de unidades funcionais diferentes que produzem resultados para ambas variáveis como operando destino (resultado).
- É construído a partir da Tabela de Conectividade.

Tabela de Conectividade

- As linhas desta tabela são as unidades funcionais identificadas no diagrama ASM.
  - Assumiremos que as operações de min e max compartilham uma unidade funcional, e que as unidades + e – também compartilham uma unidade funcional.
- As colunas são as variáveis existentes no diagrama ASM.
- Os elementos da tabela indicam se a respectiva variável é entrada ou saída da correspondente unidade funcional.

Tabela de Conectividade para exemplo

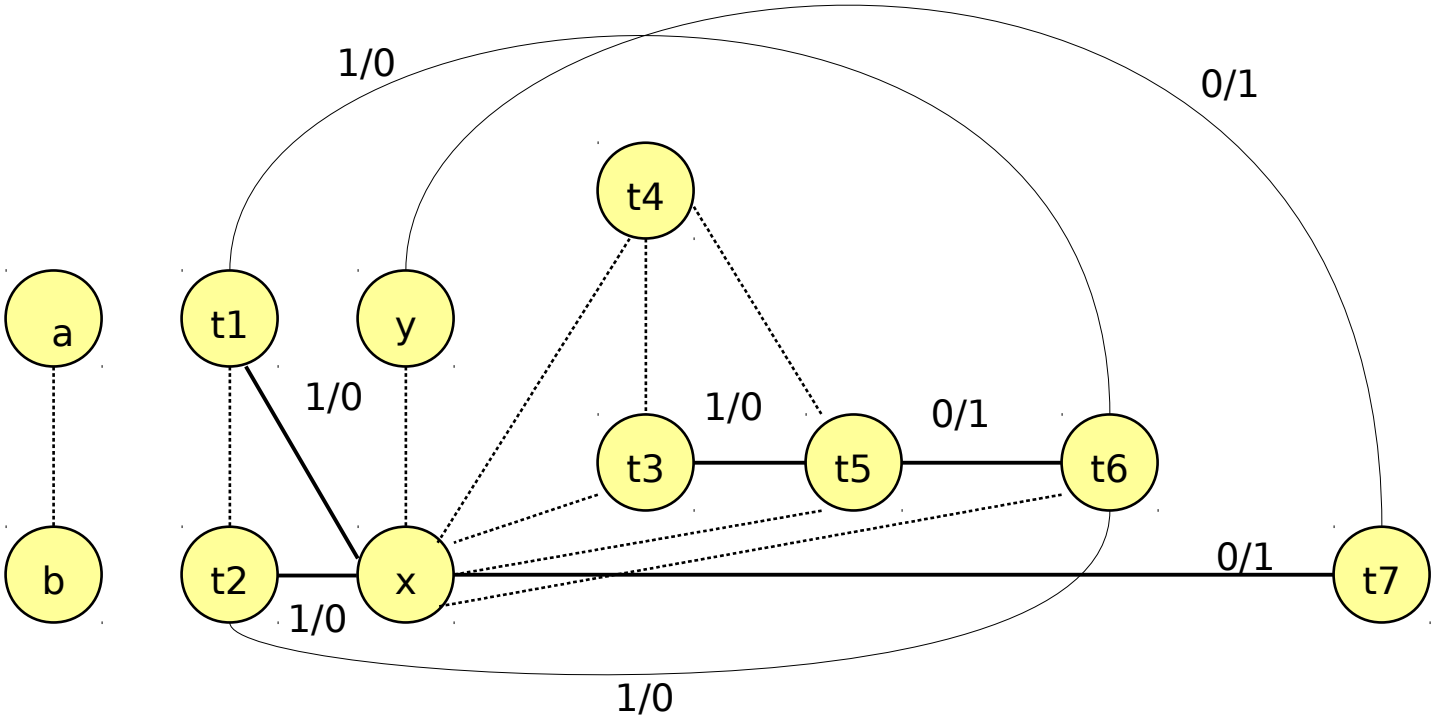
	a	b	t1	t2	x	y	t3	t4	t5	t6	t7
abs1	**E**		**S**								
abs2		**E**		**S**							
min, max			**E**	**E**	**S, E**	**S**				**E**	**S**
>>3					**E**		**S**				
>>1						**E**		**S**			
-, +					**E**		**E**	**E**	**S, E**	**S**	

Min. e max. numa mesma unidade funcional. + e – também numa mesma unidade funcional.

Arcos de Compatibilidade

- Entradas comuns:
  - t1 – x : ambas são entradas de max (1/0);
  - t1 – t6: idem acima (1/0);
  - t2 – x: idem acima (1/0);
  - t2 – t6: idem acima (1/0);
  - t3 – t5: ambas são entradas de +/- (1/0).
- Saídas Comuns:
  - x – t7: ambas são saídas de max (0/1);
  - y– t7: idem acima (0/1);
  - t5 – t6: ambas são saídas de +/- (0/1).

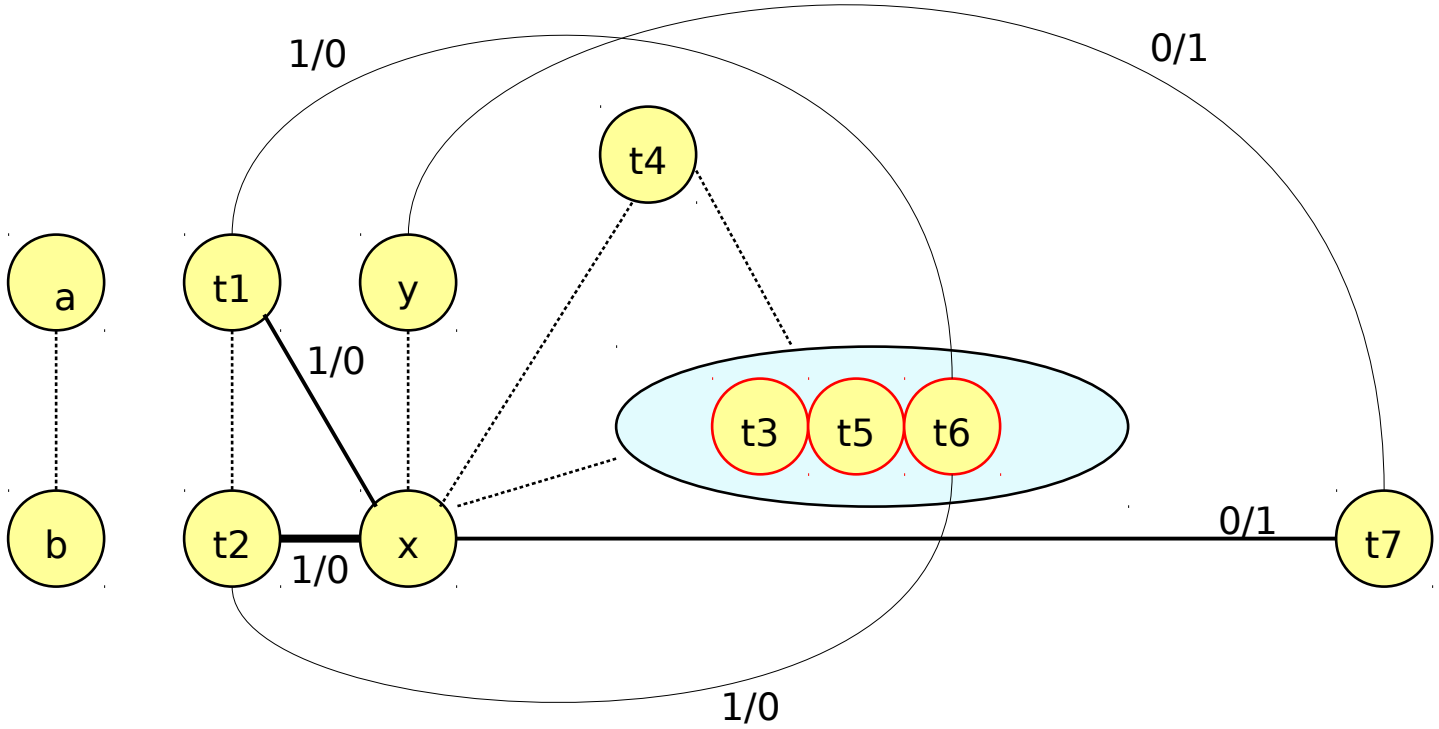
Grafo de Compatibilidade inicial (0)



Este grafo considera o uso de unidades multifuncionais: min/max e +/-.

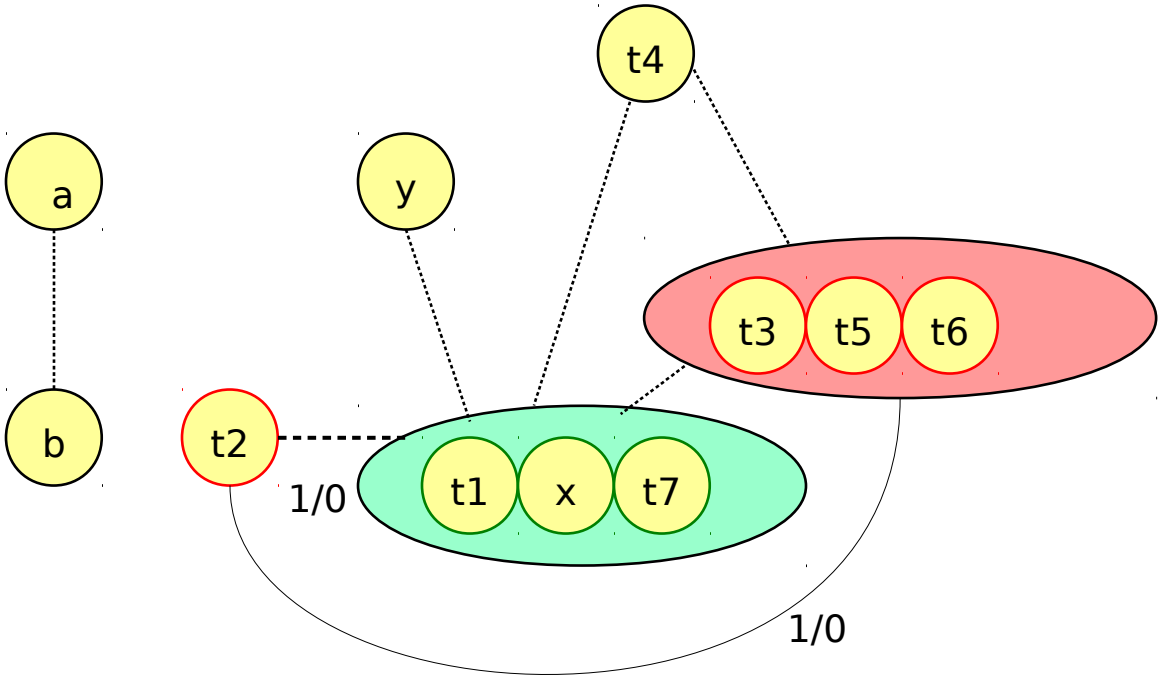
Grafo de Compatibilidade (1)

Agrupando os ramos entre t3,t5 e t6, resulta em um novo grafo.



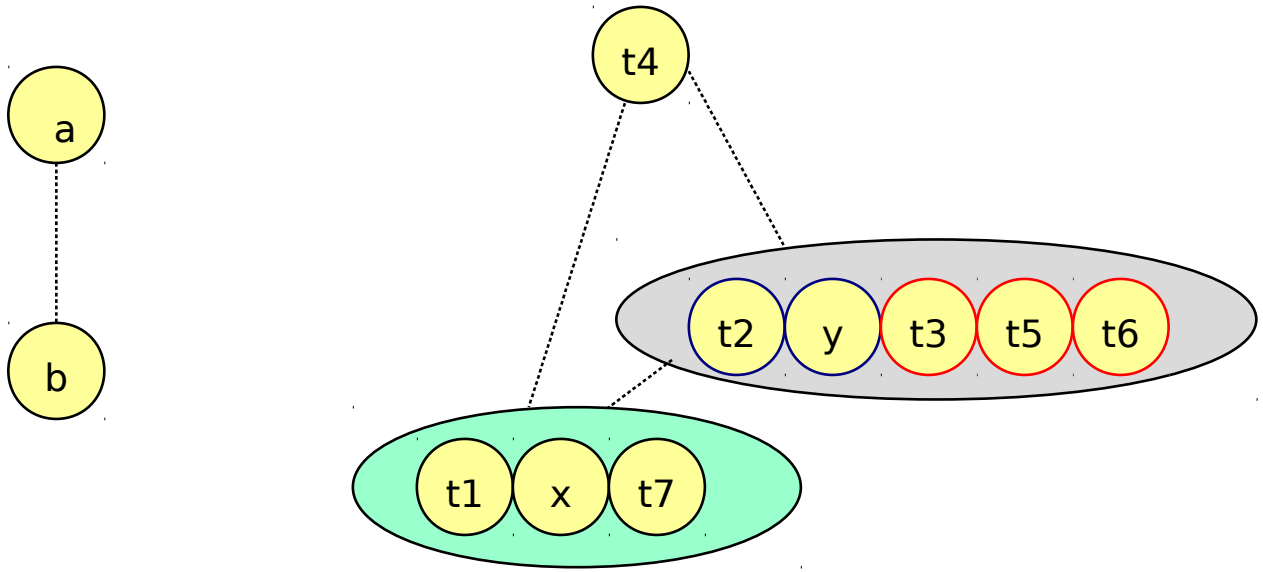
Grafo de Compatibilidade (2)

Agrupando os ramos: t1, x e t7, temos um um novo grafo.



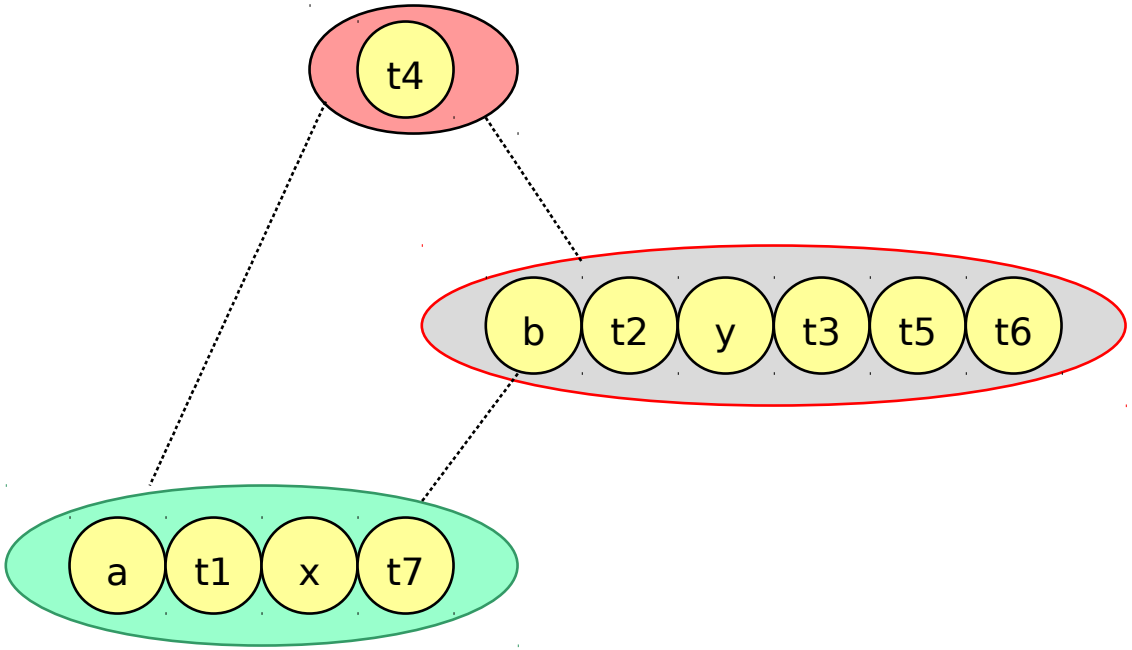
Grafo de Compatibilidade (3)

Agrupando os nós t2 e y com, o grupo formado por t3, t5 e t6, temos novo grafo de compatibilidade.



Grafo de Compatibilidade Final

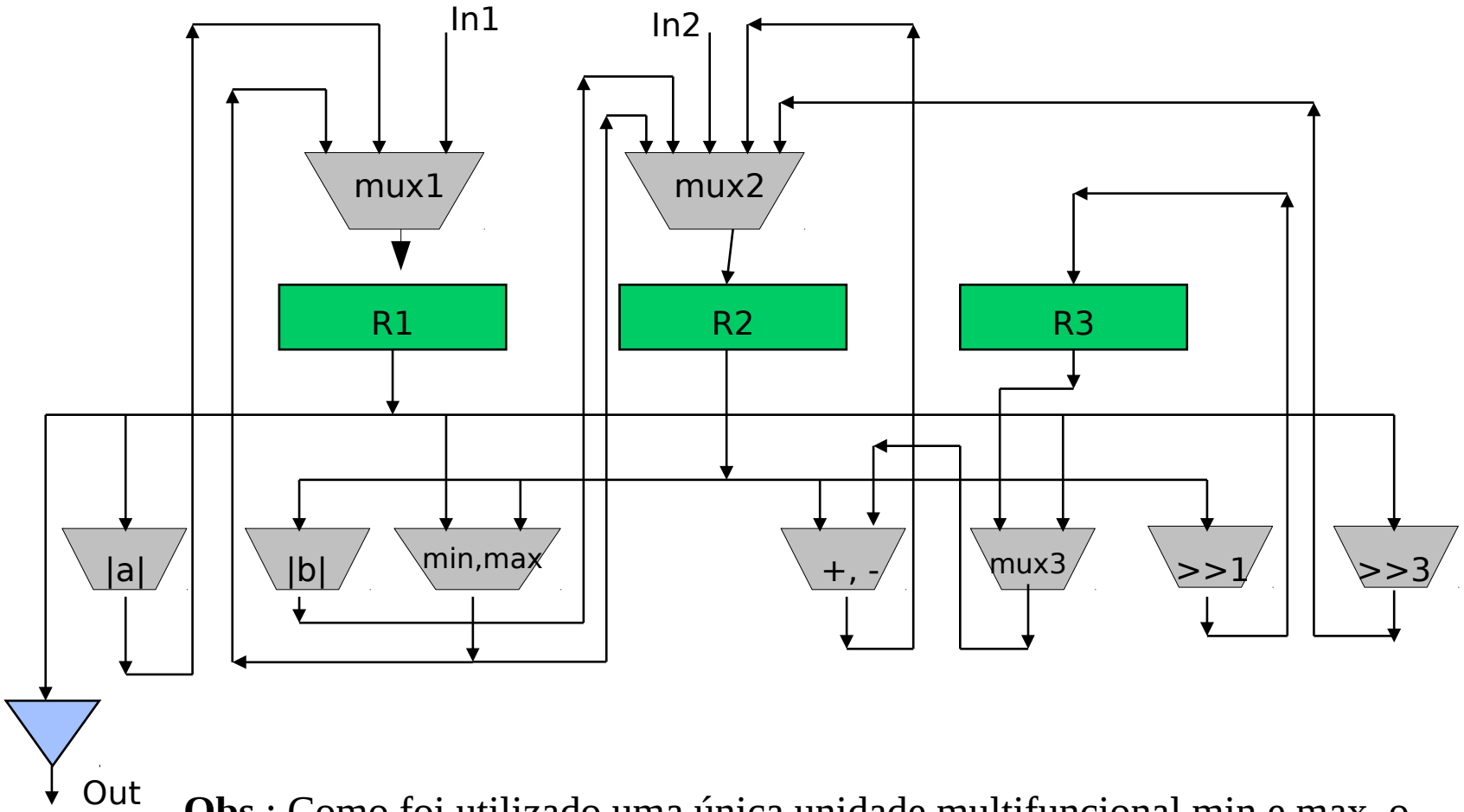
Finalmente juntando a e b com grupos sem incompatibilidade, chega-se ao grafo final de alocação.



Então, atribuímos cada grupo a um registrador, obtendo a seguinte alocação:

R1 = {a, t1, x, t7}
R2 = {b, t2, y, t3, t5, t6}
R3 = {t4}

FD: minimização com Graph-Partitioning



Obs.: Como foi utilizado uma única unidade multifuncional min e max, o diagrama ASM deve possuir mais estados que o diagrama inicial da síntese, ou seja, o estado s2 deve ser desmembrado em dois.

Minimização de unidades funcionais

Minimização de unidades funcionais

- Similarmente, se duas ou mais funções não são exigidas simultaneamente, elas podem ser combinadas em uma mesma unidade multifuncional.
- Este agrupamento, em alguns casos, pode não reduzir o custo do circuito, porém pode ter uma implicação importante na redução do número de interconexões.
- Para fazer este tipo de minimização utiliza-se também um grafo de compatibilidade.
- Considere a tabela de operações apresentada pelo diagrama ASM.

Tabela de uso de Operadores

Op. / Estado	S1	S2	S3	S4	S5	S6	S7	No. max. UF
abs	2							2
min		1						1
max		1				1		2
>>			2					2
-				1				1
+					1			1
No. operações	2	2	2	1	1	1	0	

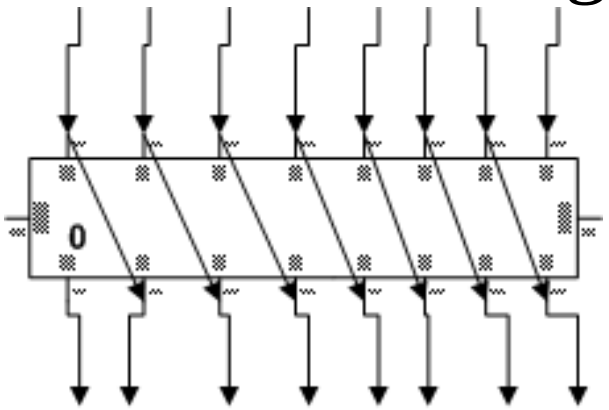
Obs.: Nesta tabela voltamos a considerar as unidade de min e max, + e - separadamente.

Operações simultâneas

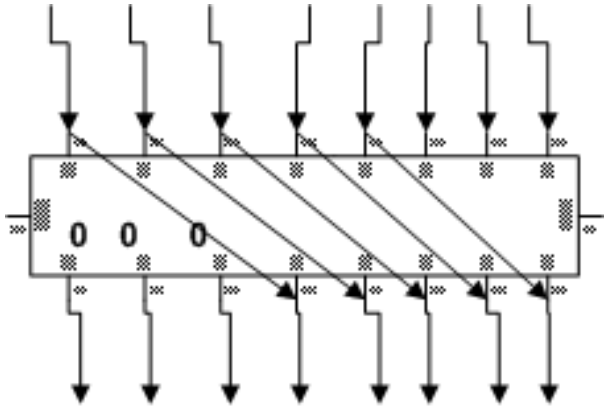
- Observando esta tabela pode-se notar quais são as operações que devem ser realizadas simultaneamente num mesmo estado.
  - abs
  - >>
- Estas operações são consideradas incompatíveis e não devem ser agrupadas numa mesma unidade multifuncional.

Operação de Deslocamento

- É a unidade funcional mais simples, e pode ser implementada através da ligação apropriadas dos “fios” de entrada e de saída.
- Assim, possui custo zero! Por isso não será considerada no grafo de compatibilidade.



Deslocador de 1 bit à esquerda

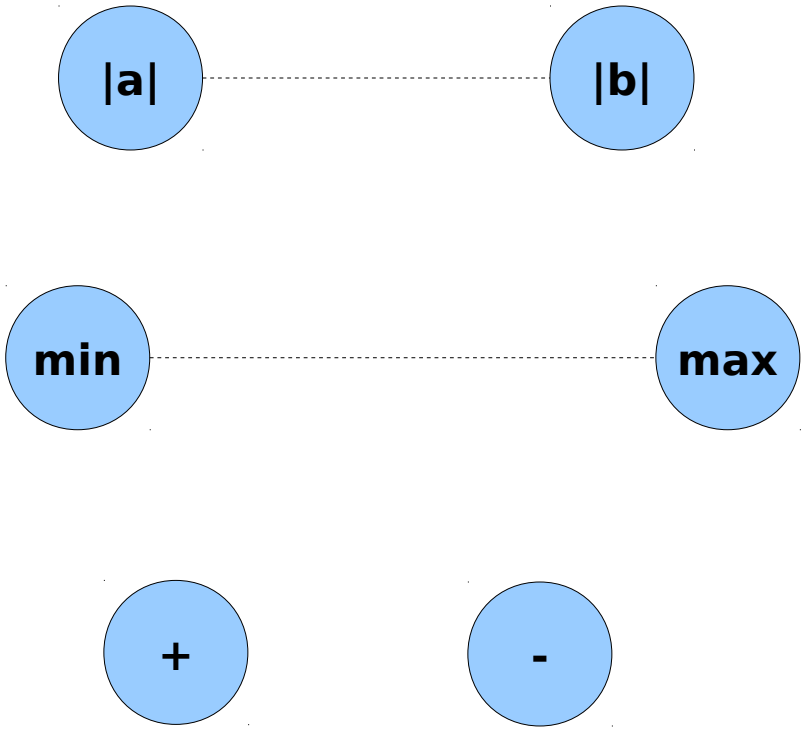


Deslocador de 3 bits à esquerda

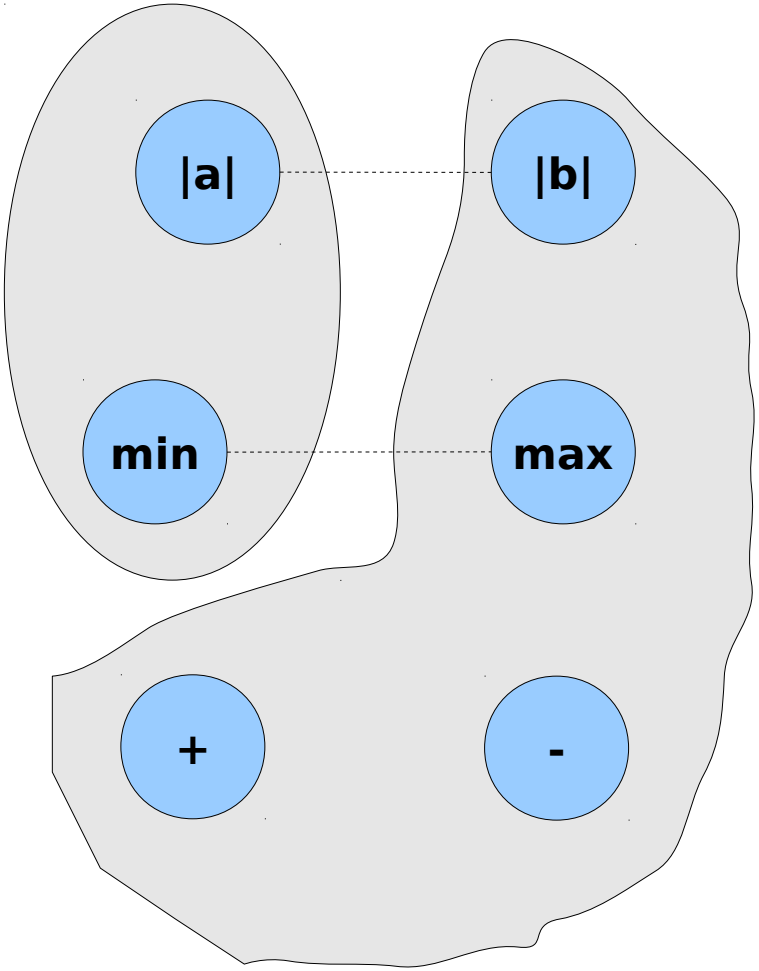
Grafo de Compatibilidade

- Similar ao grafo utilizado para minimização de registradores.
- Associado aos ramos de prioridade existe um peso (S/D), representando o número de entradas de multiplexadores que podem ser economizadas se essas variáveis forem alocadas a um mesmo registrador.

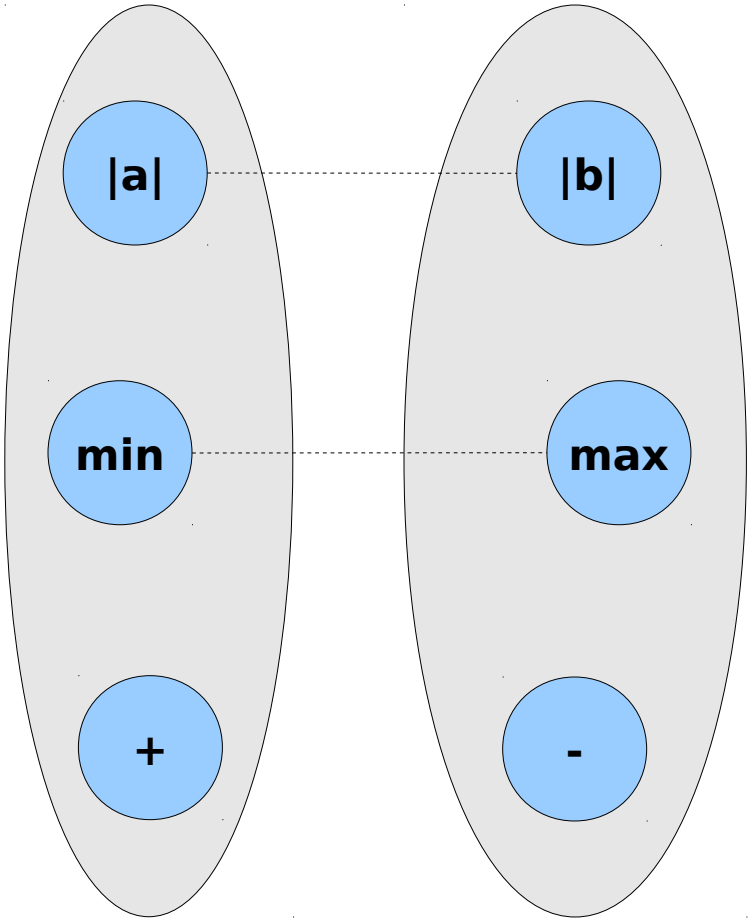
Grafo de Compatibilidade Inicial



Grafo de Compatibilidade - possibilidades



Alternativa 1

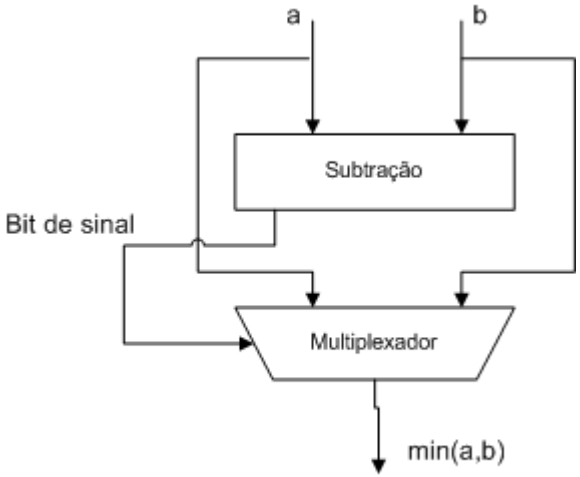


Alternativa 2

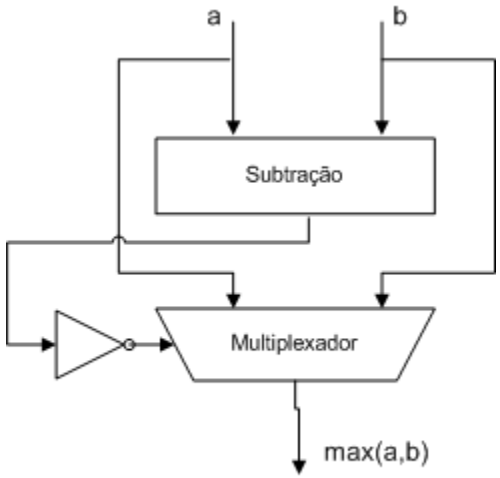
Grafo de Compatibilidade

- Como escolher o grafo de compatibilidade que será implementado?
  - Escolhe-se a alternativa de menor custo!
- É preciso sintetizar os circuitos combinatórios das unidades multifuncionais usadas em cada alternativa.
  - Neste exemplo, ambas as alternativas possuem mesmo custo.
  - Alguns exemplos de módulos funcionais a seguir.

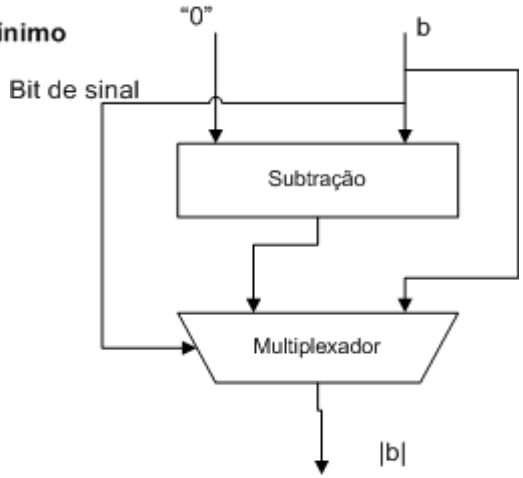
Módulos para unidades funcionais



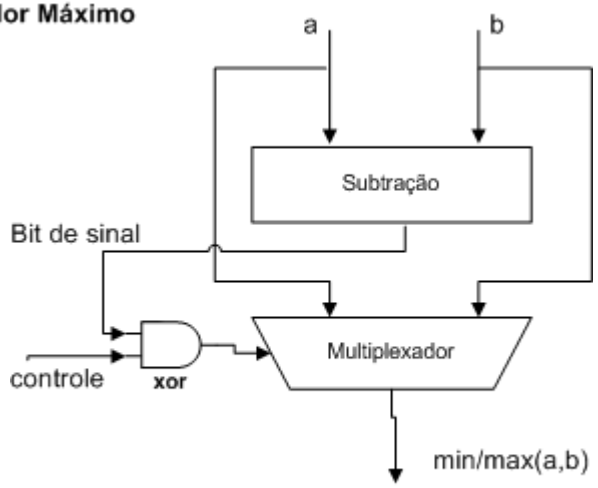
Unidade de Valor Mínimo



Unidade de Valor Máximo

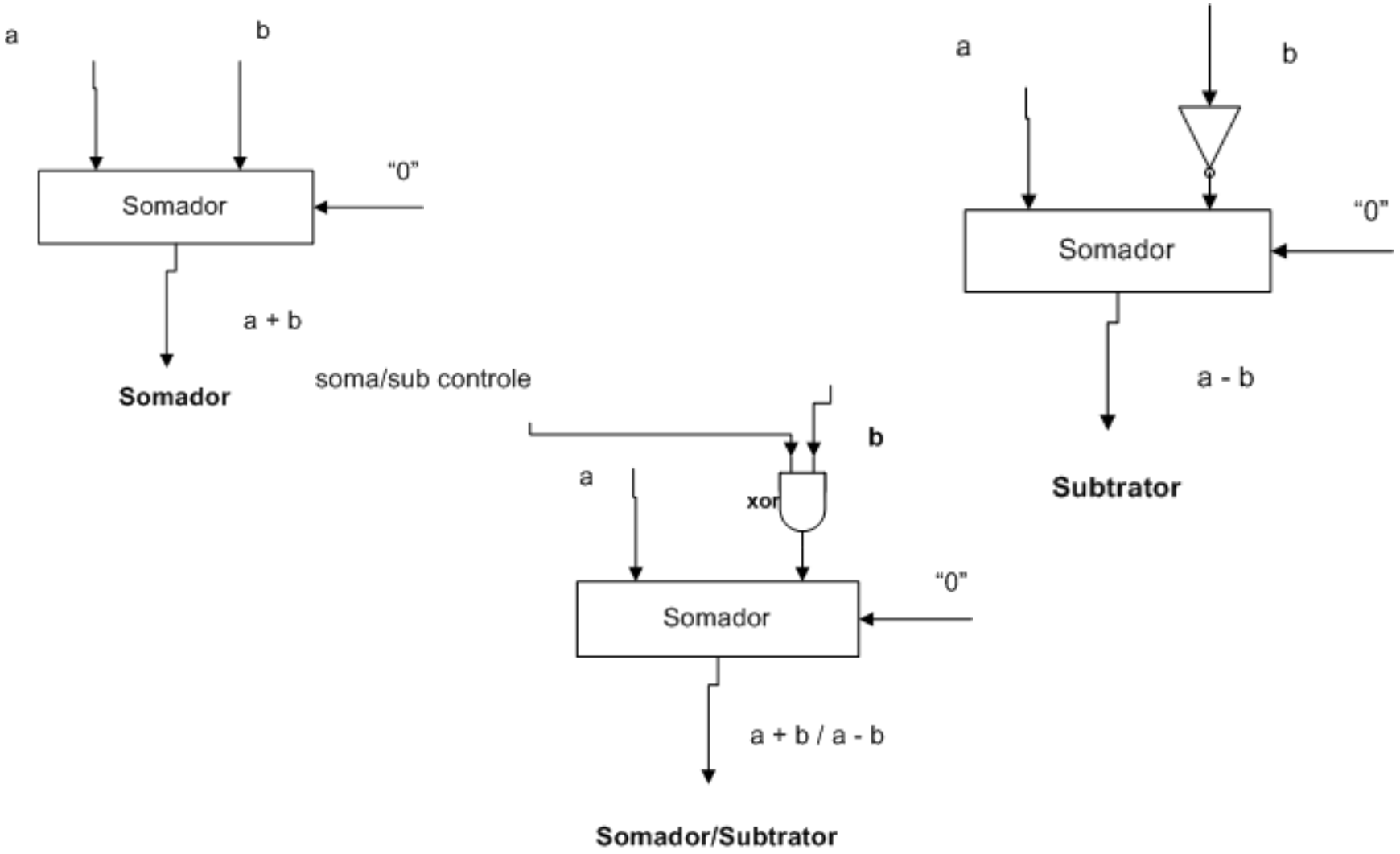


Unidade de Valor Absoluto

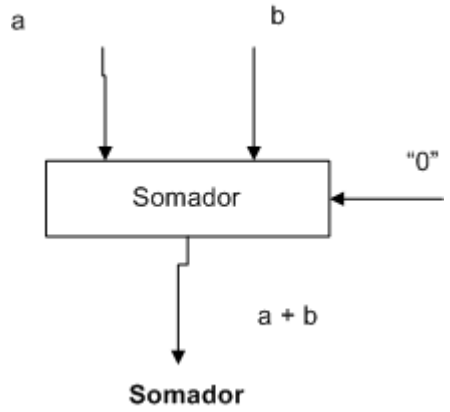


Unidade de valor Mínimo ou máximo

Mais módulos...

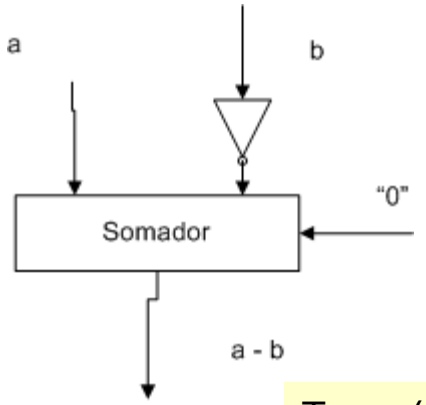


Unidades funcionais: +, -, >>1 e >>3



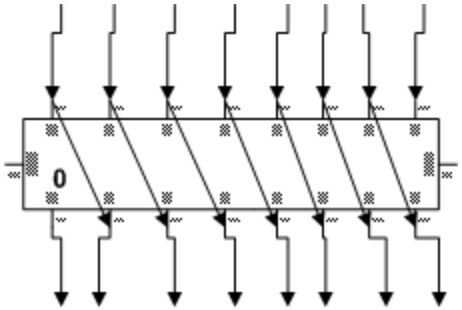
Somador

Tprop(+) = 3 ns



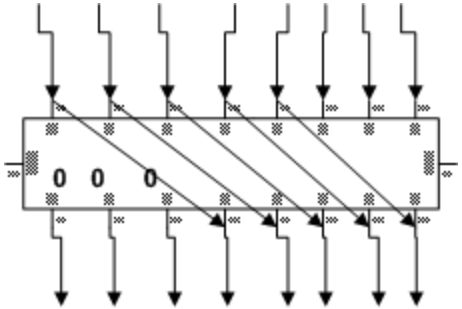
Subtrator

Tprop(-) = Tprop(not) + Tprop(+)
= 1 ns + 3 ns = 4 ns



Deslocador de 1 bit à esquerda

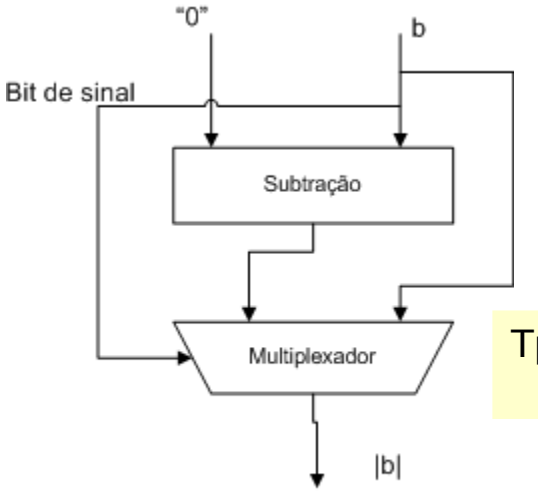
Tprop(>>1) = 0 ns



Deslocador de 3 bits à esquerda

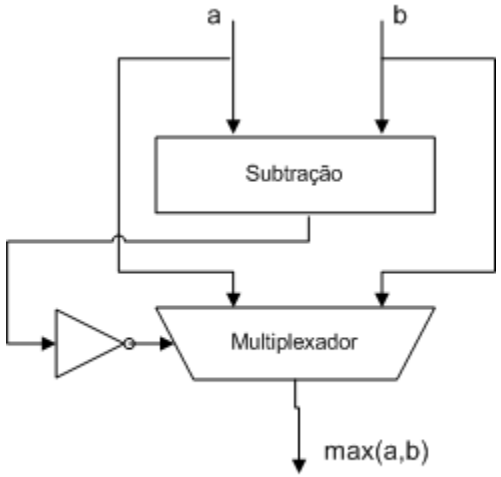
Tprop(>>3) = 0 ns

Unidades funcionais:||, min, e max

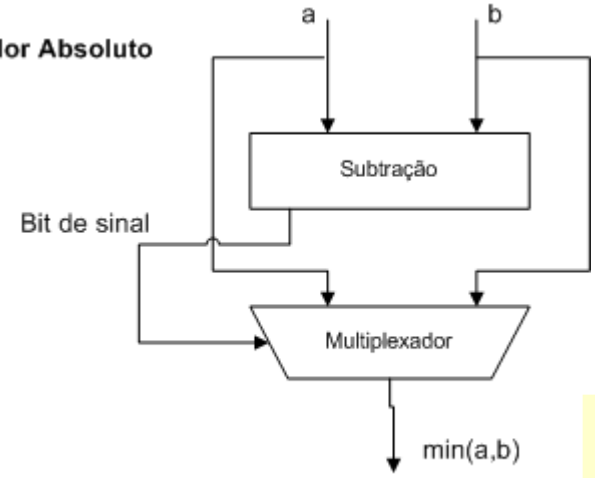


Unidade de Valor Absoluto

Tprop(||) = Tprop(-) + Tsel(mux)
= 4 ns + 3 ns = 6 ns



Unidade de Valor Máximo

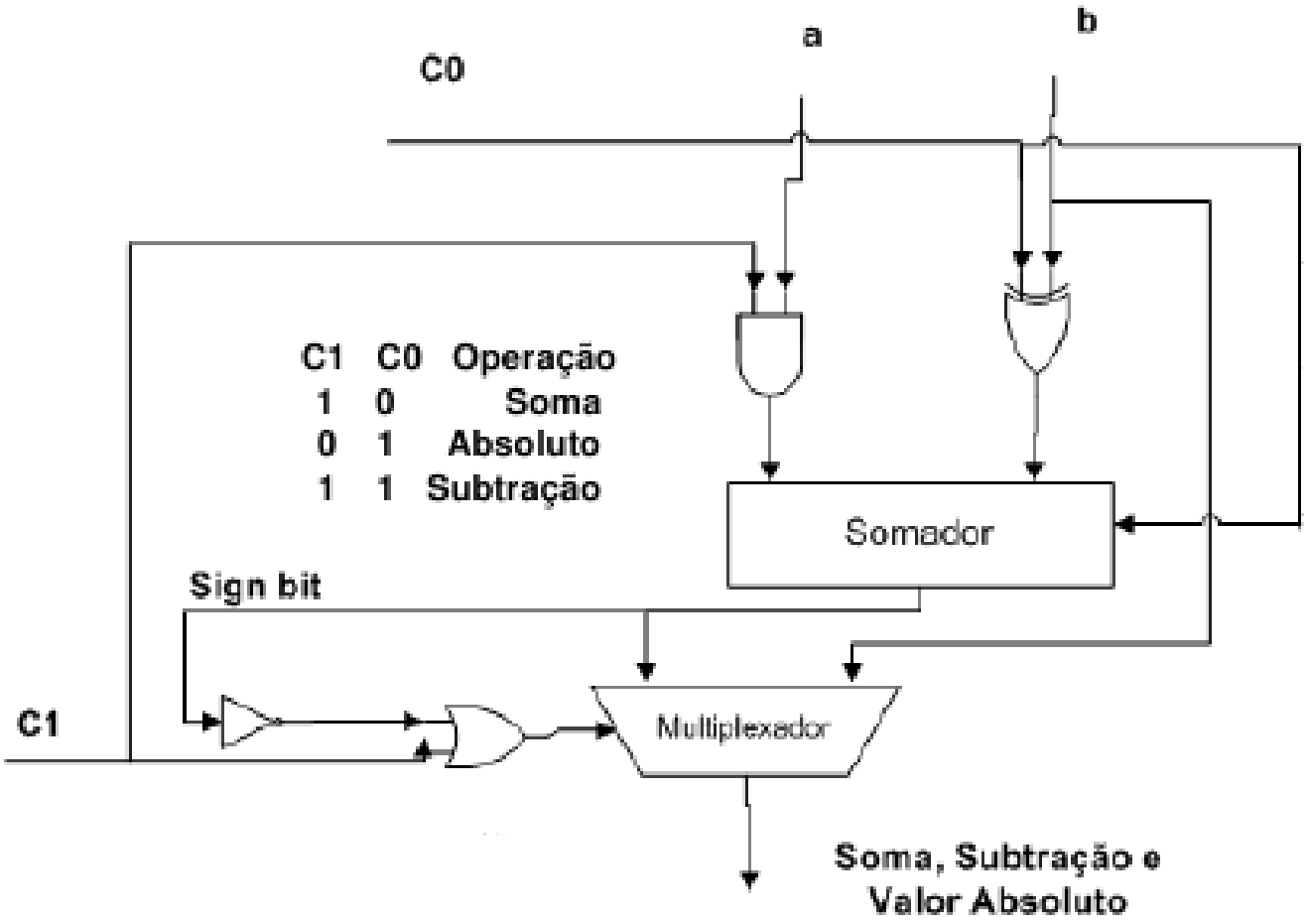


Unidade de Valor Mínimo

Tprop(max) = Tprop(-) + Tprop(not)
+ Tsel(mux)
= 4 ns + 1 ns + 3 ns
= 8 ns

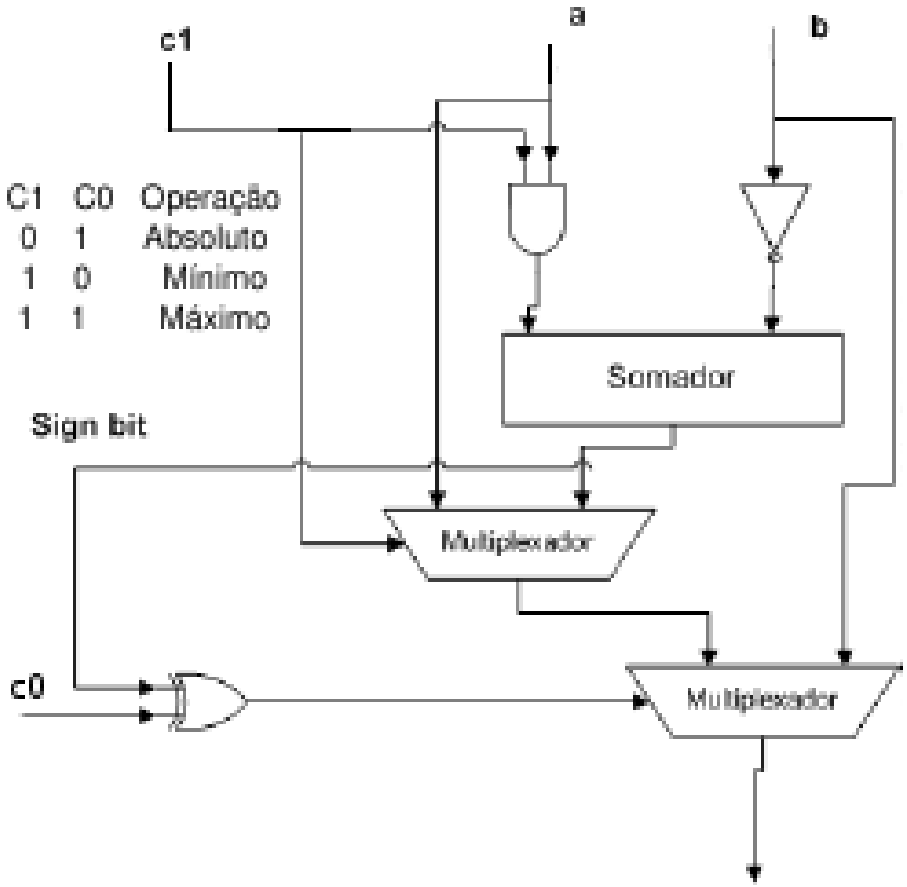
Tprop(min) = Tprop(-) + Tsel(mux)
= 4ns + 3ns = 7ns

Unidade Multi Funcional: +, - e |b|



Tprop = Tprop(and e xor) + Tsoma + Tprop (not e or) + Tsel(mux)
= 2ns + 3ns + 2ns + 3ns = 10 ns

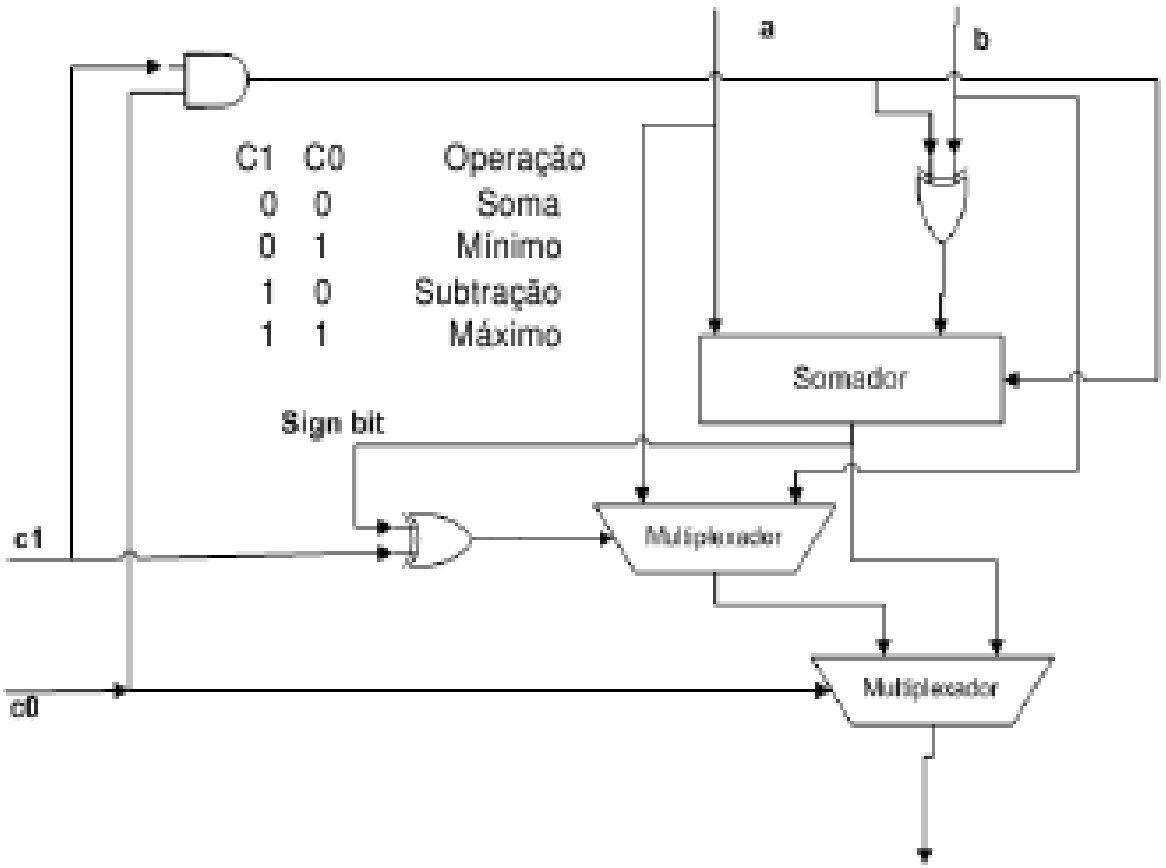
Unidade Multi Funcional: min, max e |b|



UF = {Min, Max e |b| }

Tprop = Tprop(and e not) + Tsoma + Tsel(mux)
= 2ns + 3ns + 2ns = 7n s

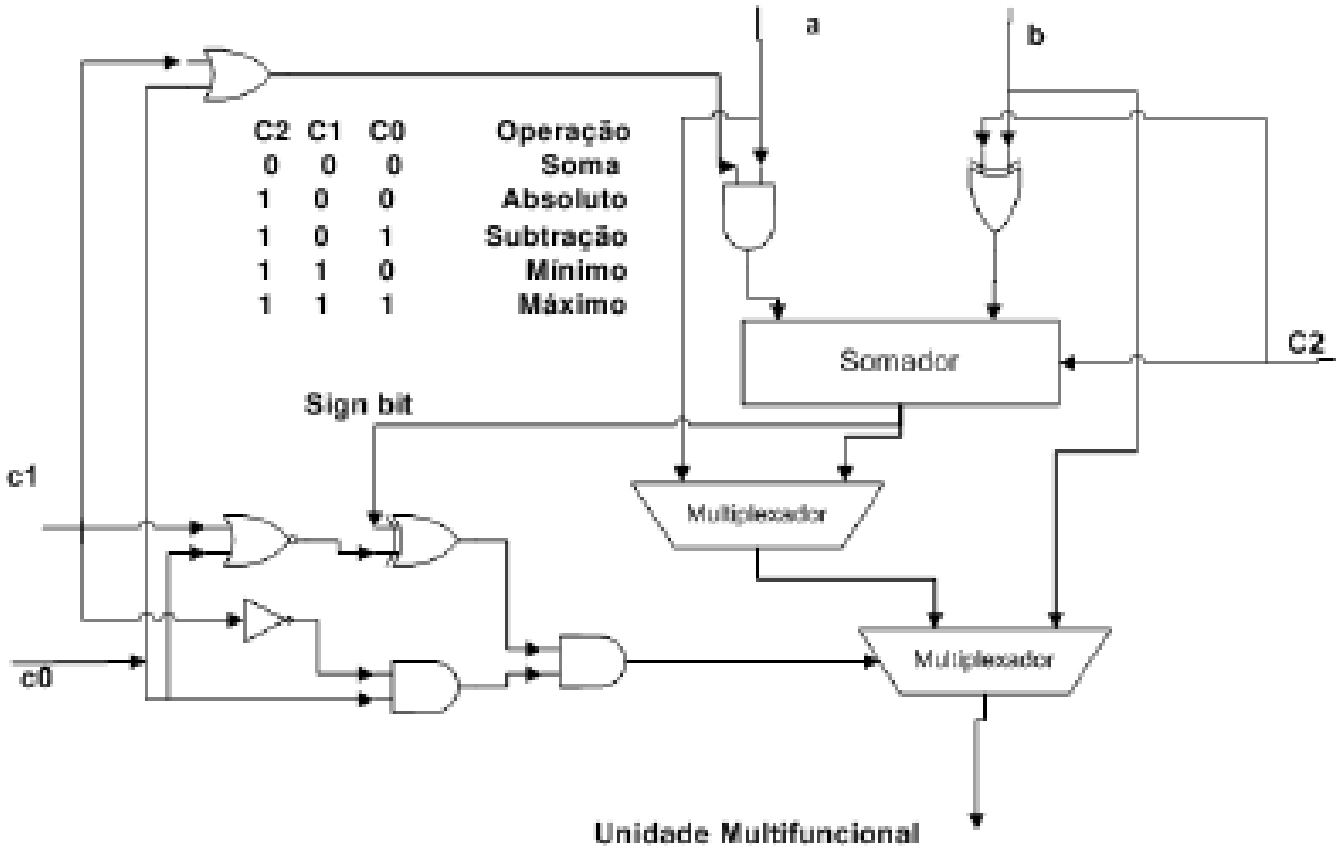
Unidade Multi Funcional: +, -, min e max



UF = {Soma, Sub, Min e Max }

Tprop = Tprop(xor) + Tsoma + Tprop(xor) + Tsel(mux1) + Tsel(mux2)
= 2ns + 3ns + 2ns + 3ns + 2ns = 12n s

Unidades Multifuncionais: +/-/min/max/||



Tprop = Tprop(lógica) + Tsoma + Tprop(lógica) + Tprop(lógica) + Tsel(mux)
= 2ns + 3ns + 2ns + 2ns + 3ns = 12n s

Tabela de Custos: UF individuais

UF/Componente	And	Not	XOR	Somador	Mux
|a|		1		1	1
|b|		1		1	1
min		1		1	1
max		1		1	1
+				1	
-		1		1	
Total		5		6	4

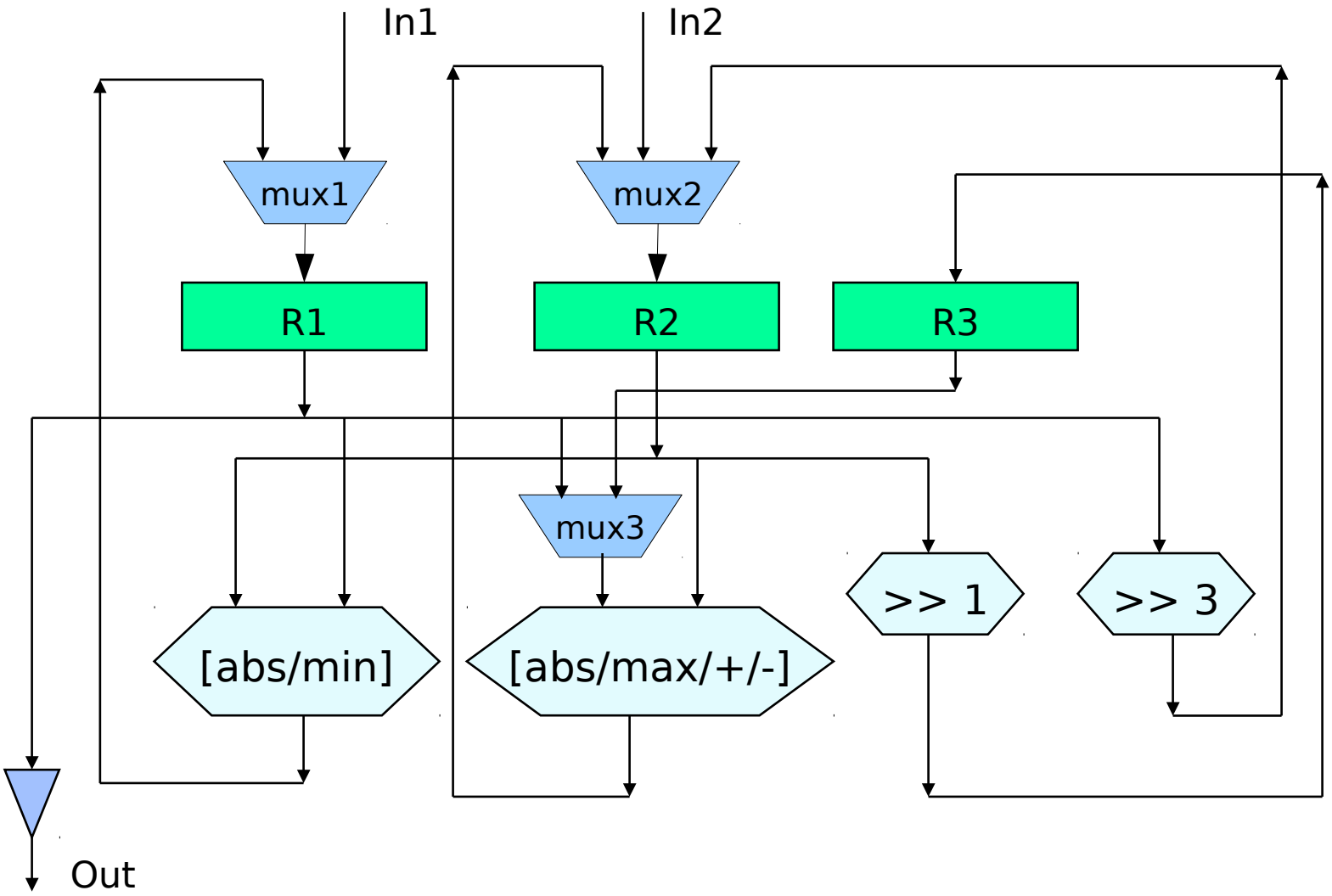
Considerando operandos de 1 bits.

Tabela de Custos: Multi Funcionais

UF\Componente	AND	NOT	XOR	Somador	MUX
[|a|, min]	**1**	**1**		**1**	**2**
[|b|, max, +, -]	**1**		**1**	**1**	**2**
Total	**2**	**1**	**1**	**2**	**4**
[|a|, min, +]	**1**		**1**	**1**	**2**
[|b|, max, -]	**1**	**1**		**1**	**2**
Total	**2**	**1**	**1**	**2**	**2**
[|b|, min, max]	**1**	**1**	**1**	**1**	**2**
[min, max, +, -]	**1**		**2**	**1**	**2**
[min, max,|b|,+,-]	**5**	**1**	**2**	**1**	**2**
Total	**7**	**2**	**5**	**3**	**6**

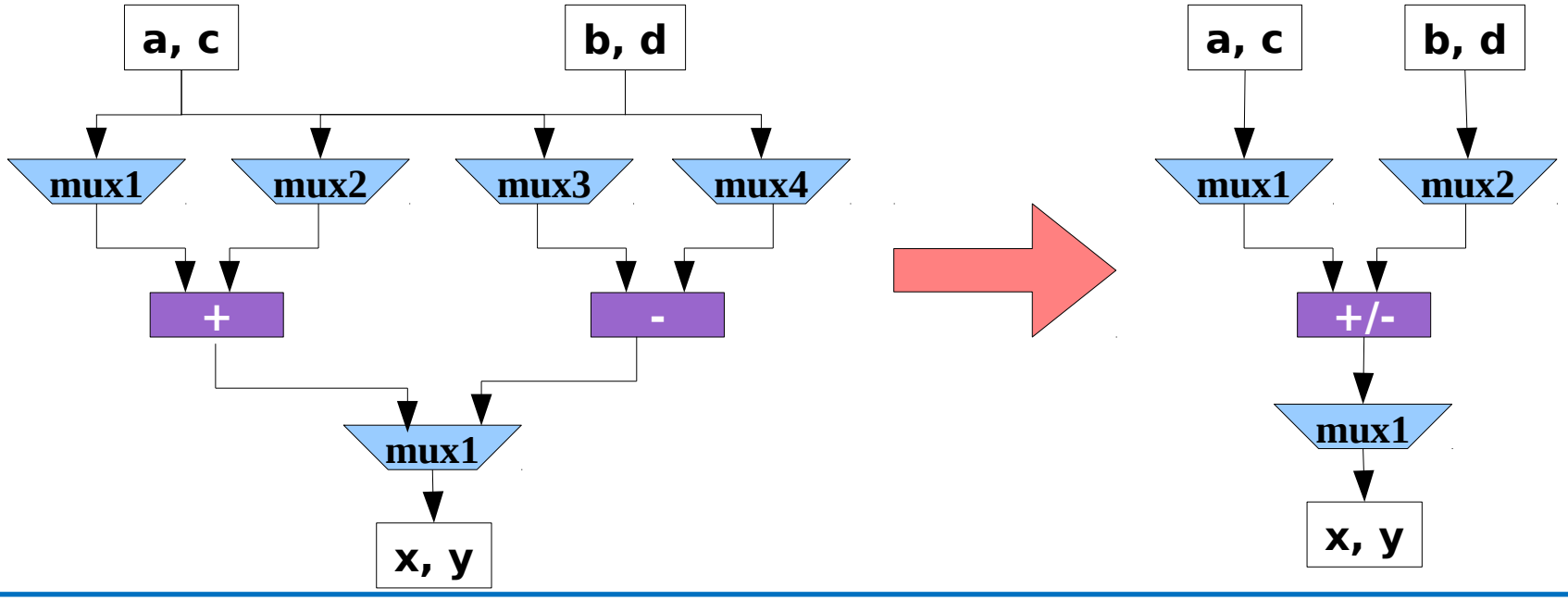
Considerando operandos de 1 bits.

Fluxo de Dados para Grafo Alternativa 1



Mais Minimização (1)

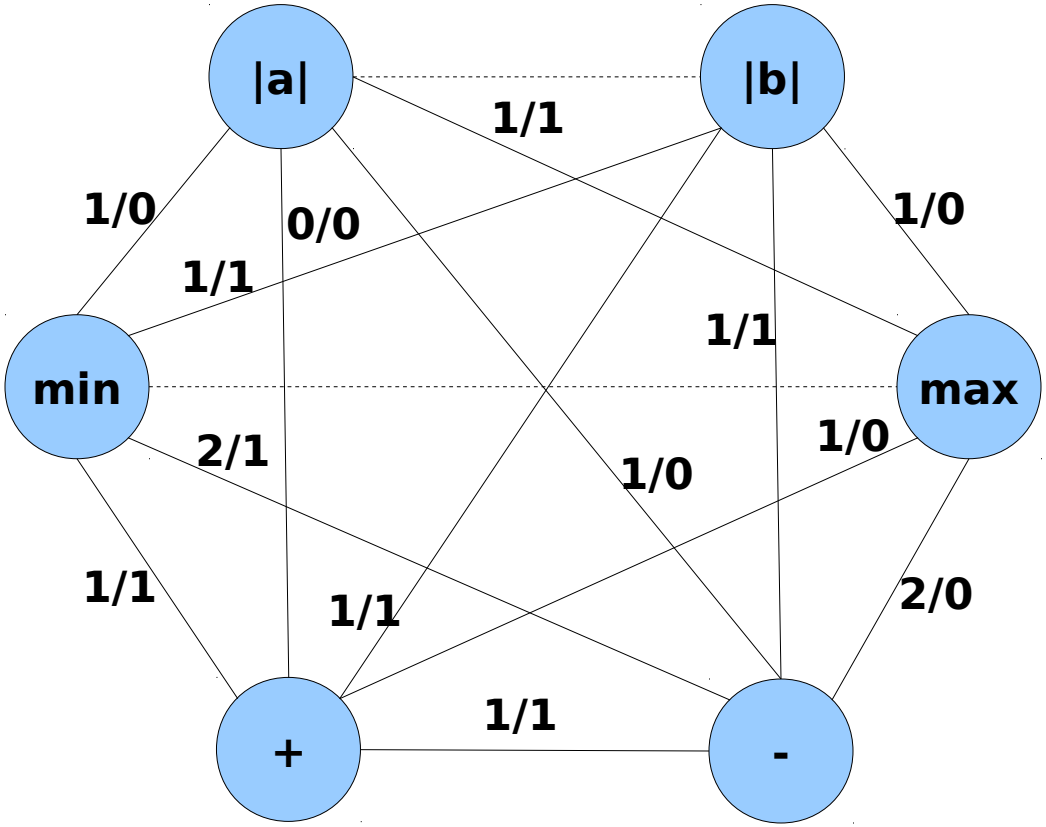
- Além de agrupar os operadores funcionais que não são utilizados no mesmo estado, pode-se fazer isto minimizando-se o número de conexões do FD.



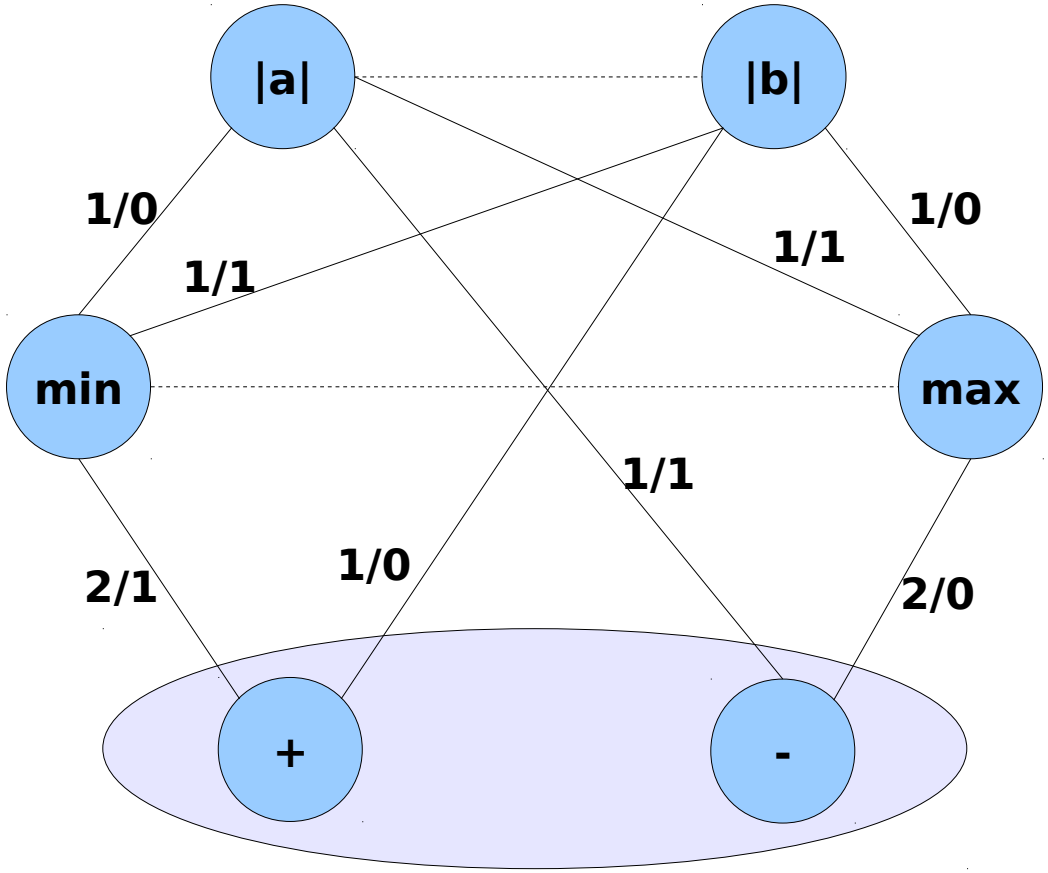
Mais Minimização (2)

- Para tanto, deve-se construir um grafo de compatibilidade com prioridades.
  - Similar a minimização de registradores.
- Dois operadores serão agrupados prioritariamente se eles possuírem origem ou destino comum.

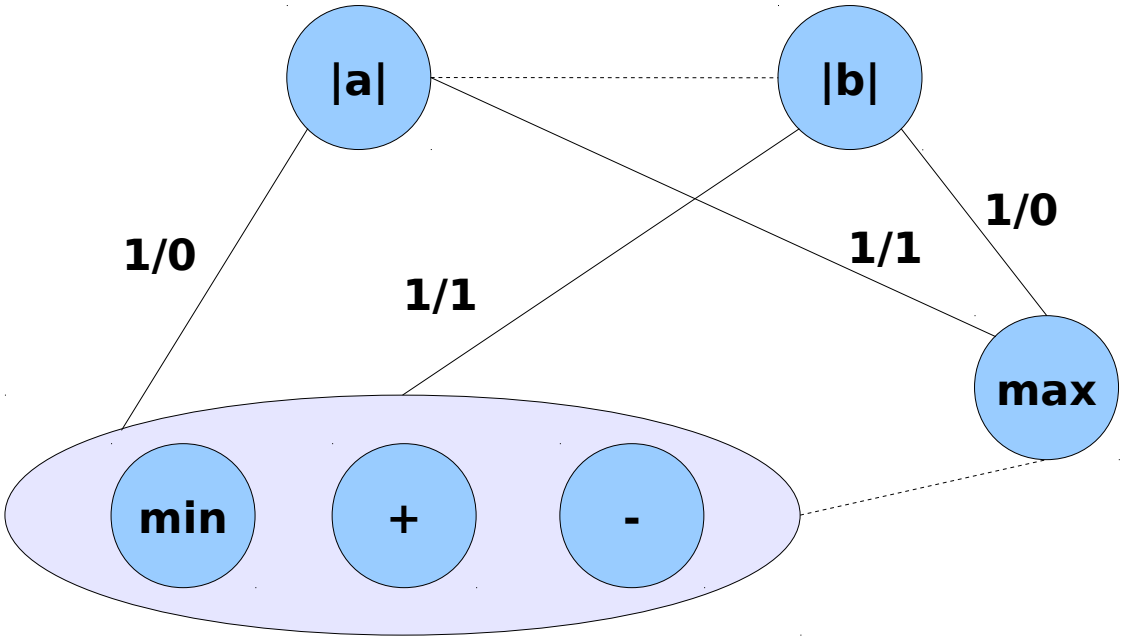
Grafo de Compatibilidade Inicial



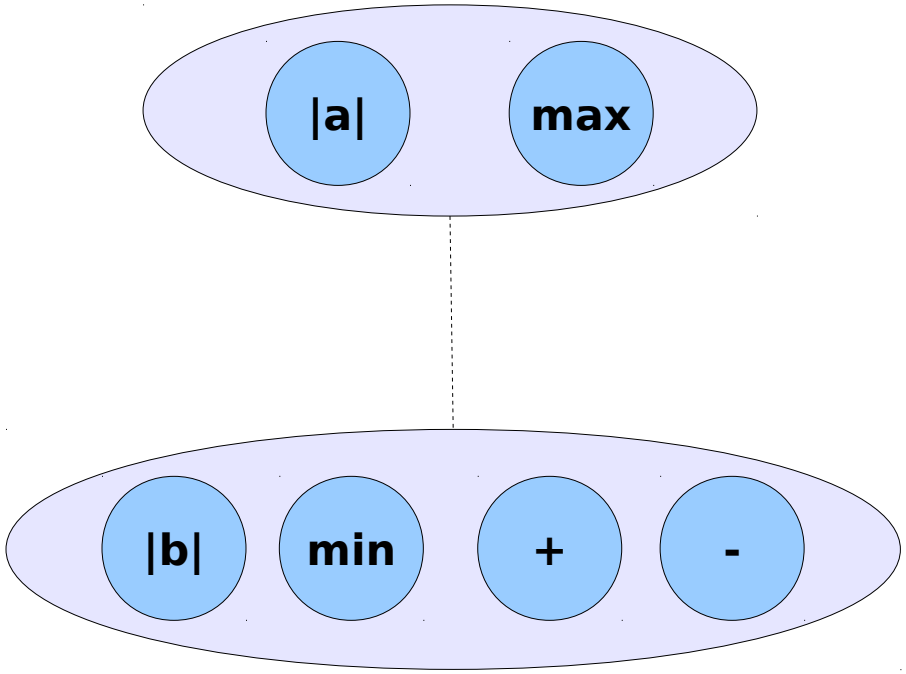
Grafo de Compatibilidade: junção de + e -



Grafo de Compatibilidade: junção de min, + e -



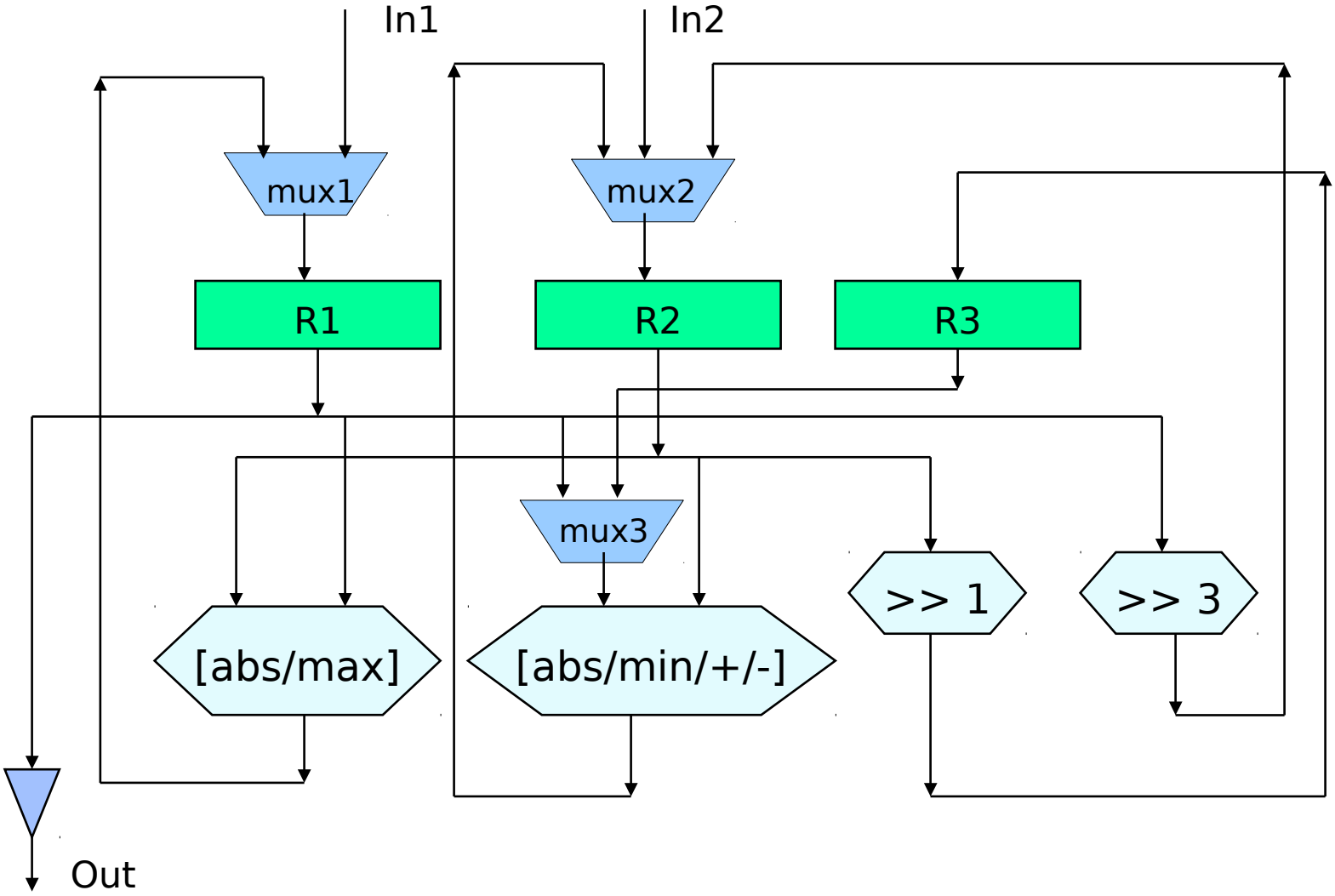
Grafo de Compatibilidade Final



Fluxo de Dados

- Após a minimização com o agrupamento de variáveis e de operadores, determina-se que o circuito pode ser implementado com 3 registradores e 4 unidades funcionais:
  - R1 = {a, t1, x, t7}, R2 = {b, t2, y, t3, t5, t6}, e R3 = {t4}
  - UF1 = [|b|, min, +, -], UF2 = [|a|, max], UF3 = [>>1], e UF4 = [>>3]

Fluxo de Dados: Registradores e Unidades Funcionais minimizados



Minimização de Vias ou Interconexões

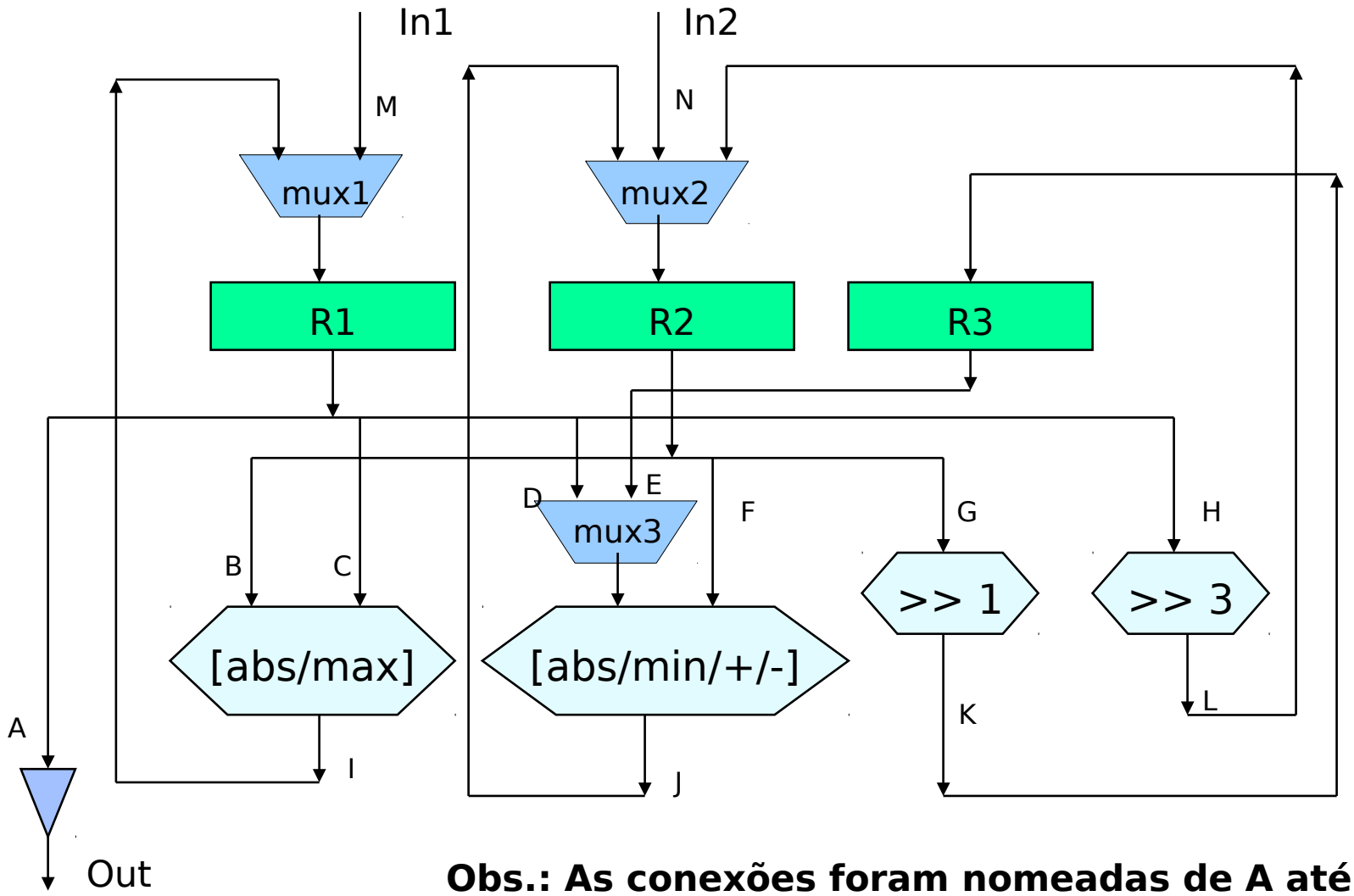
Agrupamento de interconexões (1)

- O último elemento que falta para ser agrupado!
- Interconexões que não são utilizadas simultaneamente podem ser agrupadas formando vias múltiplas (barramentos ou *buses*).
- As saídas de registradores e de unidade funcionais são chamadas de fonte de conexões.
- Enquanto que as suas entradas são chamadas de destino de conexões.

Agrupamento de interconexões (2)

- A técnica de se agrupar conexões é similar à de se agrupar variáveis ou operações: utiliza-se um grafo de compatibilidade para minimização das vias.
  - **Primeiro passo**: escrever a Tabela de Conexões do FD.
  - **Segundo passo**: desenhar grafo de compatibilidade de entradas e de saídas.
  - **Terceiro passo**: agrupar vias compatíveis.

Fluxo de Dados: minimizar interconexões

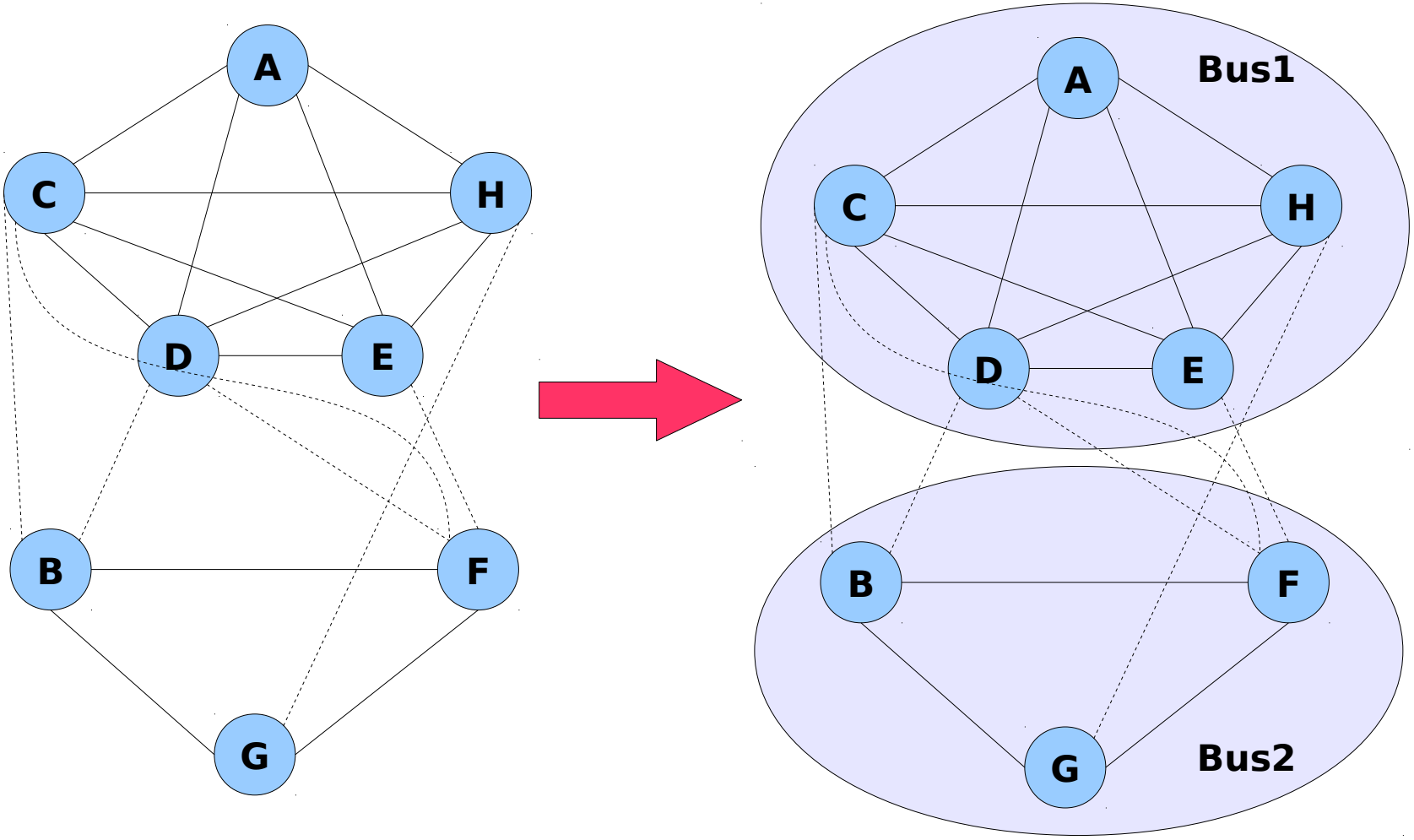


Obs.: As conexões foram nomeadas de A até N.

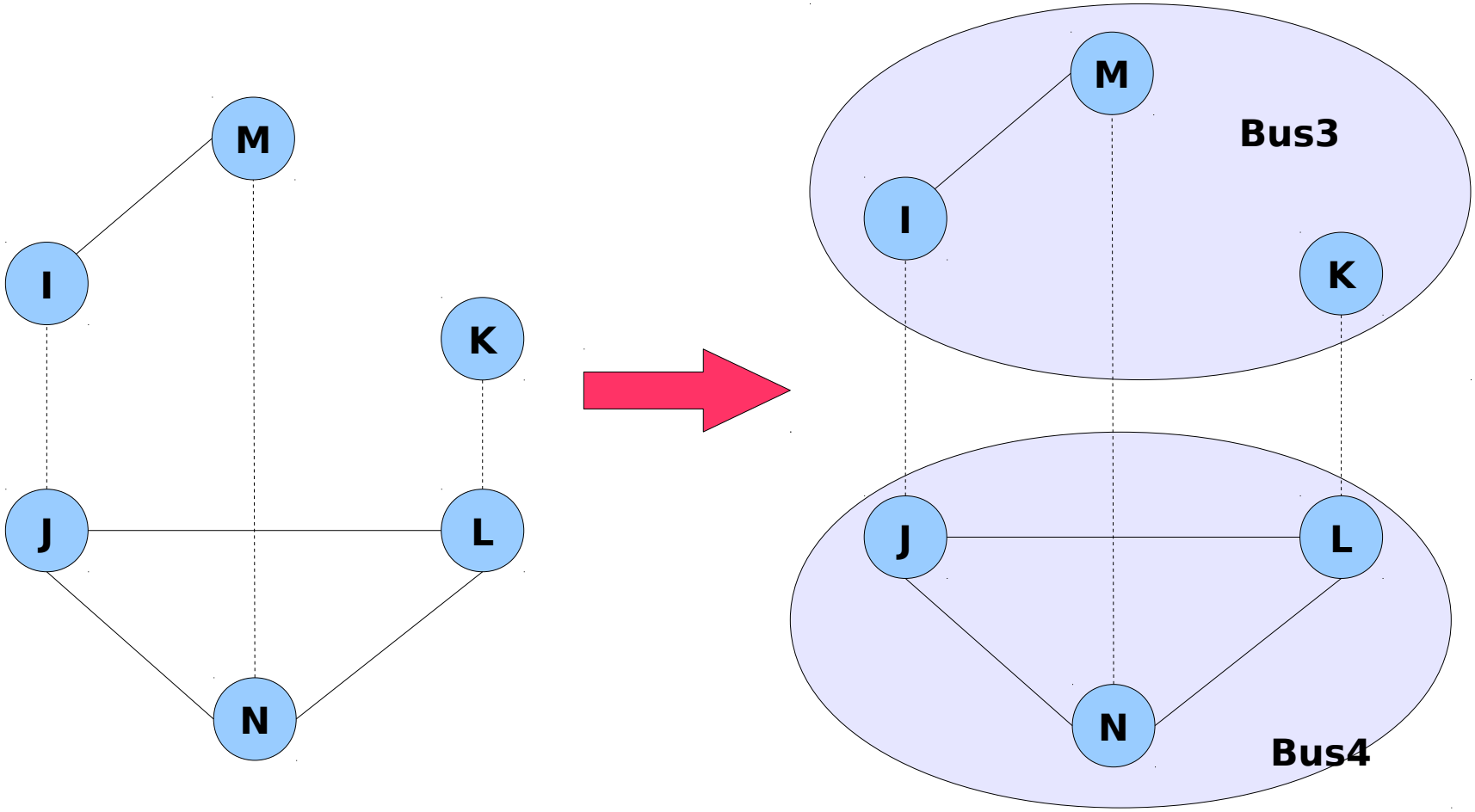
Tabela de uso de Conexões

	S0	S1	S2	S3	S4	S5	S6	S7
A								X
B			X				X	
C		X	X				X	
D			X		X			
E						X		
F		X	X		X	X		
G				X				
H				X				
I		X	X				X	
J		X	X		X	X		
K				X				
L				X				
M	X							
N	X							

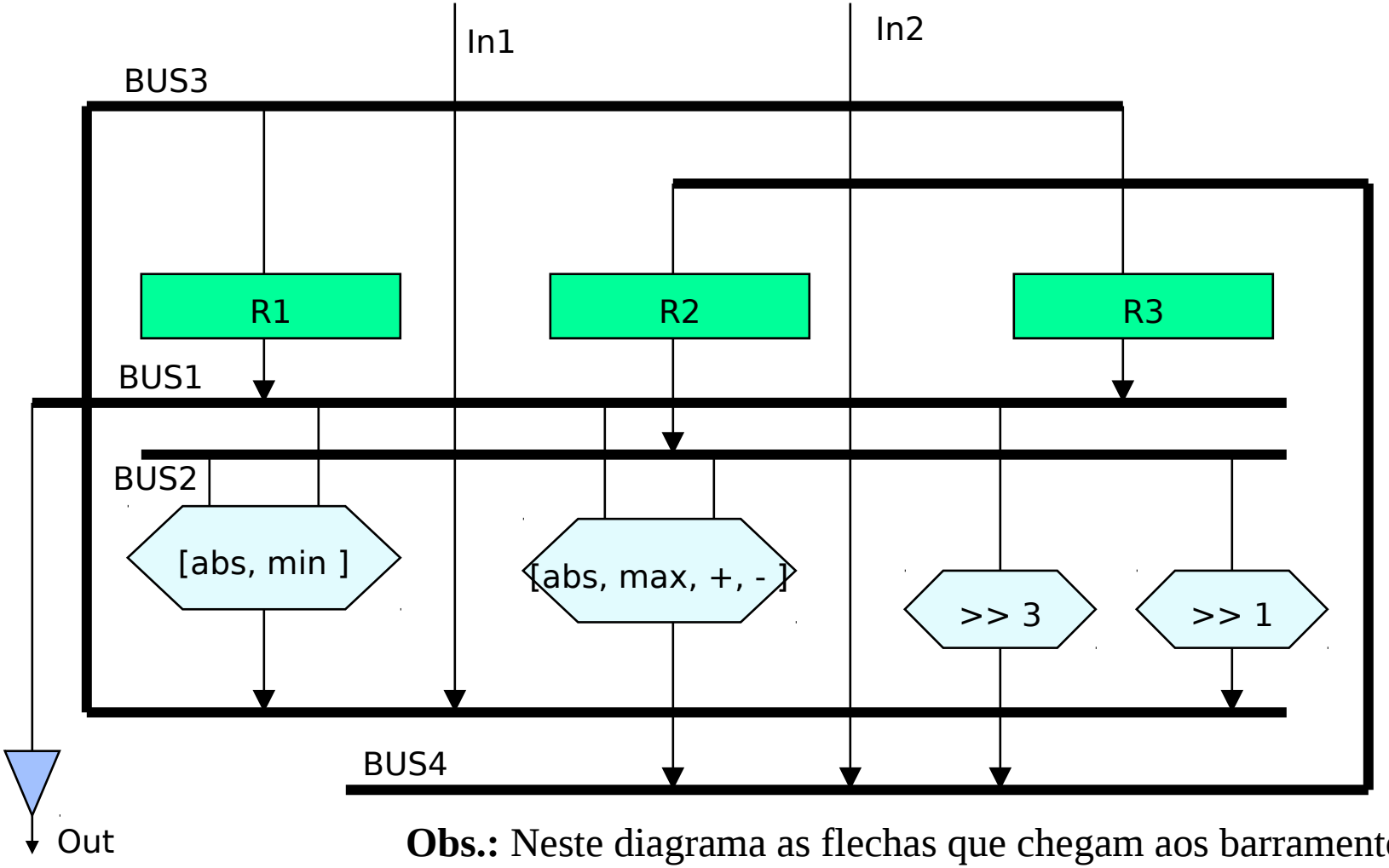
Grafo de Compatibilidade: Vias de Entrada



Grafo de Compatibilidade: Vias de Saída



Fluxo de Dados após minimização de vias



Obs.: Neste diagrama as flechas que chegam aos barramentos significam portas de terceiro estado.

Agrupamento de Registradores

Agrupamento de registradores em bancos de registradores (1)

- Quando dois ou mais registradores não são acessados simultaneamente em nenhum estado eles podem residir numa mesma rede de registradores (“Register File”).
- Estes registradores irão partilhar as mesmas portas da rede de registradores e consequentemente diminuindo o número de conexões do circuito.

Agrupamento de registradores em bancos de registradores (2)

- Para se proceder a esta otimização, faz-se um grafo de compatibilidade.
  - Similar a variáveis, operadores e interconexões.
- Registradores que não são nunca acessados ao mesmo tempo são compatíveis.
  - 1. Montar tabela de acesso a registradores em cada estado;
  - 2. Desenhar grafo de compatibilidade;
  - 3. Identificar agrupamentos.

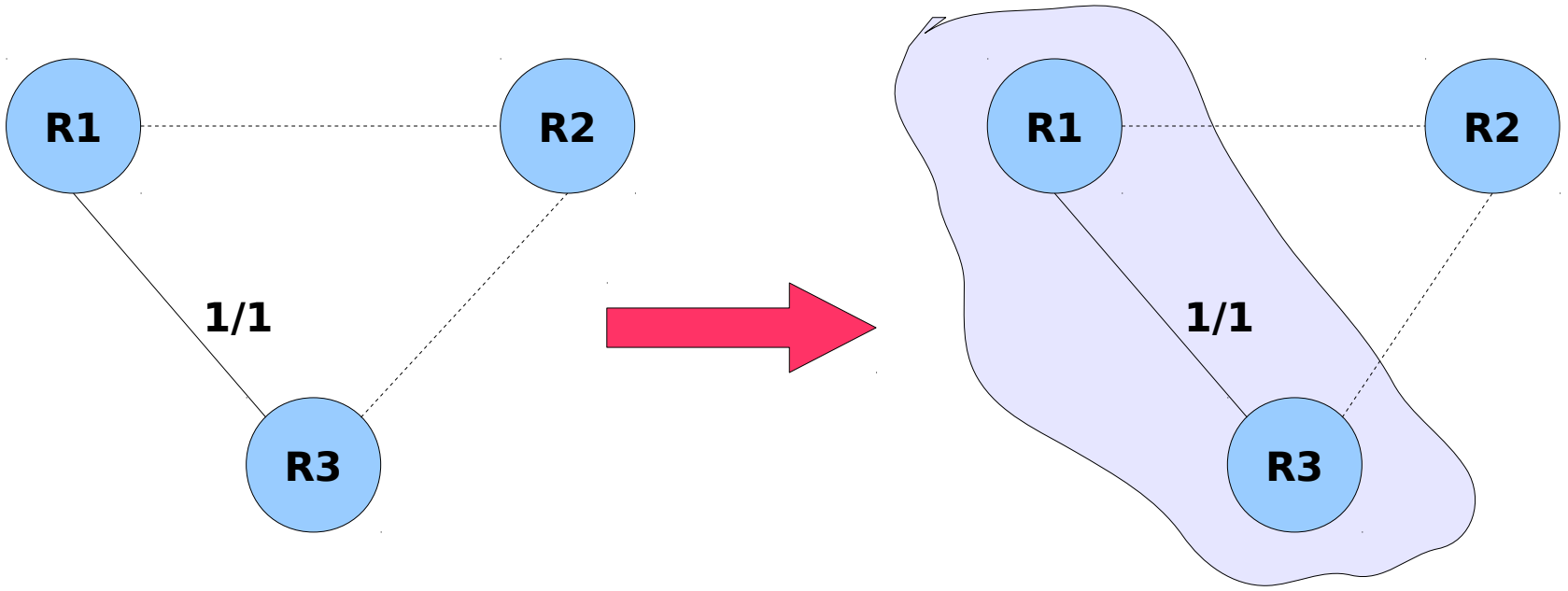
Tabela de Acesso aos Registradores

R1={a, t1, x, t7}
R2={b, t2, y, t3, t5, t6}
R3={t4}

	S0	S1	S2	S3	S4	S5	S6	S7
R1	E	L E	L E	L	L		L E	L
R2	E	L E	L E	L E	L E	L E	L	
R3				E		L		

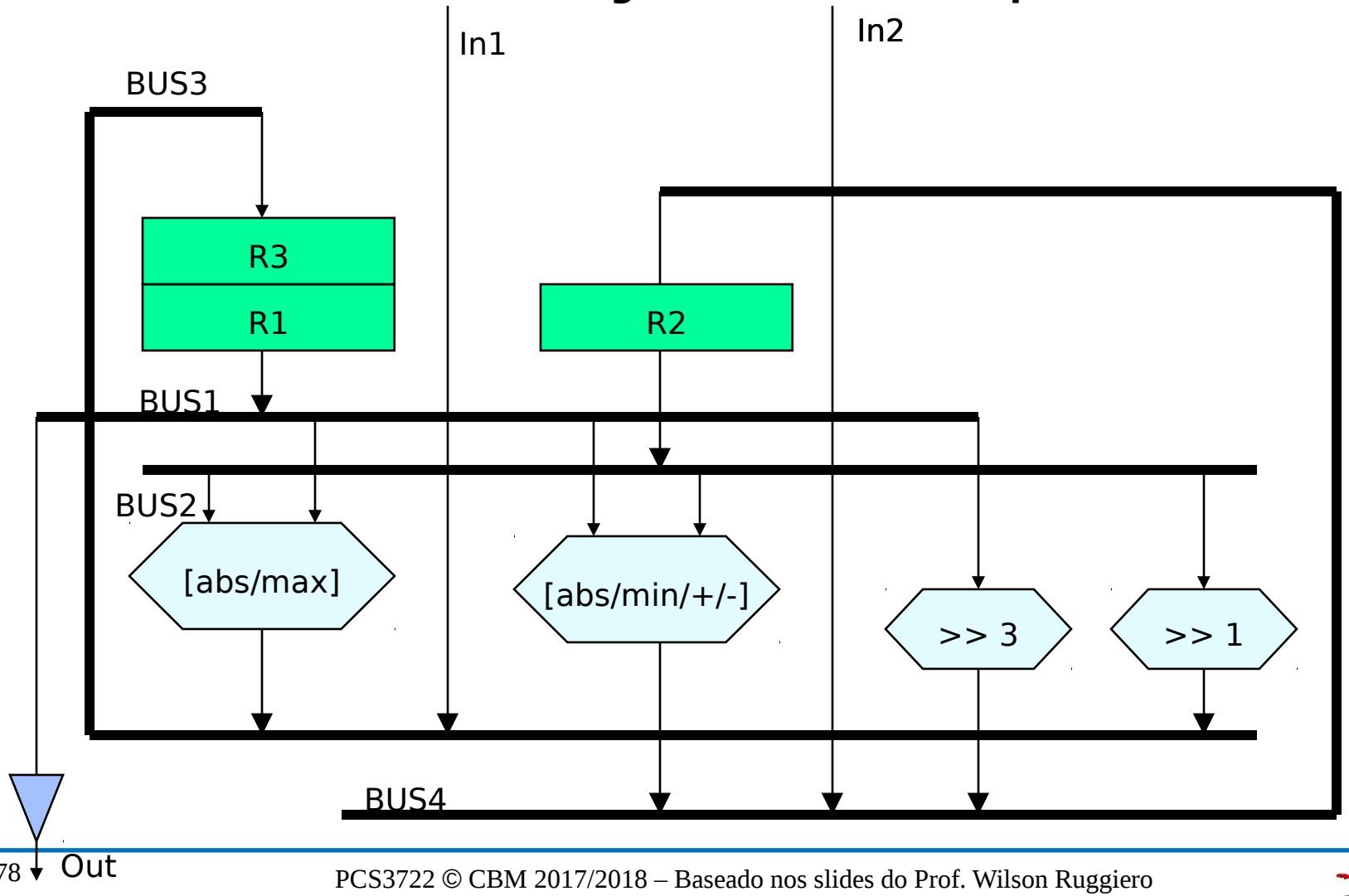
E = escrita
L = leitura

Grafo de Compatibilidade



Projeto Final do Fluxo de Dados

Obs.: usando um banco de registradores de uma porta só.

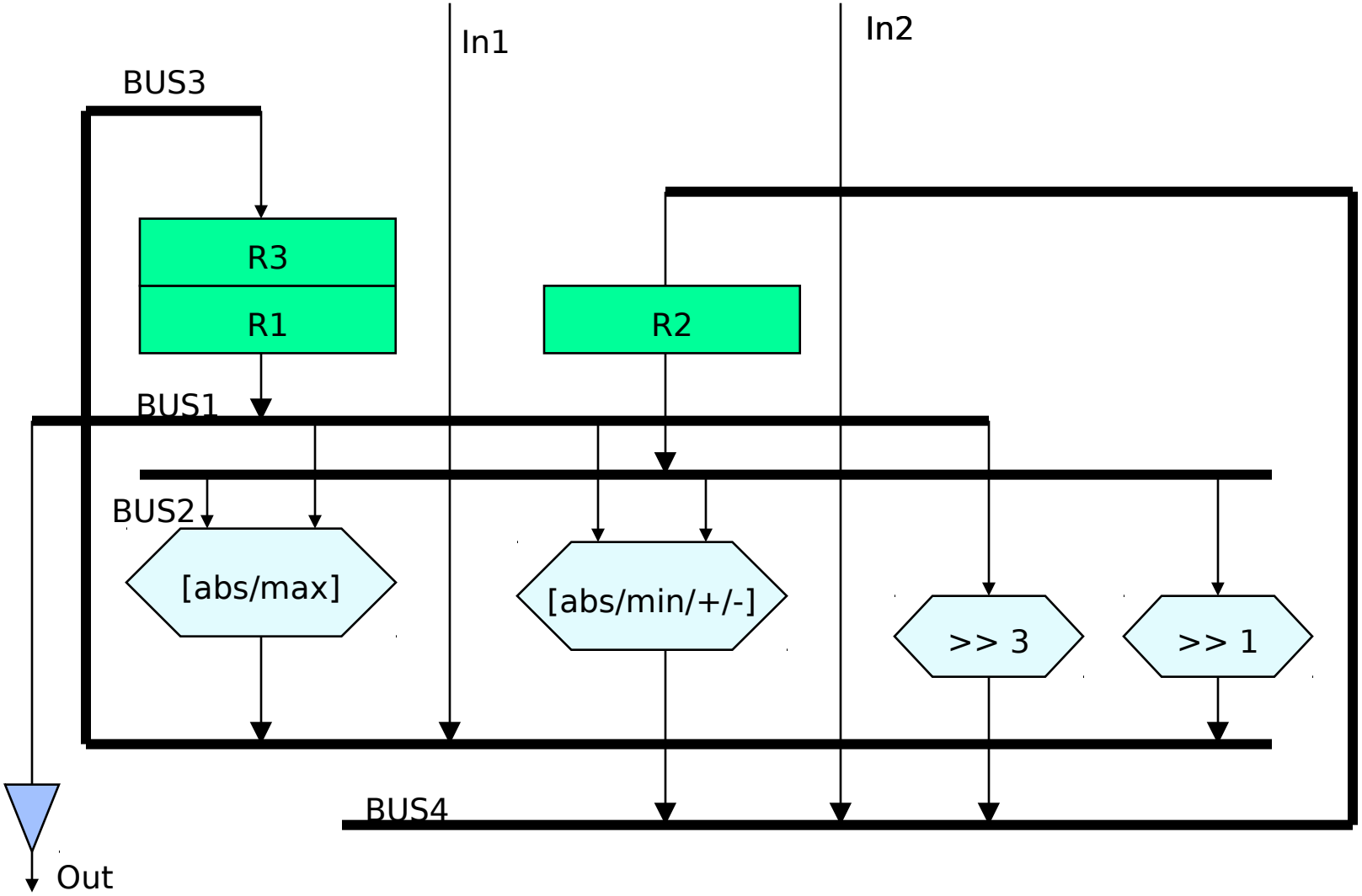


Resumo e Complementação: Etapas do projeto do Fluxo de Dados

Etapa 1: Síntese da estrutura do Fluxo de Dados

- Em função da especificação do módulo digital e do diagrama ASM em nível de algoritmo, aplicam-se os métodos de mapeamento com minimização: variáveis em registradores, operações em unidades funcionais e transferências em interconexões.
- Como resultado desta etapa obtém-se a estrutura do fluxo de dados: registradores e unidades funcionais interconectados.

Resultado após a Etapa 1

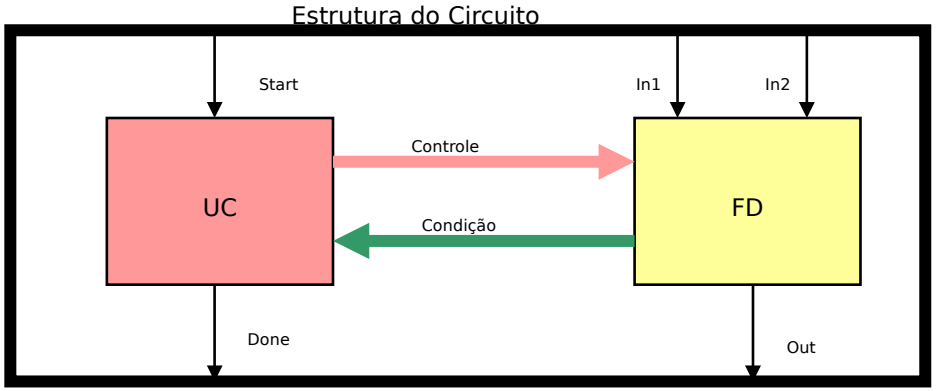


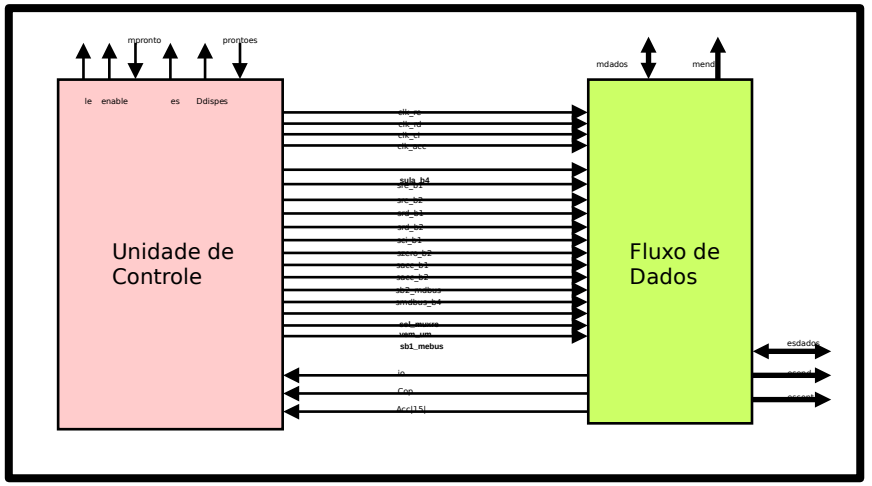
Etapa 2: Escolha de Componentes

- Baseado na estrutura do fluxo de dados obtida na etapa 1, requisitos de desempenho do projeto, e catálogos de componentes.
- Determinam-se os componentes para os registradores, as unidades funcionais e os elementos de interconexão (multiplexadores e portas lógicas).
- Quando uma unidade funcional não possui implementação direta através de um componente físico da biblioteca de componentes disponível, ela deve ser sintetizada.

Etapa 2: Escolha de Componentes (cont.)

- O resultado produzido nesta etapa é o circuito do fluxo de dados com todos os componentes escolhidos e os sinais de controle e sinais de condição determinados.





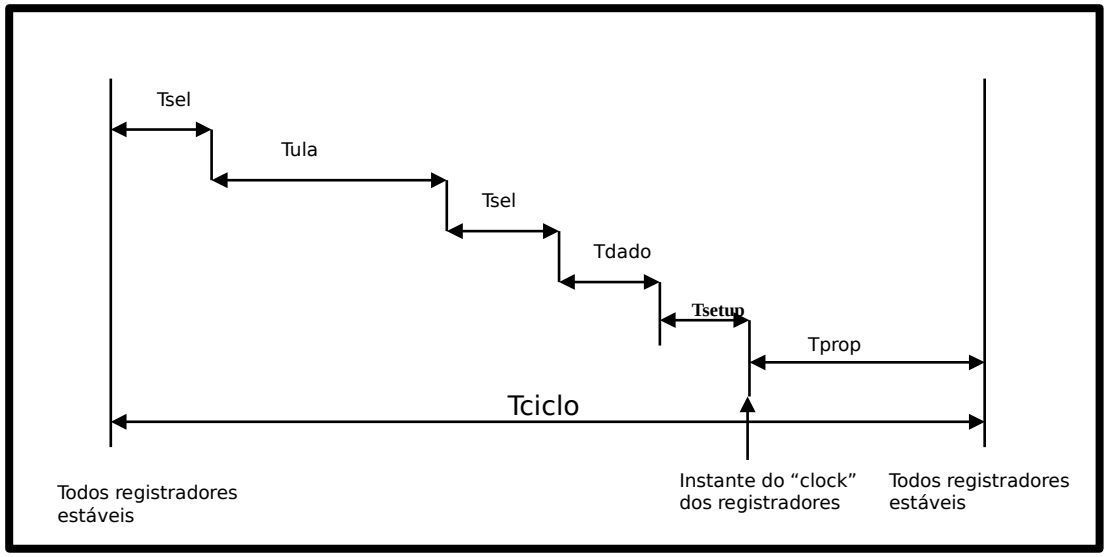
Etapa 3: Cálculo do Ciclo do FD

- O ciclo do fluxo de dados é o intervalo de tempo necessário para realizar a operação mais demorada do sistema.
- Dentro do ciclo do fluxo de dados deve-se identificar os instantes relevantes, ou seja, todos os instantes que devem ser acionados sinais de clock em registradores.
- Baseado nesses instantes relevantes determinam-se todos os sinais de tempo necessários para a operação do fluxo de dados.

Etapa 3: Cálculo do Ciclo do FD

- Nesta etapa obtêm-se o valor do ciclo do fluxo de dados e todos os sinais de tempo que devem ser sintetizados pela unidade de controle.

Resultado após Etapa 3



- **Tsel : Tempo de seleção de porta para bus => Tsel = 10 ns**
- **Tsmux : Tempo de seleção de multiplexador ==> Tsmux = 15 ns;**
- **Tula : Tempo de operação da ULA ==========> Tula = 35 ns;**
- **Tdmux : Tempo de dado de multiplexador =====>Tdmux = 10 ns;**
- **Tsetup : Tempo de set-up de registrador ======> Tsetup = 10 ns;**
- **Tprop : Tempo de propagação de registrador==> Tprop = 30 ns;**

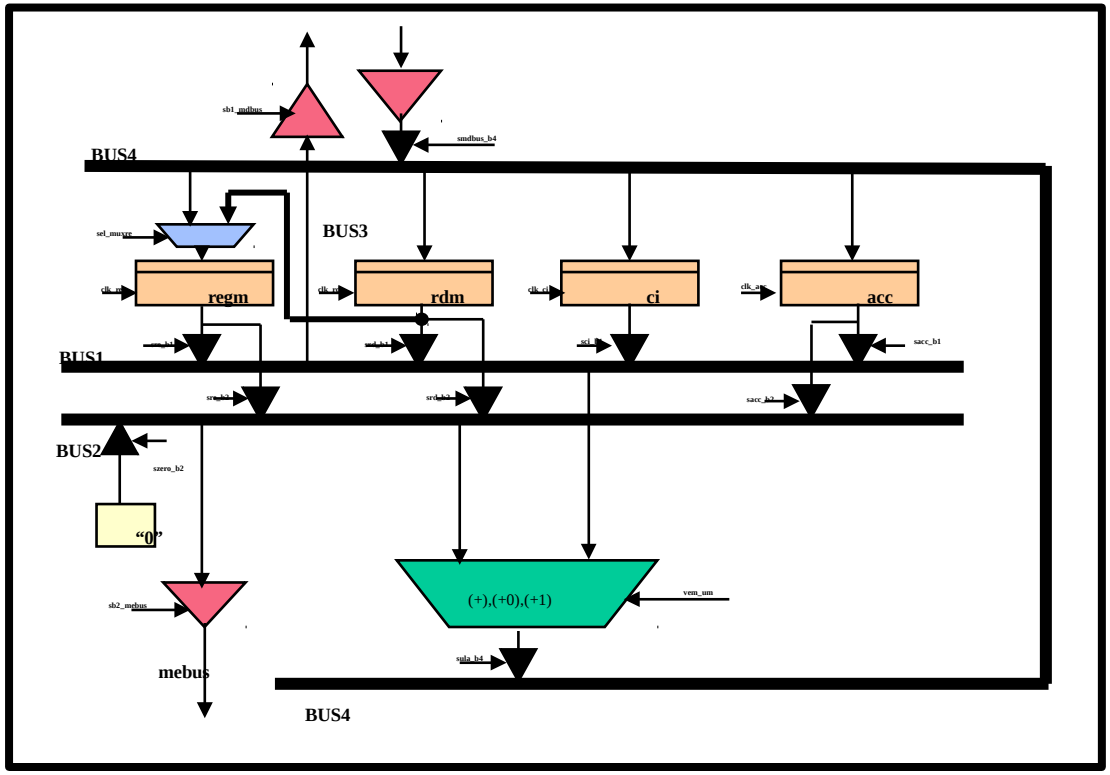
Tciclo = Tsel + Tula + Tsel + Tdado+ Tsetup + Tprop >= 105 ns;

- **Tclock = Tsel + Tula + Tsel + Tdado+ Tsetup >= 75 ns.**

Etapa 4: Descrição da estrutura do FD em VHDL

- Descrição do circuito projetado do fluxo de dados em VHDL com o objetivo de realizar uma simulação para testar o resultado de projeto alcançado.
- Cada um dos componentes do FD deve ser descrito através de uma descrição de comportamento.
- A interconexão entre os componentes do FD deve ser descrita em uma descrição de estrutura em VHDL.

Resultado após Etapa 4




```
ARCHITECTURE struct_2  OF reg4 IS
  COMPONENT reg1 IS
    PORT (d, clk : IN level;
          q : OUT level);
  END COMPONENT reg1;
  CONSTANT enabled : level := '1';
  FOR ALL : reg1 USE
   work.dff(behav)
     PORT
   MAP(d=>d,clk=>clk,enable=>enabl
   ed,q=>q,qn=>OPEN);
 BEGIN
    r0 : reg1 PORT MAP
   (d=>d0,clk=>clk,q=>q0);
    r1 : reg1 PORT MAP
   (d=>d1,clk=>clk,q=>q1);
    r2 : reg1 PORT MAP
   (d=>d2,clk=>clk,q=>q2);
    r3 : reg1 PORT MAP
   (d=>d3,clk=>clk,q=>q3);
END struct_2;
```

Etapa 5: Tradução do Diagrama ASM para uma descrição de chaveamento (1)

- Baseado nos sinais de controle e sinais de condição identificados, reescrever o diagrama ASM originalmente feito em nível de algoritmo para um outro em nível de chaveamento:
  - Transformar todas as atribuições mnemônicas para atribuições a sinais de controle e todos os testes a variáveis para testes de sinais de condição.

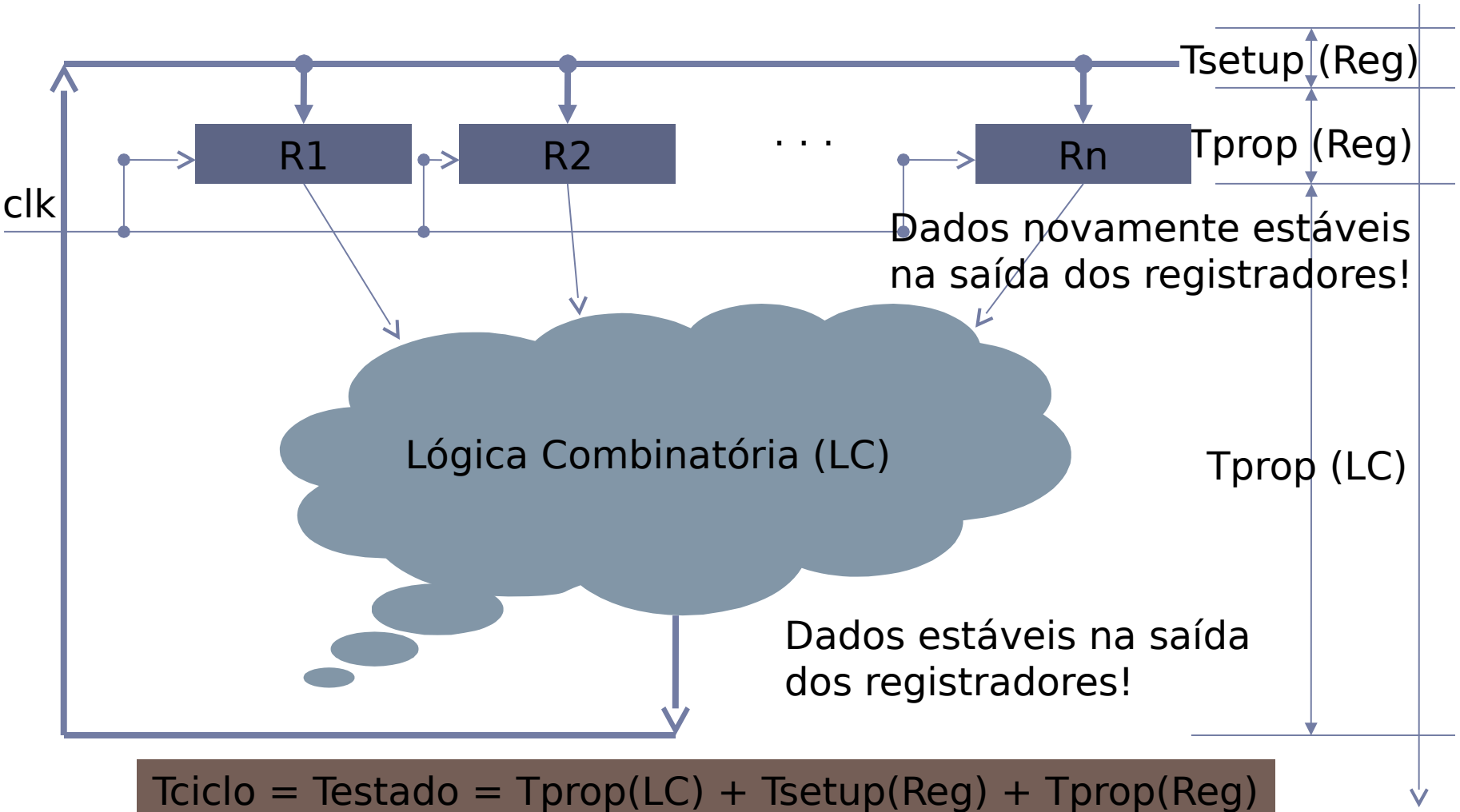
Etapa 5: Tradução do Diagrama ASM para uma descrição de chaveamento (2)

- O resultado desta etapa é um diagrama ASM que servirá de base para o projeto da Unidade de Controle. Este diagrama é a função de transferência da Unidade de Controle. Com esta descrição associada à descrição da estrutura do FD (etapa anterior) pode-se realizar uma simulação para testar o circuito projetado e a especificação da unidade de controle desenvolvida.

Ciclo do Fluxo de Dados (FD)

- O ciclo do FD é definido como o intervalo de tempo necessário para se executar a operação mais demorada.
  - Também é chamado de tempo de estado, e é igual para todos os estados do SD síncrono.
- A estrutura de um FD define um conjunto de registradores, cujos valores de saída passam por uma rede combinatória genérica que produz novos valores que devem ser armazenados em registradores (que podem ser os mesmos).

Estrutura do Fluxo de Dados



Tciclo = Testado = Tprop(LC) + Tsetup(Reg) + Tprop(Reg)

Desempenho: Como melhorar?

- Projeto do fluxo de dados voltado para o máximo desempenho, observa-se:
  - Tempo de execução;
  - Taxa de execução de operações na unidade de tempo.
- Dependendo do indicador de desempenho adotam-se métodos e organizações diferenciadas para otimizar o desempenho.

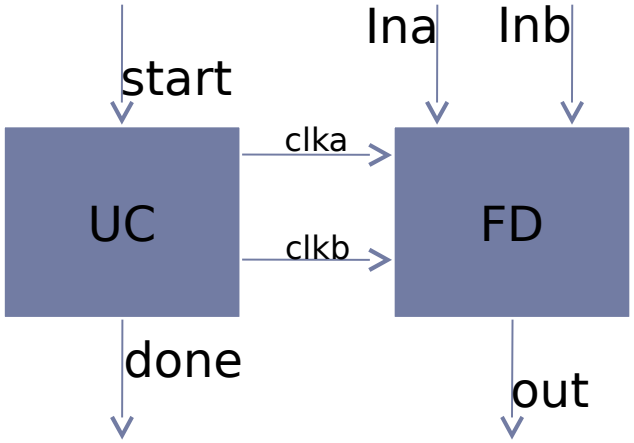
Desempenho: organizações

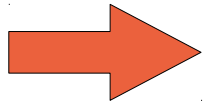
- Tipicamente, as organizações se alternam entre estruturas:
  - combinatórias com número mínimo de estados, ou
  - divididas no tempo e nas funções envolvendo múltiplos ciclos de execução sequenciados.

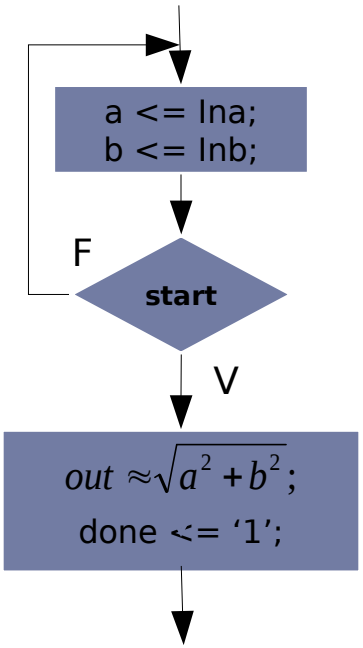
Qual indicador de desempenho considerar?

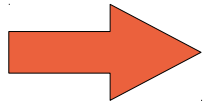
- Para minimizar o **tempo de execução** deve-se **minimizar o número de estados** utilizados para implementar o algoritmo funcional.
- Nestes casos, as unidades funcionais tendem a se agrupar formando unidades multifuncionais e no caso extremo, um bloco único com todas as funções necessárias compondo um circuito combinatório.
- Em geral, o FD se constitui em um conjunto de registradores que armazenam os dados de entrada e alimentam circuitos funcionais combinatórios que produzem o resultado final do algoritmo, que é disponibilizado em uma das saídas do sistema.

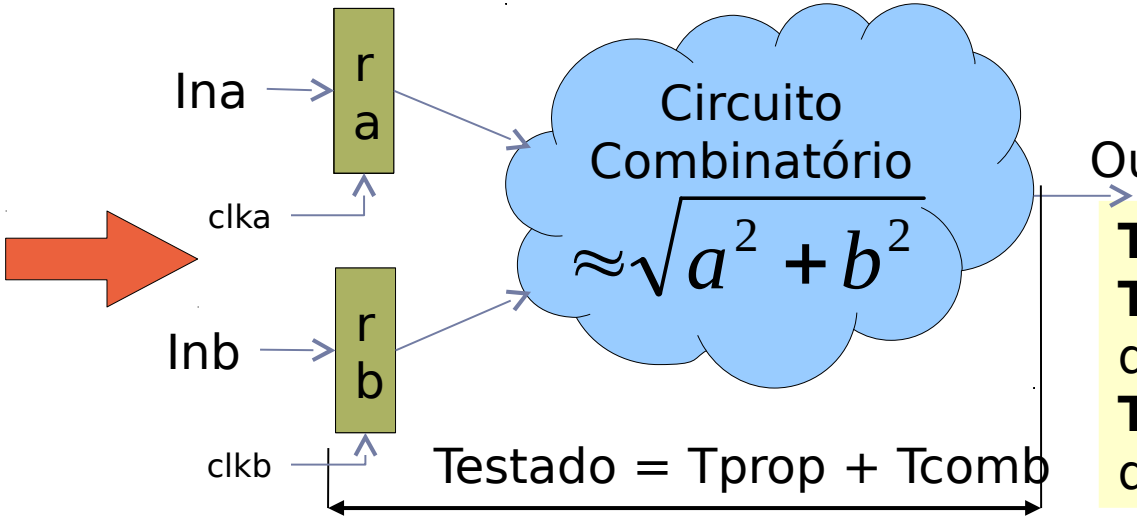
Exemplo: Calcule $\approx\sqrt{a^2+b^2}$





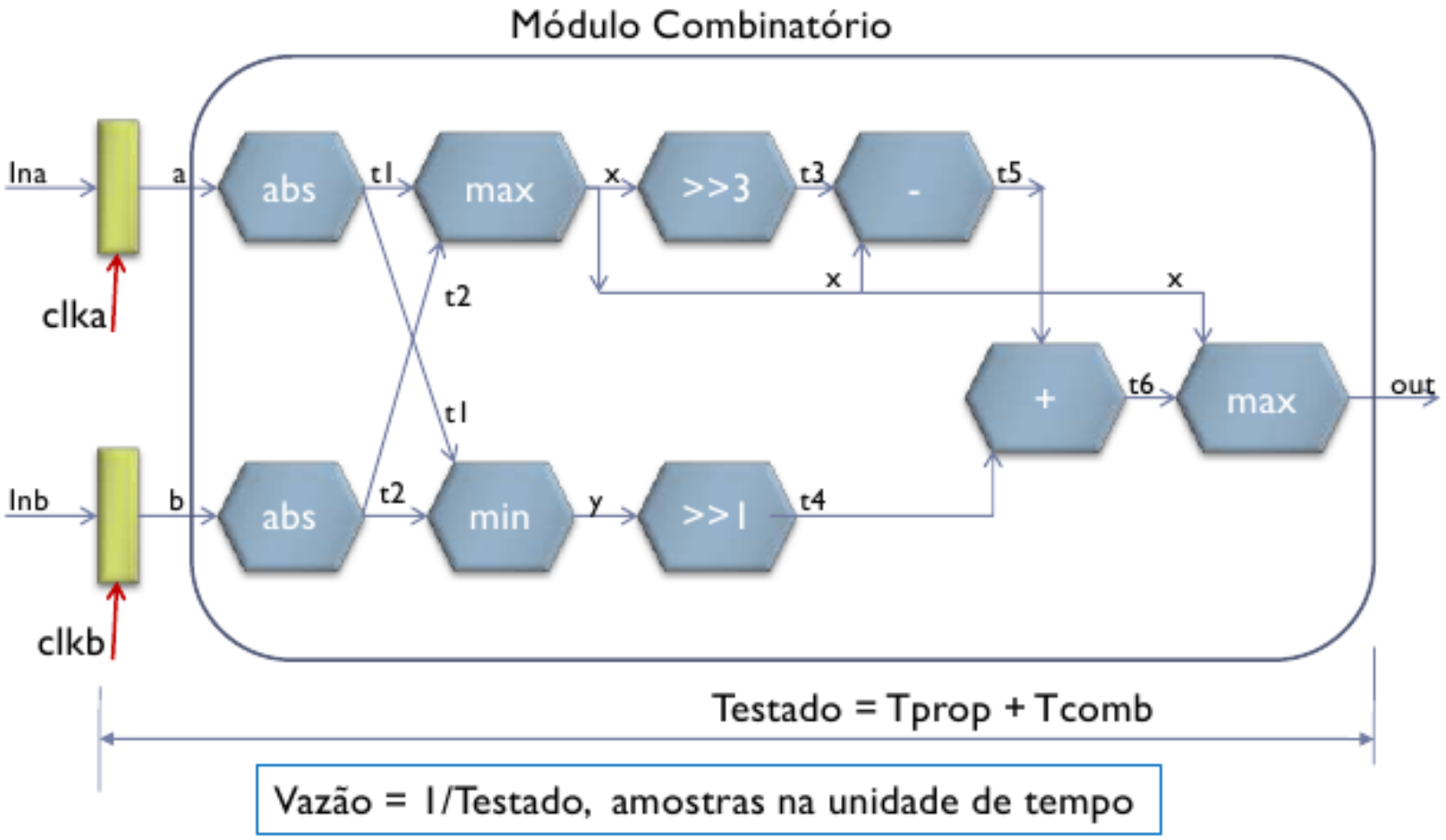




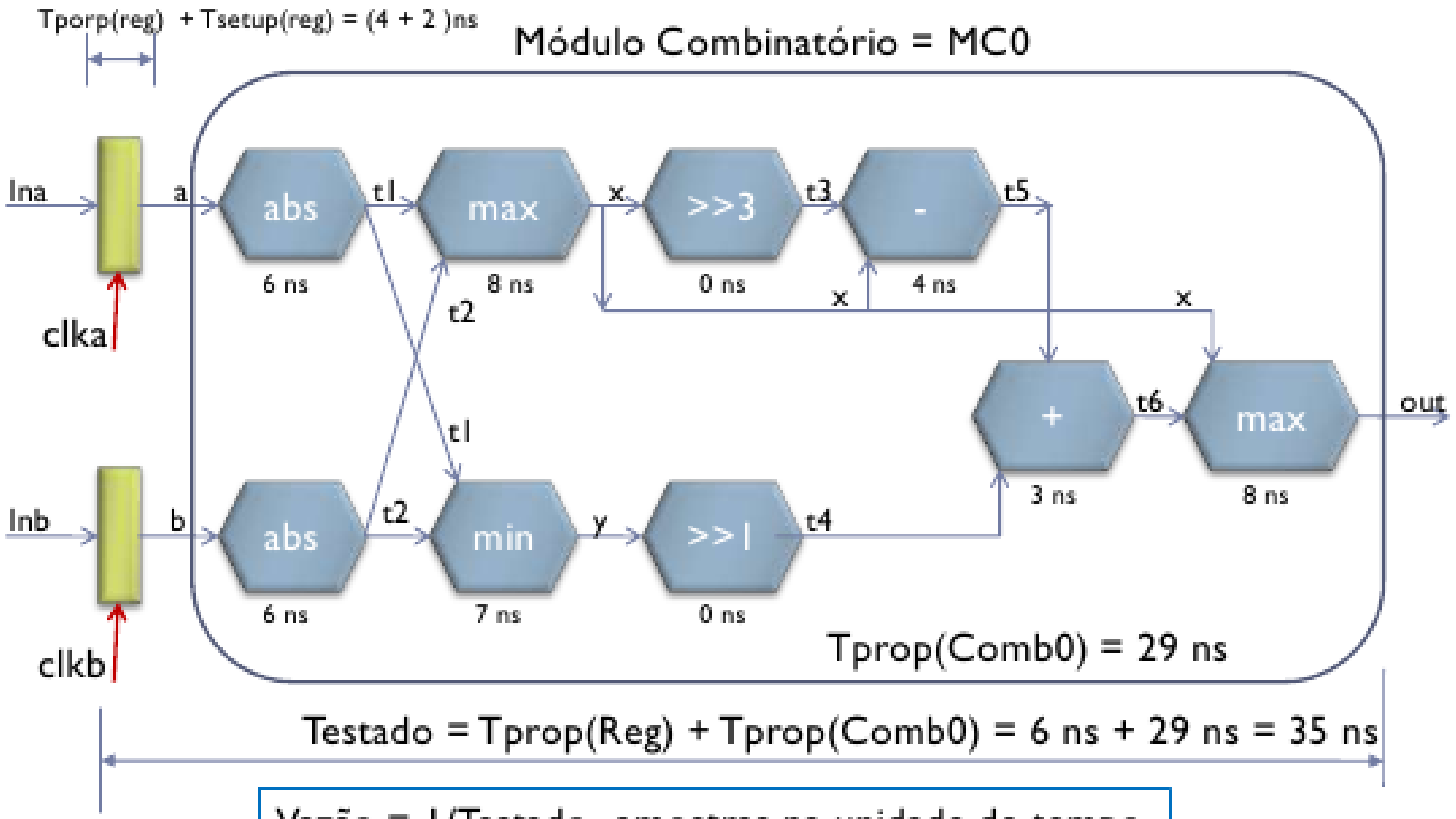


Testado: tempo de estado;
Tprop: tempo de propagação do registrador de entrada;
Tcomb: tempo de propagação do circuito combinatório.

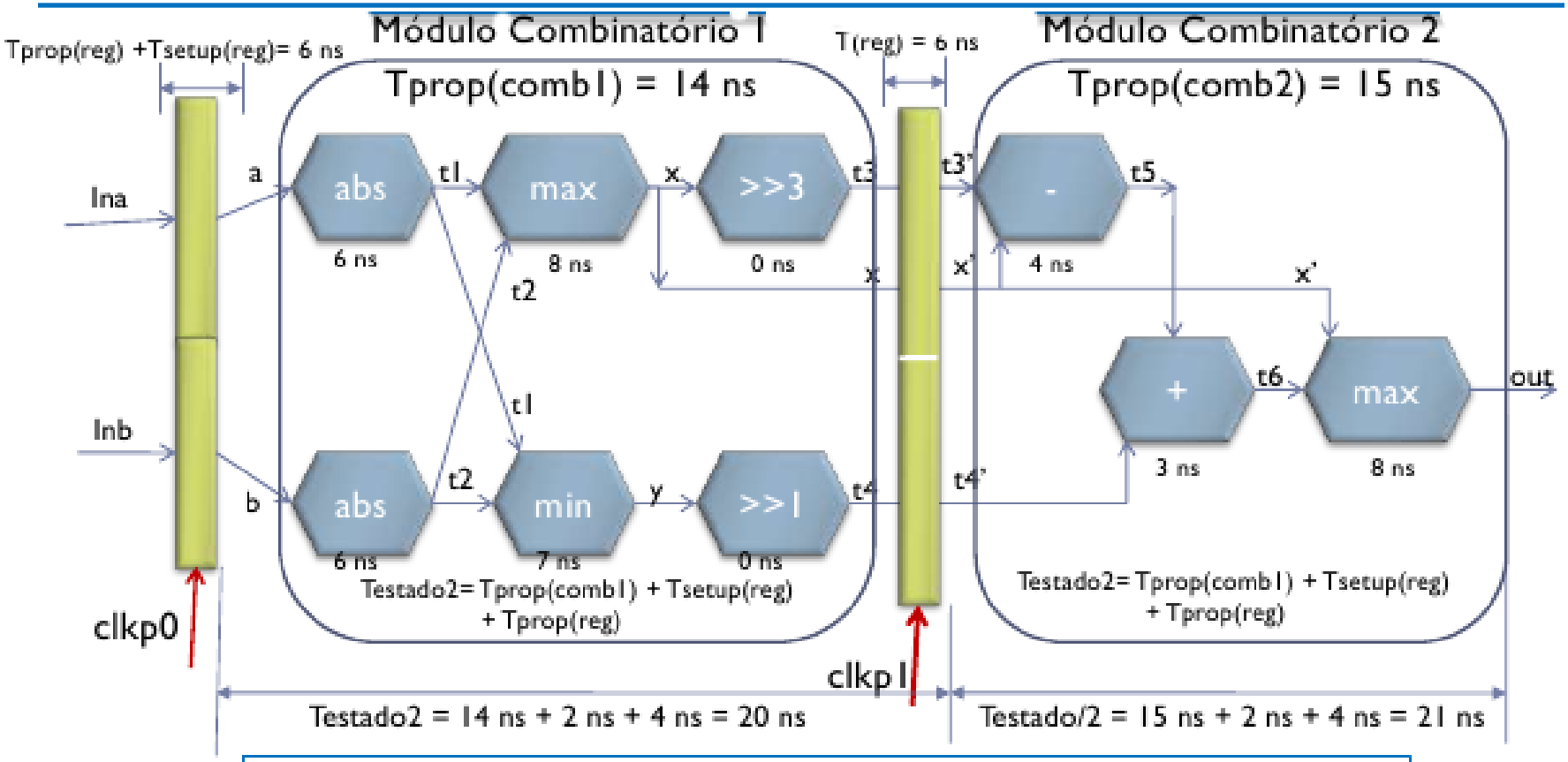
Módulo Combinatório



Módulo Combinatório



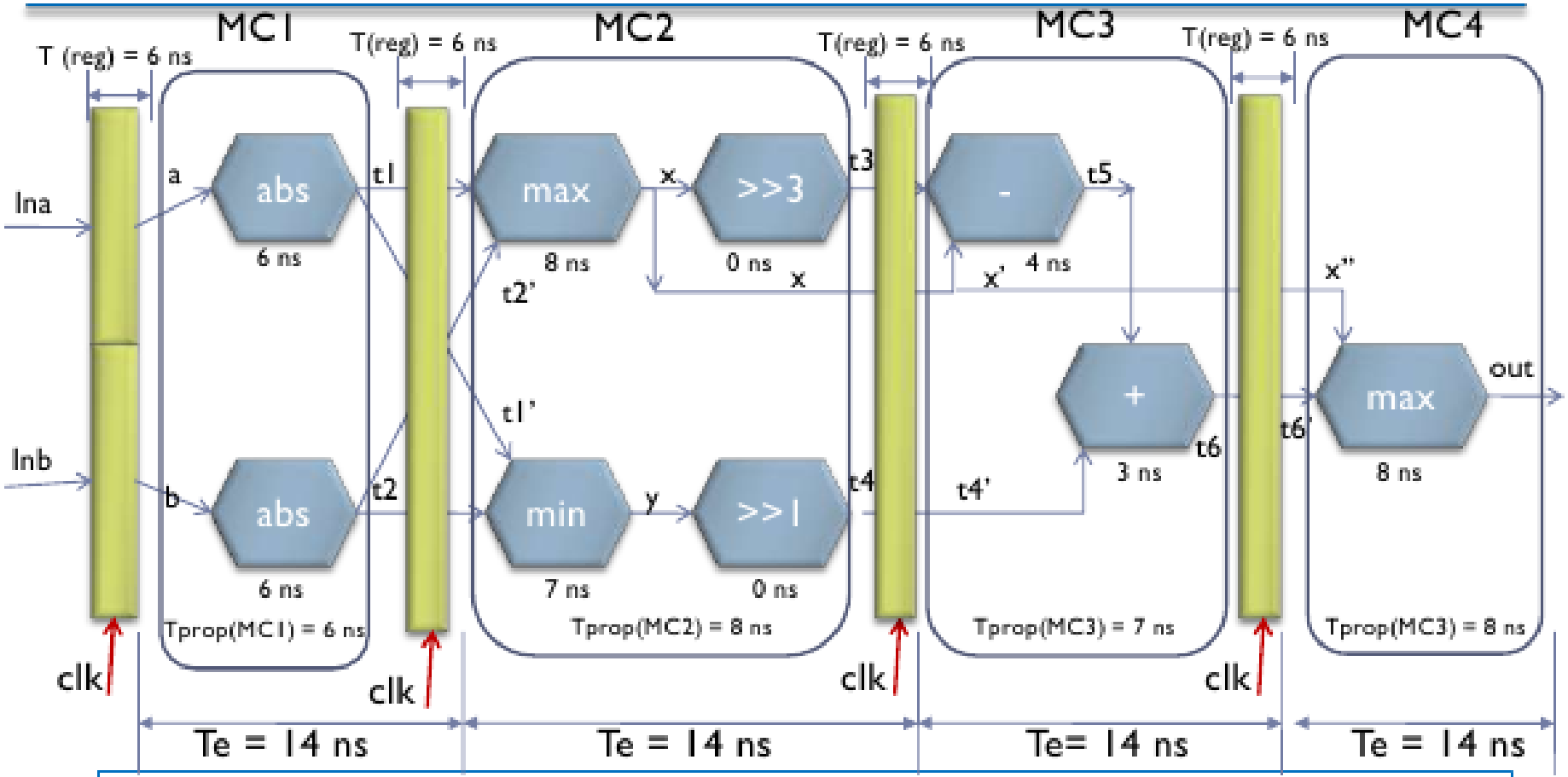
Adicionando estágios



Como fica o desempenho com a adição de um estágio intermediário?

Vazão = 1/Testado2, consegue-se o dobro da vazão anterior!
= 47,6 Mega amostras por segundo

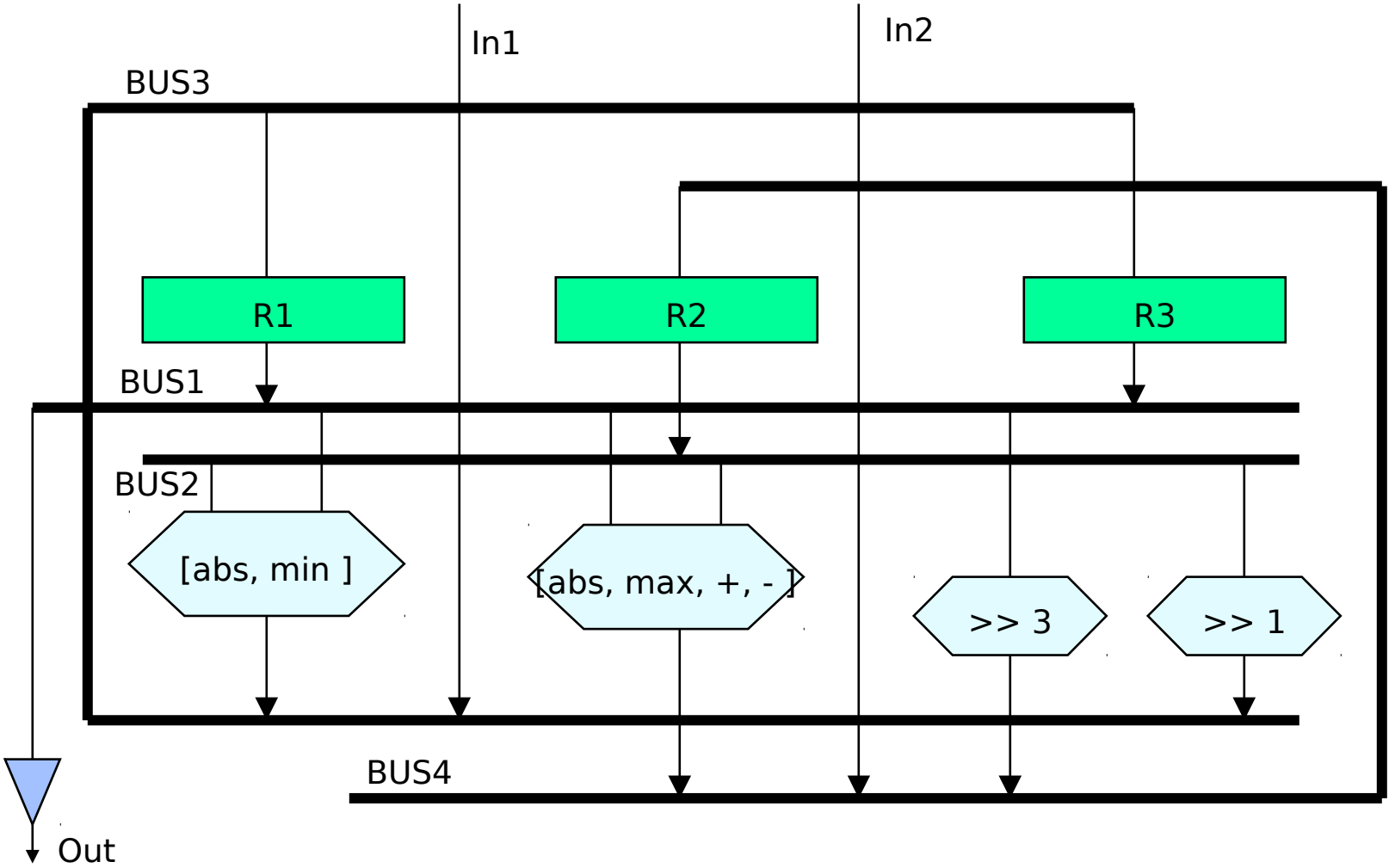
Adicionando mais estágios



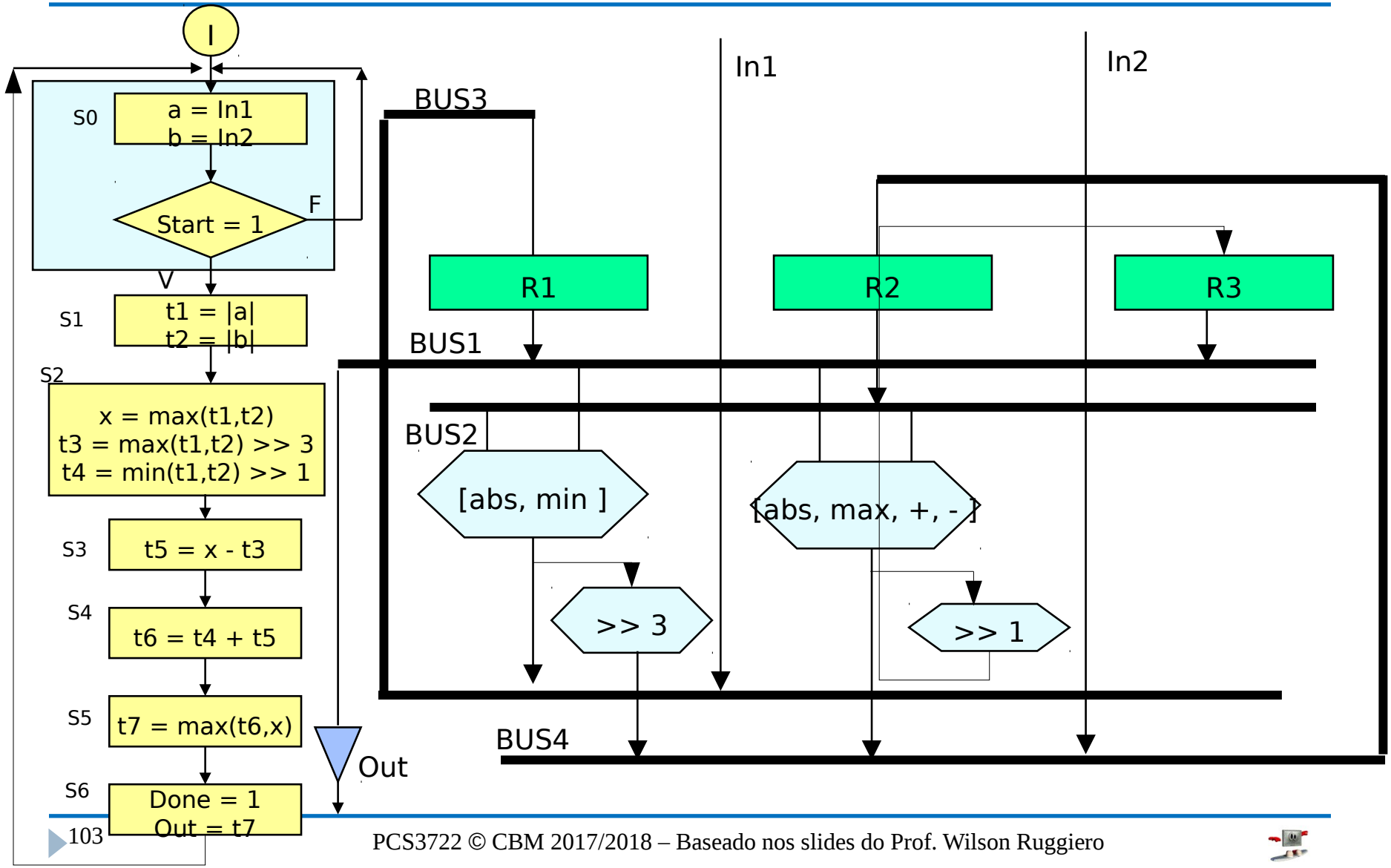
Até onde se pode chegar com a adição de mais estágios em "Pipeline"?
O ganho potencial teórico apresentado por uma organização em "Pipeline" é igual ao número de estágios do sistema!!! VAZÃO = 71,4 Mega amostras por segundo!

Encadeamento de UFs

Fluxo de Dados $\approx\sqrt{a^2+b^2}$

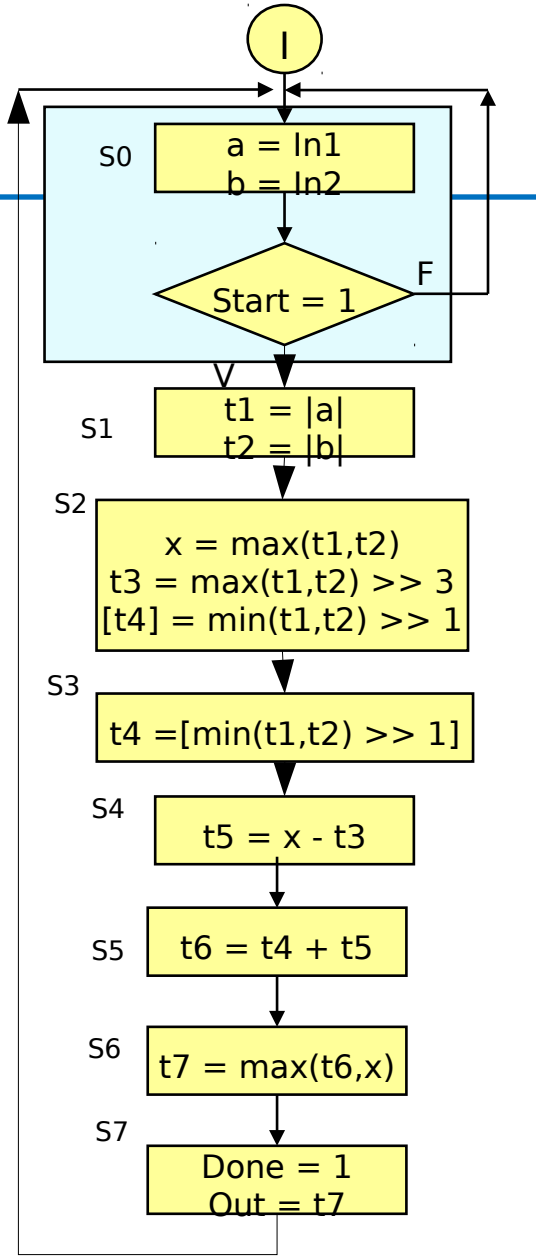


FD com UF “encadeadas”

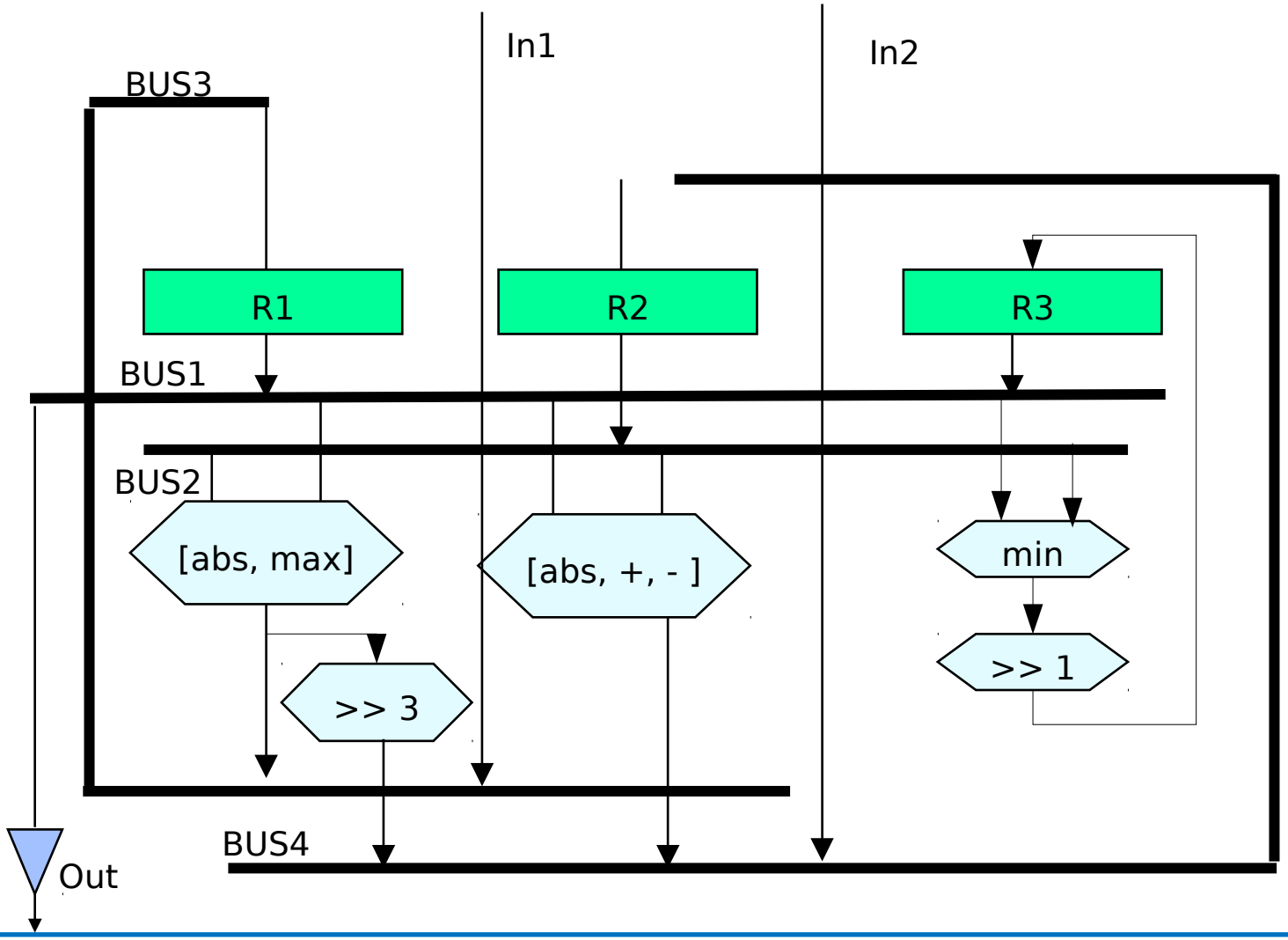


FD Multiciclo (1)

- UF min demora dois ciclos para produzir resultado.
  - Isso é indicado no diagrama ASM pelos [].
  - t4 disponível somente ao final de S3.
- Custo de UF é reduzido.



FD Multiciclo (2)



Escalonamento

Projeto do diagrama ASM (1)

- A definição do algoritmo começa pela organização das operações que devem ser executadas pelo algoritmo sem se preocupar com a definição dos estados que irão compor o ASM.
- Inicialmente deve ser feito um fluxograma ordenando as operações executadas pelo algoritmo.

Projeto do diagrama ASM (2)

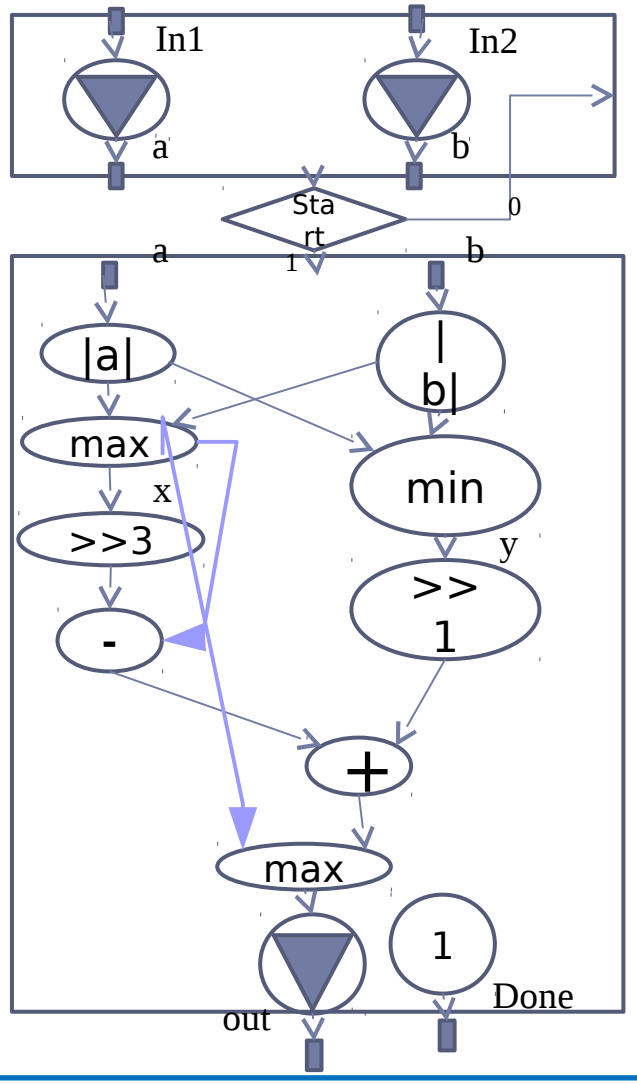
- Considere o algoritmo descrito anteriormente sem a preocupação de se determinar os estados existentes no sistema, mas simplesmente a ordem de execução das operações que compõem o algoritmo em função da disponibilidade dos dados necessários para cada operação.

Fluxograma da "Raiz Quadrada"

a = In1;
b = In2;
t1 = |a|;
t2 = |b|;
x = max(t1,t2);
y = min(t1,t2);
t3 = x>>3;
t4 = y>>1;
t5 = x – t3;
t6 = t4 + t5;
t7 = max(t6,x);
Done = 1;
Out = t7.

Expressa uma possível ordem entre as operações.

Expressa a dependência de dados existente no algoritmo.



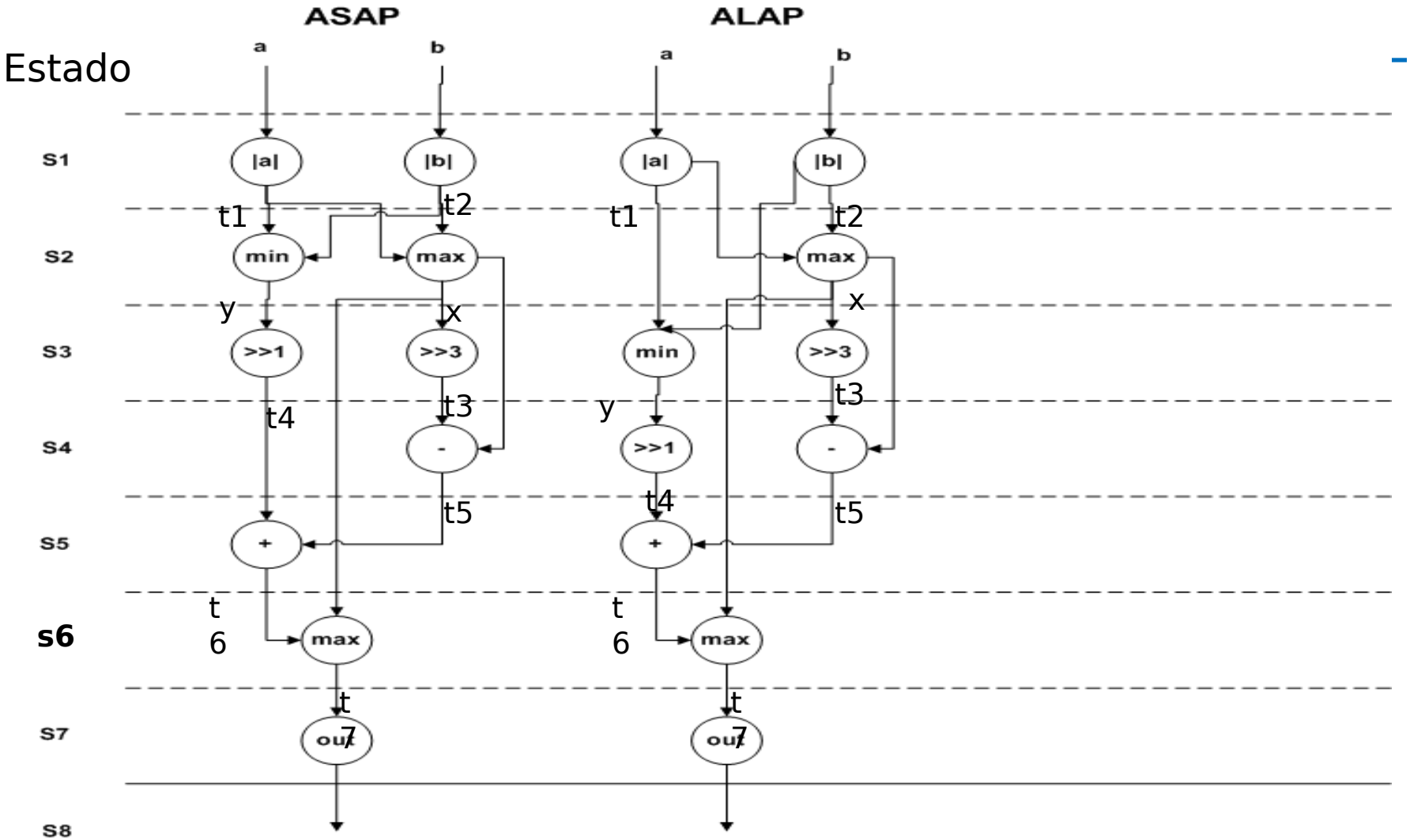
Escalonamento de Operações (1)

- O processo de escalar as operações do algoritmo para serem executadas em estados diferentes da implementação é chamada de “escalonamento de operações”.
- Supondo que se tenha tantos registradores e unidades funcionais quanto necessário, deve-se escalar as operações em função da disponibilidade dos dados, ou seja, executa-se uma operação assim que os seus operandos estiverem disponíveis = Escalonamento **ASAP** (*As Soon As Possible*) → **s**intetiza o algoritmo no menor número de estados permitido face as dependências existentes.

Escalonamento de Operações (2)

- No outro extremo da escala de tempo, devemos escalar as operações no momento em que os resultados que elas produzem são necessários para a sequência do algoritmo = escalonamento ALAP (*As Last As Possible*).
- Estes dois escalonamento são feitos inicialmente pois servem de referência para o métodos utilizados no processo de determinação do número de estados que devem existir e como as operações são alocadas nesses estados.
- Esses dois escalonamentos estão mostrados a seguir para o algoritmo da “Raiz Quadrada”.

Escalonamento de Operações



ASAP: As soon as Possible ALAP: As Last as Possible

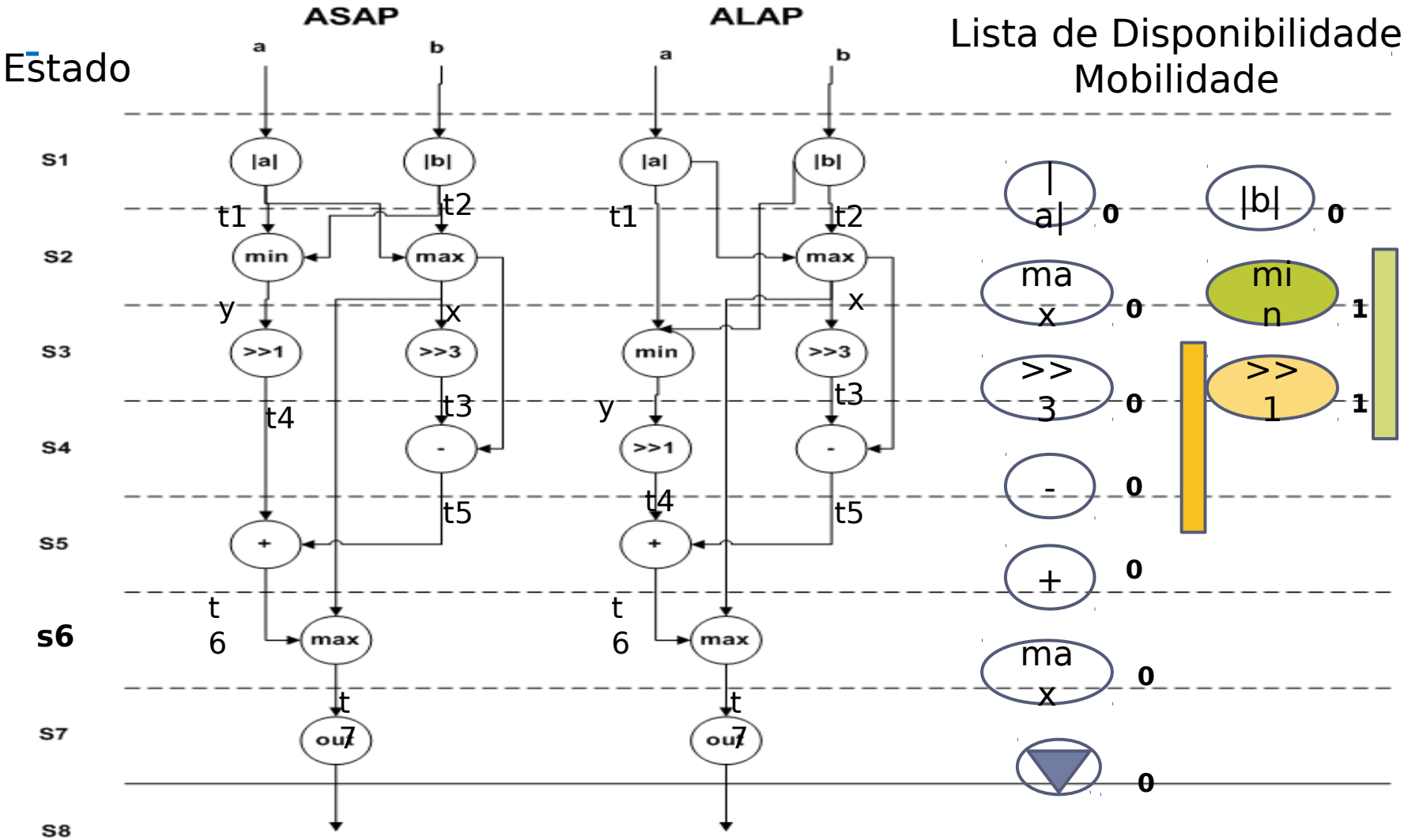
Mobilidade das unidades funcionais

- Se uma UF for alocada no escalonamento ASAP no estado s_k e a mesma unidade for alocada no escalonamento ALAP no estado s_j, a diferença $(s_k - s_j)$ é chamada de mobilidade dessa UF, ou seja,

  $$m(UF) = (s_k - s_j).$$

- No escalonamento feito para o algoritmo da raiz quadrada obteve-se mobilidade igual a 0 (zero) para todas as unidade funcionais menos para a operação de mínimo e a de deslocamento à direita de 1 bit, que ficaram com mobilidade igual a 1.

Escalonamento de Operações



ASAP: As soon as Possible ALAP: As Last as Possible

Escalonamento com Restrições

- Existem dois tipos de restrições que costumam ser consideradas:
  - Restrição de Recursos de Hardware: registradores e unidades funcionais, ou
  - Restrições Temporais: número de estados.
- É possível se aplicar os dois tipos de restrições de maneira combinada ou alternada.
- Vamos ilustrar estes método começando com um exemplo que utiliza restrição de recursos.

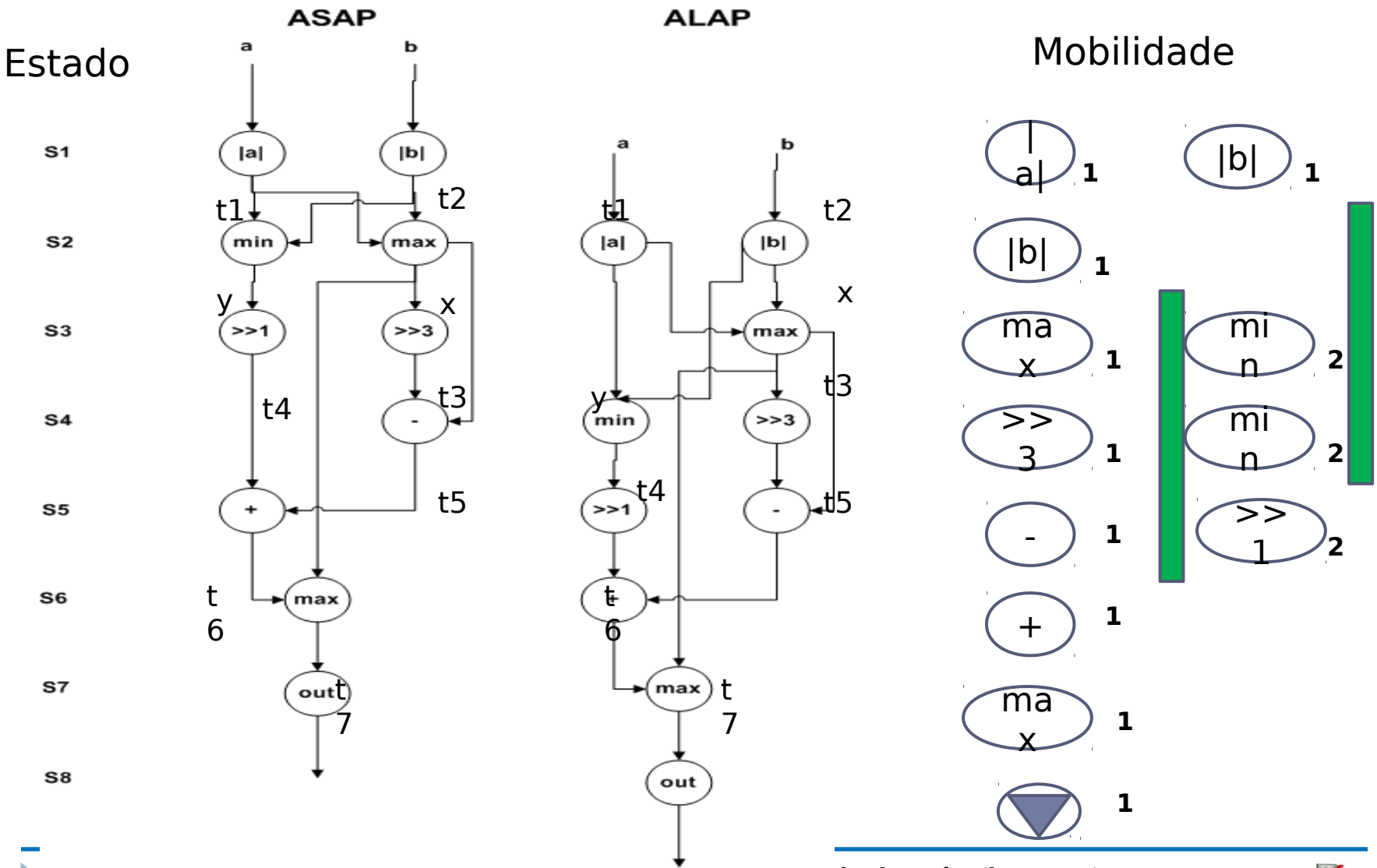
Ex.: Restrição de Recursos (1)

- Restrição no número e tipo de UFs (unidades funcionais).
- Escalonar o algoritmo da raiz quadrada para operar num circuito que utiliza as seguintes UFs:
  - UF1 = { | |};
  - UF2 = {max, min, +, - };
  - UF2 = { >>1};
  - UF3 = {>>3};

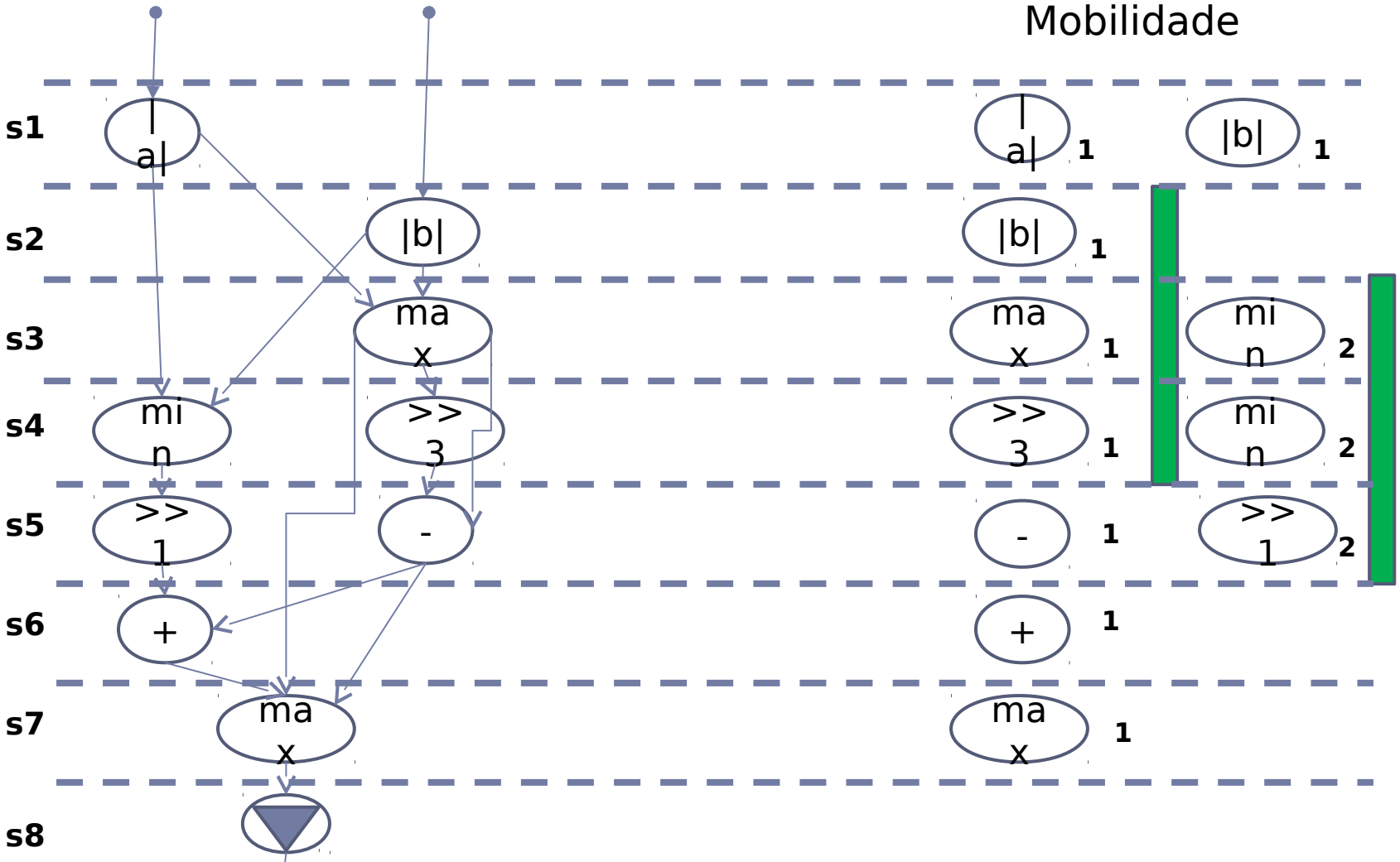
Ex.: Restrição de Recursos (2)

- Determine o menor número de unidades funcionais necessárias no sistema para implementar esse algoritmo com 8 estados no sistema.
- Inicialmente devemos fazer os escalonamentos ASAP e ALAP, e determinar as mobilidades das unidades funcionais para uma implementação com 8 estados.

Escalonamento: Mobilidade



Escalonamento com Restrições



Escalonamento de Recursos (1)

- Objetivo: determinar o maior número de UFs (operações) dentro de um mesmo estado, sujeito as limitações dos recursos de hardware disponíveis.
- Quando há mais de uma operação para ser alocada num dado estado, utilizamos uma métrica de prioridade para decidir qual delas deve ser alocada ao estado (sem violar as restrições impostas pelos requisitos).

Escalonamento de Recursos (2)

- Neste exemplo, nós obtivemos uma alocação que utiliza um estado a mais que no escalonamento ASAP ou ALAP, porém com um custo menor devido a restrição imposta ao uso das unidade funcionais.

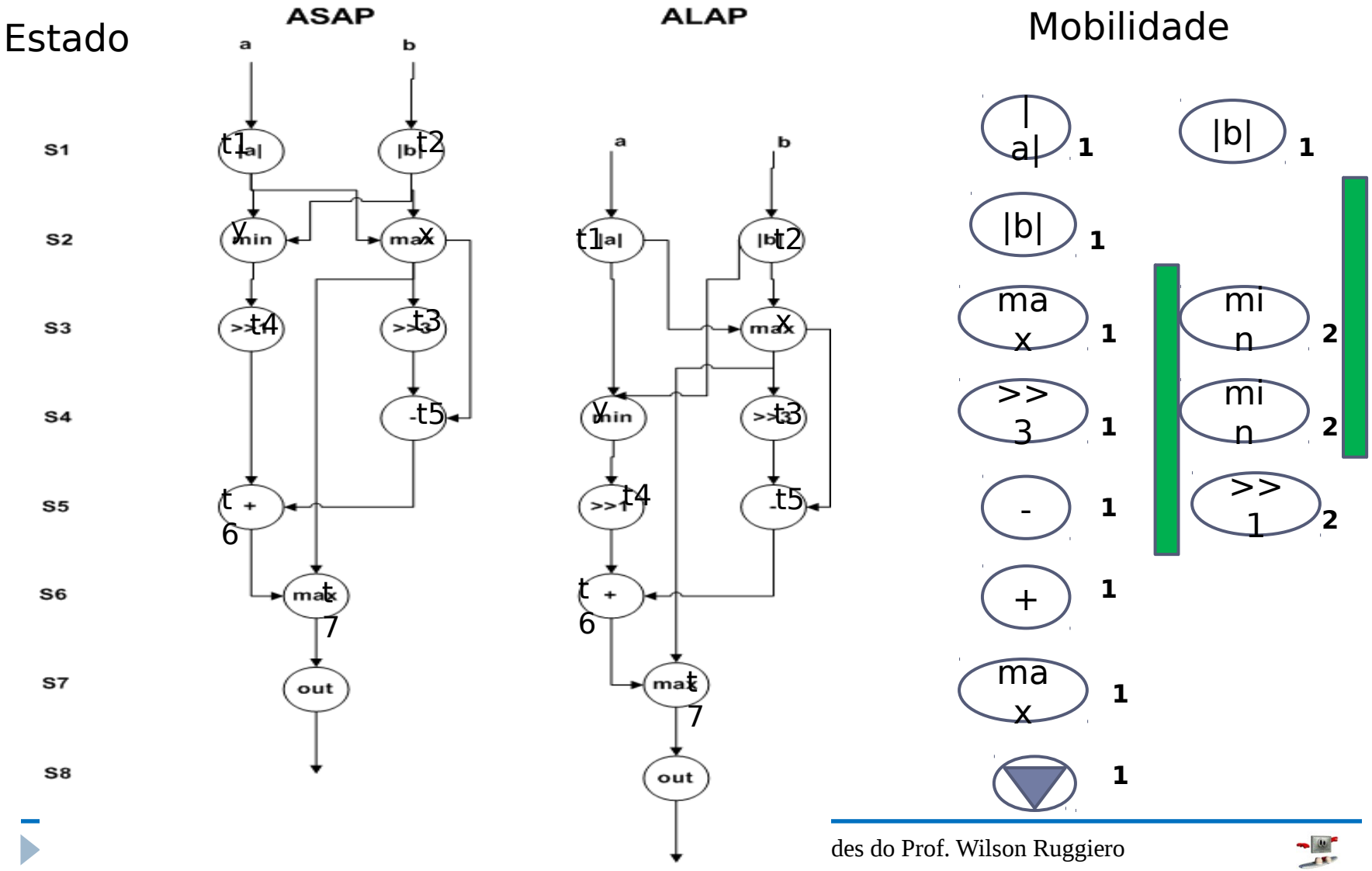
Escalonamento Temporal (1)

- Na maioria das vezes, objetiva-se otimizar o desempenho e não somente o custo de um projeto.
- Para estes casos, define-se o escalonamento Temporal.
  - Visa a minimização do número de estados ao mesmo tempo em que procura reduzir o custo do sistema resultante.

Escalonamento Temporal (2)

- Este objetivo múltiplo é conseguido construindo-se uma “Tabela de Distribuição de Probabilidades”.
- Baseado nessa Tabela alocam-se as operações, uma de cada vez, aos estado considerando-se o critério de escolha onde a maior soma de probabilidades para uma operação e um estado tenha o maior decréscimo.
- Vejamos como este método funciona:

Passo Inicial: ASAP e ALAP



Probabilidades de alocação (1)

- Calculando as mobilidades para este caso temos que todos as operações terão Mobilidade 1 exceto UFs mínimo e deslocamento à direta de 1 bit que terão mobilidade 2.
- UFs com mobilidade 1 poderão ser alocadas em 2 diferentes estado e como essa distribuição é uniforme a probabilidade delas serem alocadas em cada um desses estados é 0,5.

Probabilidades de alocação (2)

- As operações de mínimo e deslocamento à direita de 1 bit possuem mobilidade 2, portanto a probabilidade delas serem alocadas em um dos estado possíveis será 0,33.
- Com estas informações constrói-se a Tabela de Distribuição das Probabilidades para cada estado:

Tabela de Distribuição das Probabilidades – 2

Estados	Unidades Funcionais Aritméticas	Soma das Probabilidades	Unidades Funcionais de Deslocamento	Soma das Probabilidades
s1	|a| |b|	1.0		
s2	max min	1,83		
s3		0.83	>>3	0.83
s4	-	0.83	>>1	0.83
s5	+	1.0		0.33
s6	max	1.0		
s7		0.5		
s8	out			

Tabela de Distribuição de Probabilidades - 2

Estados	Unidades Funcionais Aritméticas	Soma das Probabilidades	Unidades Funcionais de Deslocamento	Soma das Probabilidades
s1	|a| |b|	1.0		
s2	max min	1,83		
s3		0.83	>>1 >>3	0.33
s4		0.33		1.33
s5	-	1.0		0.33
s6	+	1.0		
s7	max	1.0		
s8	out	1.0		

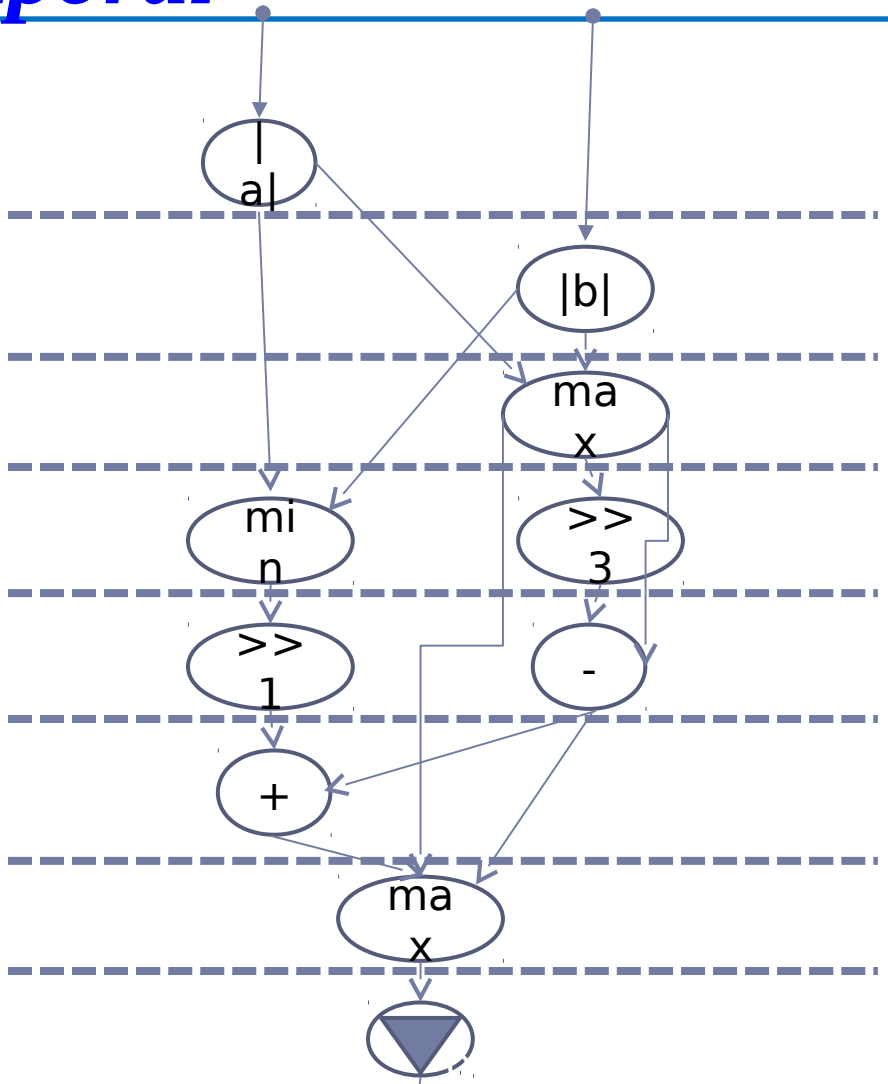
Tabela de Distribuição de Probabilidades - 3

Estados	Unidades Funcionais Aritméticas	Soma das Probabilidades	Unidades Funcionais de Deslocamento	Soma das Probabilidades
s1	|a| |b|	1.0		
s2		1,0		
s3	max	1.0		
s4	min	1.0	>>3	1.0
s5	-	1.0	>>1	1.0
s6	+	1.0		
s7	max	1.0		
s8	out	1.0		

Tabela de Distribuição de Probabilidades - 4

Estados	Unidades Funcionais Aritméticas	Soma das Probabilidades	Unidades Funcionais de Deslocamento	Soma das Probabilidades
s1	**|a|**	**1.0**		
s2	**|b|**	**1,0**		
s3	**max**	**1.0**		
s4	**min**	**1.0**	**>>3**	**1.0**
s5	**-**	**1.0**	**>>1**	**1.0**
s6	**+**	**1.0**		
s7	**max**	**1.0**		
s8	**out**	**1.0**		

ASM resultante da alocação com restrição temporal



Por coincidência chega-se à mesma alocação que no caso da alocação com restrição de Hardware.

Perguntas e/ou Comentários?

- Tarefas:
  - Exercícios disponíveis no Ae.
  - Leitura do capítulo 8 - Registrer-Transfer Design, do livro "Principles of Digital Design", Daniel Gajski, Prentice Hall, 1997.
  - Assistir vídeos referentes ao capítulo 6 - Hardware Synthesis do livro “Embedded System Design: Modeling, Synthesis, Verification”.
    - Specification and Architecture: http://www.cecs.uci.edu/esd/viewESDVideo.htm?id=25
    - High-level Synthesis: http://www.cecs.uci.edu/esd/viewESDVideo.htm?id=26