GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Politécnica UFRJ

COPPE UFRJ

Circuitos Lógicos

Aula 16

cruz@gta.ufrj.br http://gta.ufrj.br/~cruz

Poli – Dept. de Eng. Eletrônica e de Computação
Universidade Federal do Rio de Janeiro

Na última aula

- Comentários sobre os trabalhos
- Máquinas de estado
  - Projeto
    - Concepção
    - Simplificação
  - Implementação

Hoje

- Unidade Lógico-Aritmética
  - ULA
- Lógica Programável
  - PLA
  - LUT
  - FPGA
- Oscilador
  - Gerador de clock
- Paridade

Projeto de circuitos sequenciais

Etapas de projeto

- Definição de estados
  - Construção do diagrama de estados e tabela de transição de estados
  - Redução do número de estados
- Escolha da codificação dos estados
- Escolha do tipo de FF a ser usado
- Projeto da lógica combinacional
  - Projeto dos circuitos de excitação
  - Projeto dos circuitos de saída

Lista de exercícios

- Exercícios com menos variáveis

Unidade Lógica e Aritmética

Unidade Lógica e Aritmética - ULA

- Entradas
  - Operando A
  - Operando B
  - Operação F
  - Status (opcional)
- Saídas
  - Resultado R
  - Status D

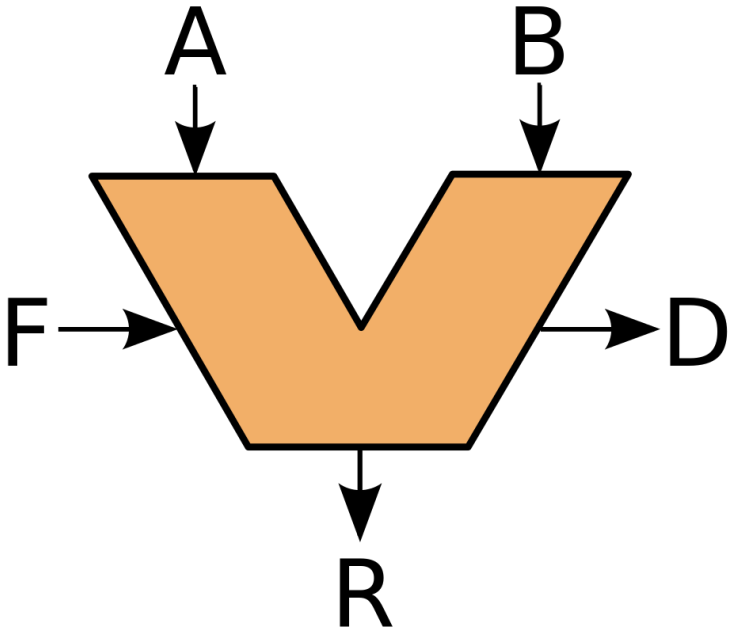


Imagem retirada da Wikipedia

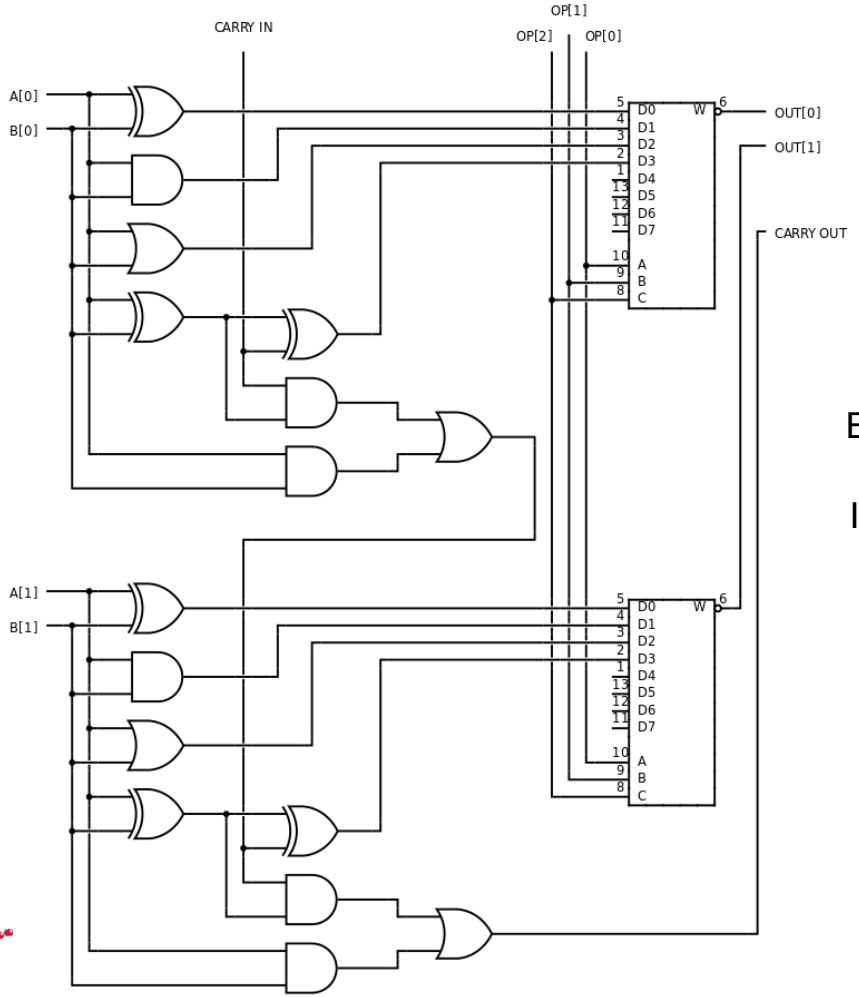
Unidade Lógica e Aritmética - ULA

- Diversos circuitos internos
  - Aritméticos
    - Soma
    - Subtração
    - Multiplicação
    - Divisão
  - Portas lógicas
    - OR
    - AND
    - XOR
- Saída selecionada por um multiplexador
  - Bits de operação controlam qual circuito está conectado à saída

ULA – Exemplo

- Operações
  - Soma
  - OR
  - AND
  - XOR
- Saída selecionada por um multiplexador

OP[2]	OP[1]	OP[0]	Operação
0	0	0	$A \oplus B$
0	0	1	$A \& B$
0	1	1	$A \mid B$
0	1	0	$A + B$

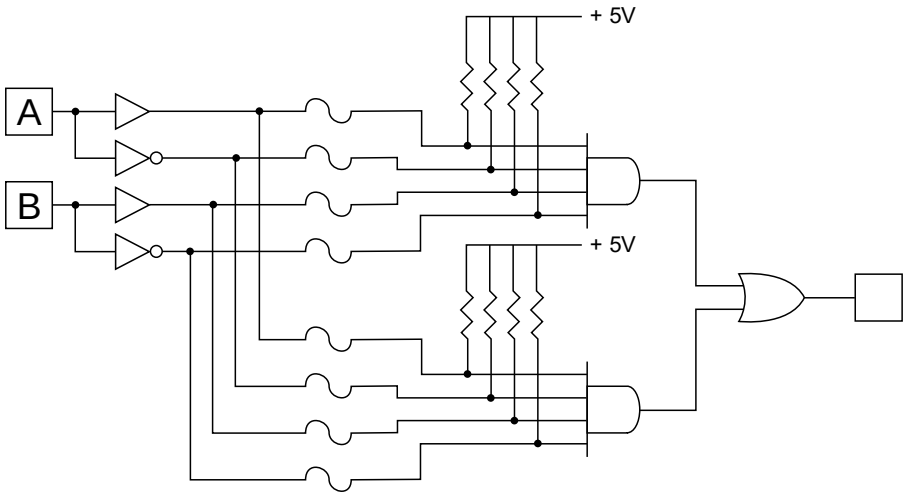


Exemplo de ULA de 2 bits. Imagem retirada da Wikipedia

Lógica programável

Programmable Array Logic*

- Entradas e seus complementos vão para diversas portas AND
- Saídas das portas AND são agregadas por porta OR
- Fusíveis são ‘queimados’ durante a programação
  - Entradas são combinadas em seus mintermos
- Tecnologia proprietária



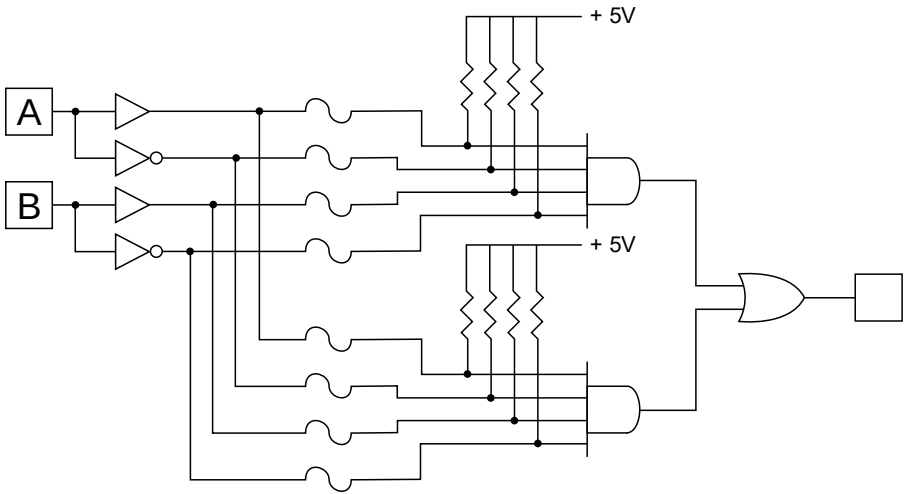
Simplified programmable logic device

*Lógica de Vetor Programável

Imagem retirada da Wikipedia

Programmable Array Logic*

- Vamos fazer um A $\oplus$ B
  - Devemos queimar todos os fusíveis, exceto os que formam os mintermos
    - $A \cdot \bar{B}$
    - $\bar{A} \cdot B$



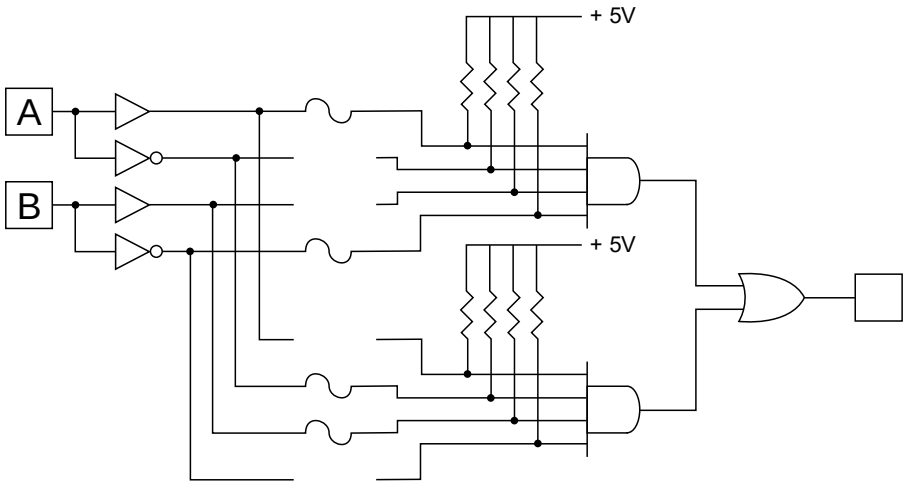
Simplified programmable logic device

*Lógica de Vetor Programável

Imagem retirada da Wikipedia

Programmable Array Logic*

- Vamos fazer um A $\oplus$ B
  - Devemos queimar todos os fusíveis, exceto os que formam os mintermos
    - $A \cdot \bar{B}$
    - $\bar{A} \cdot B$



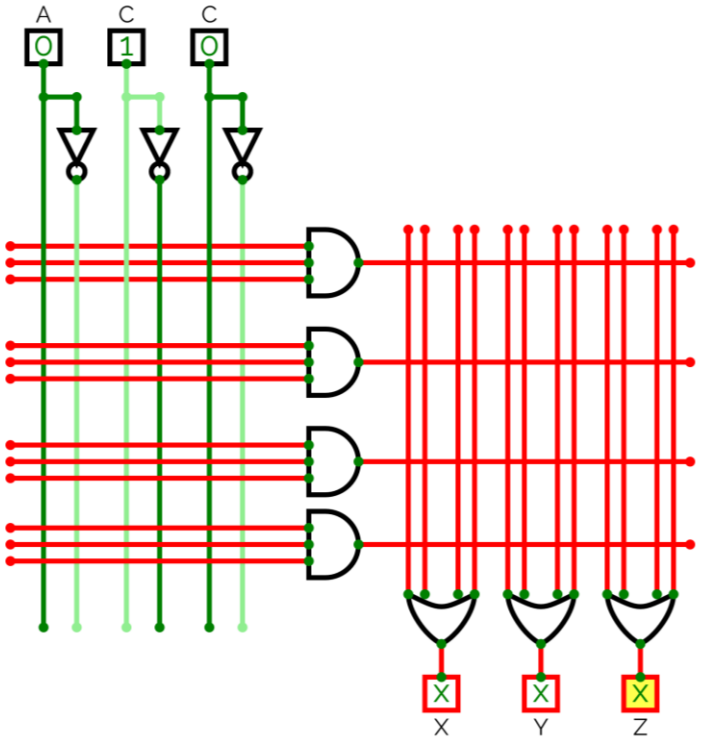
Simplified programmable logic device

*Lógica de Vetor Programável

Imagem retirada da Wikipedia

Programmable Logic Array*

- Matriz de ANDs
  - Entradas chaveadas até portas AND
- Matriz de OR
  - Saídas da matriz de ANDs chaveadas até portas OR
- Múltiplas saídas



*Vetor de Lógica Programável

Programmable Logic Array*

- Matriz de ANDs
  - Entradas chaveadas até portas AND
- Matriz de OR
  - Saídas da matriz de ANDs chaveadas até portas OR
- Múltiplas saídas

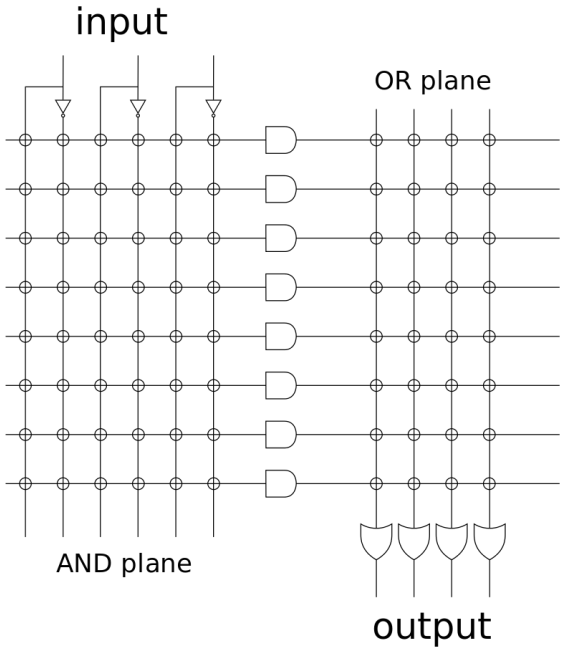


Imagem retirada da Wikipedia

*Vetor de Lógica Programável

Programmable Logic Array*

- Complicado de programar
  - Necessita “queimar” ligações
  - Feito selecionando cada cruzamento a ser queimado e aplicando uma tensão “alta”

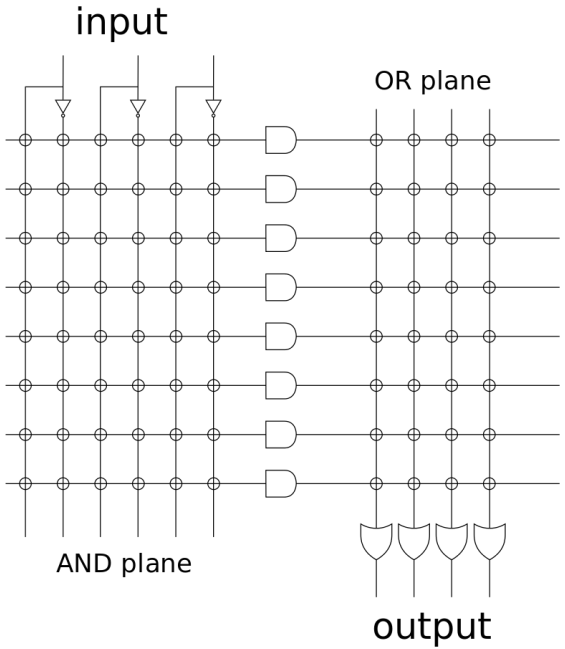
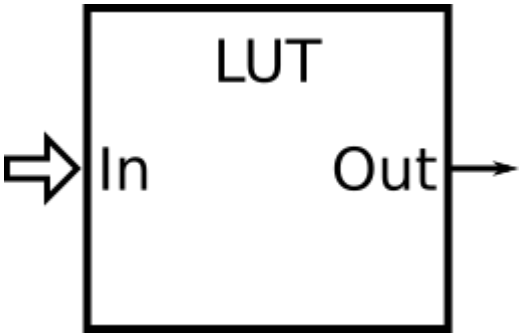


Imagem retirada da Wikipedia

*Vetor de Lógica Programável

Look-up Table (LUT)

- Conjunto de entradas e uma saída
- Saída programada para cada configuração de entrada
  - Função lógica é implementada enquanto sua tabela verdade
- Implementada normalmente com memória



In1	In0	Out
0	0	1
0	1	1
1	0	0
1	1	1

*Tabela de consulta (tradução livre)

Field-Programmable Gate Array – FPGA

- Módulos
  - ULAs
  - LUTs
  - Mux
  - Demux
  - Memórias
- Matrixes de conexão entre módulos
- Programado com HDL ou VHDL
  - Carga das LUTs
  - Interconexão entre módulos

Vetor de Portas Lógica Programável em Campo*

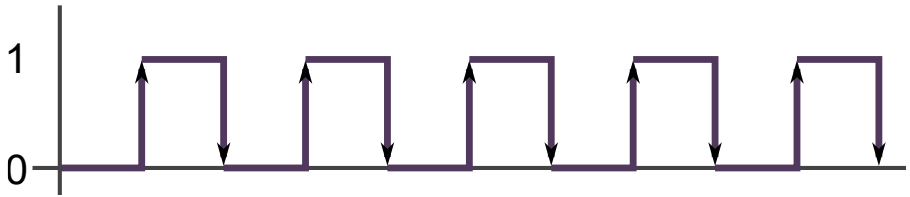
VHDL

- **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage
  - Padronizada pelo IEEE
  - Foca em descrever um hardware
  - Compilação gera um circuito
    - Definir conexões entre módulos da FPGA correspondente
    - Carregar informações nas LUTs

Temporização

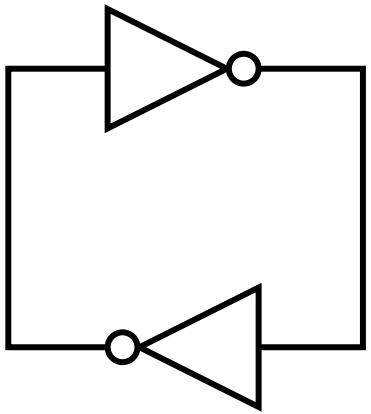
Gerador de clock

- Oscilador
- Valores sucessivos de 0s e 1s
- Periodização
  - Período t
  - Frequência f
  - f = 1/t
- Duty cycle de 50%
    - Tempo em 1 = tempo em 0



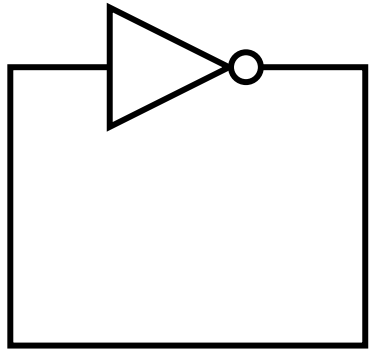
Realimentação positiva

- Ciclo de portas
  - Reforçam o valor uma das outras
    - Pode ter uma ou mais portas



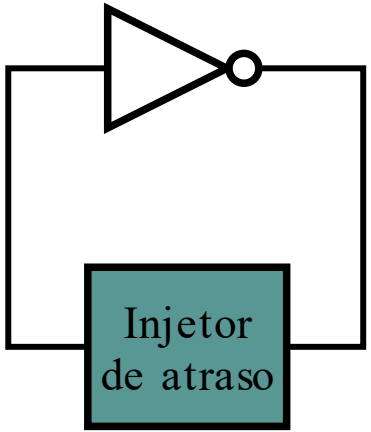
Realimentação negativa

- Ciclo de portas
  - Invertem o valor uma das outras
    - Pode ter uma ou mais portas
  - Resultado depende das condições elétricas da porta
    - Estabilidade em 0,5 Vcc
    - Estabilidade em 0
    - Estabilidade em 1



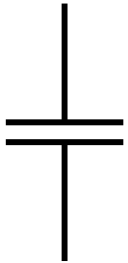
Realimentação negativa com atraso

- Saída assume um valor
- Entrada recebe o valor com um atraso
- Saída assume valor inverso
- Entrada recebe novo valor com um atraso
- Saída assume valor inverso
- Entrada recebe o valor com um atraso
- .....



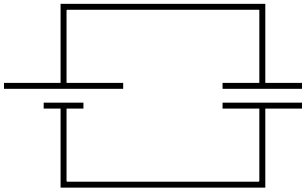
O capacitor

- Duas placas condutoras separadas por um isolante
- Armazena energia na forma de campo elétrico
  - Carga
- Funciona como uma pilha recarregável muito simples
  - Descarrega rápido
  - Carrega rápido



O capacitor

- Carrega quando ligado em uma fonte
  - Rápido
- Mantém a tensão quando a fonte vai embora
  - Por um curto período de tempo
- Descarrega “sozinho”
  - Na DRAM comercial, demora 1/16s

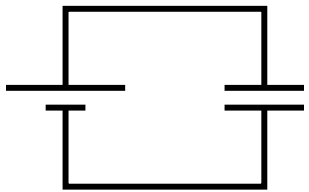


Capacitância

- Propriedade do capacitor de acumular carga
  - Carga armazenada de acordo com a tensão

$$C = \frac{q}{V}$$

- Medida em Farads



Resistência, tensão e corrente

- Tensão é a energia potencial elétrica entre dois pontos [Volts]
  - Tendência de elétrons irem de um ponto a outro
  - Energia por carga

$$V = \frac{Energia}{q}$$

- Corrente [Ampère]
  - Quantidade de carga que passa em um ponto, por tempo

$$i = \frac{q}{t}$$

- Resistência [Ohm]
  - Oposição à passagem de corrente por um corpo

$$V = Ri$$

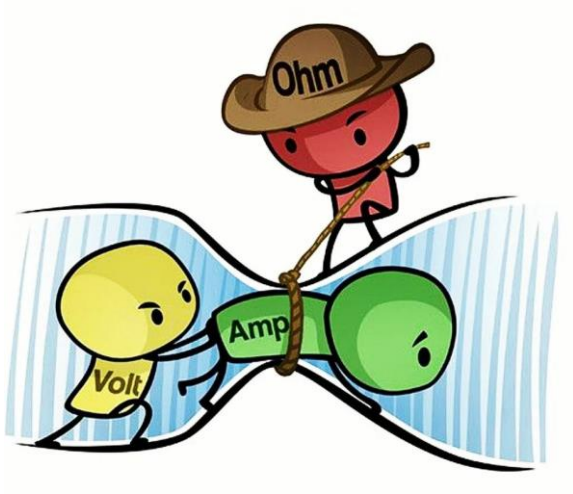
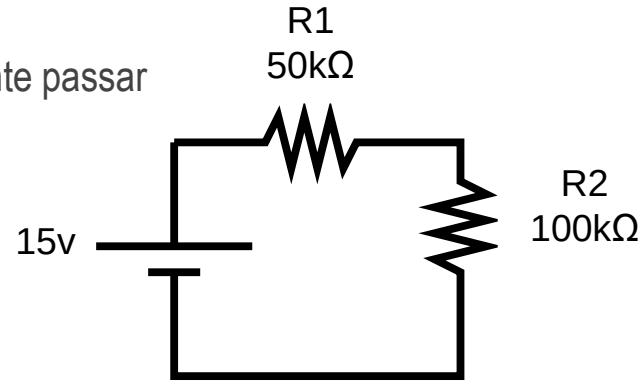


Imagem retirada de r/DamnUEngineering

Resistência, tensão e corrente

- Carga vai perdendo energia para completar um caminho
  - A cada resistência, perda de energia
- Tensão é maior entre dois pontos onde é “difícil” pra corrente passar
  - Gasto maior de energia
- Tensão é zero quando não há resistência
- Se não há corrente
  - Tensão é máxima sobre a parte aberta do circuito
  - Tensão é zero em todo o resto

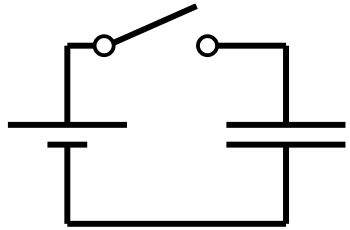


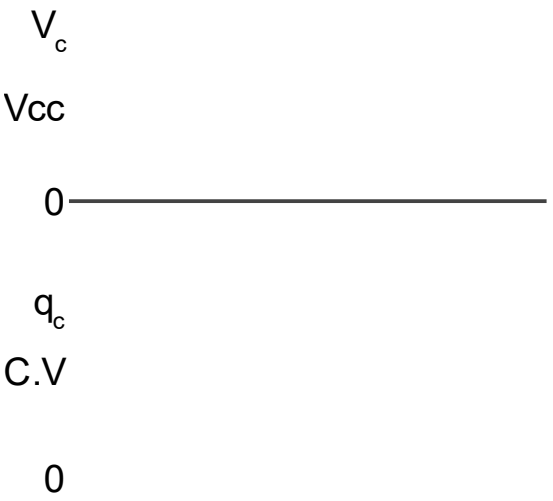
Capacitância

- Chave fecha
- Toda a tensão cai sobre o capacitor
  - Capacitor está “livre” e age como um condutor de resistência zero
- Capacitor armazena *imediatamente* toda a carga possível

$$q = C \cdot V$$

  - Capacitor fica “cheio” e age como isolante



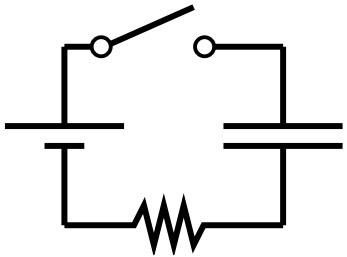


Circuito RC série

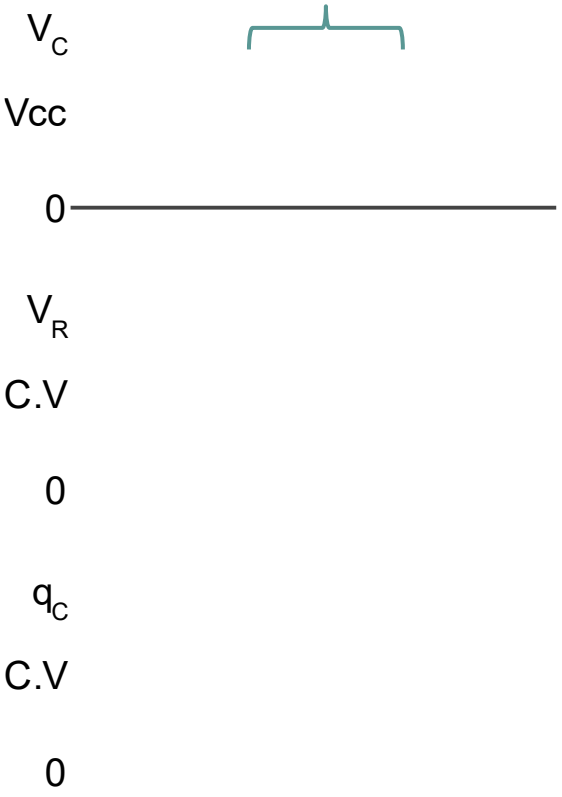
- Chave fecha
- Resistor limita a quantidade de carga que passa por ele

$$i = \frac{q}{t}$$

- Capacitor demora um tempo para carregar
  - Tempo proporcional à capacitância e à resistência

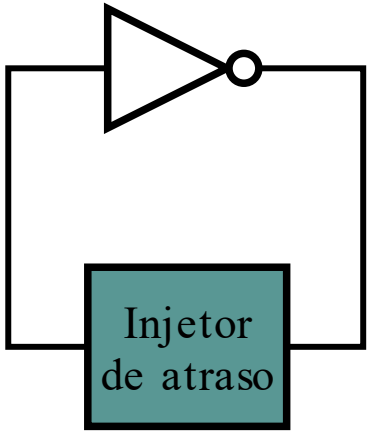


Tempo de carga pode ser usado como atraso!



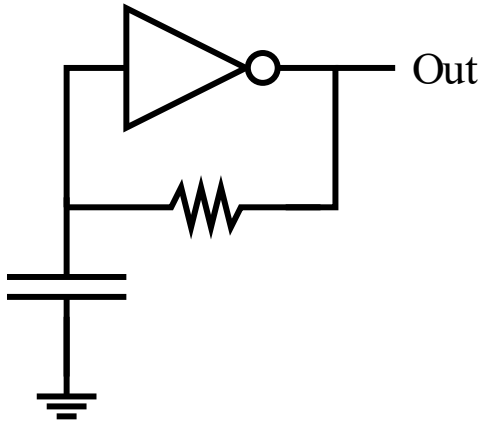
Realimentação negativa com atraso

- Usar circuito RC para atrasar a realimentação



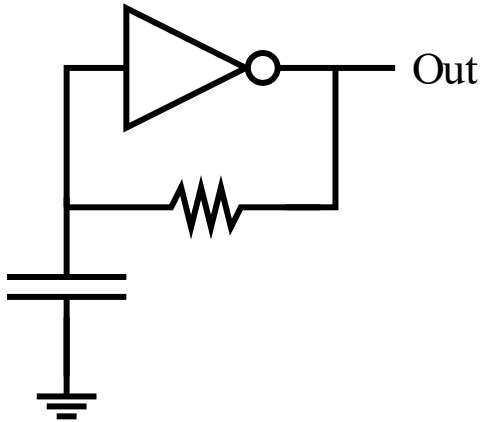
Realimentação negativa com atraso

- Usar circuito RC para atrasar a realimentação
- Tensão da saída vai ser “atrasada” antes de chegar na entrada



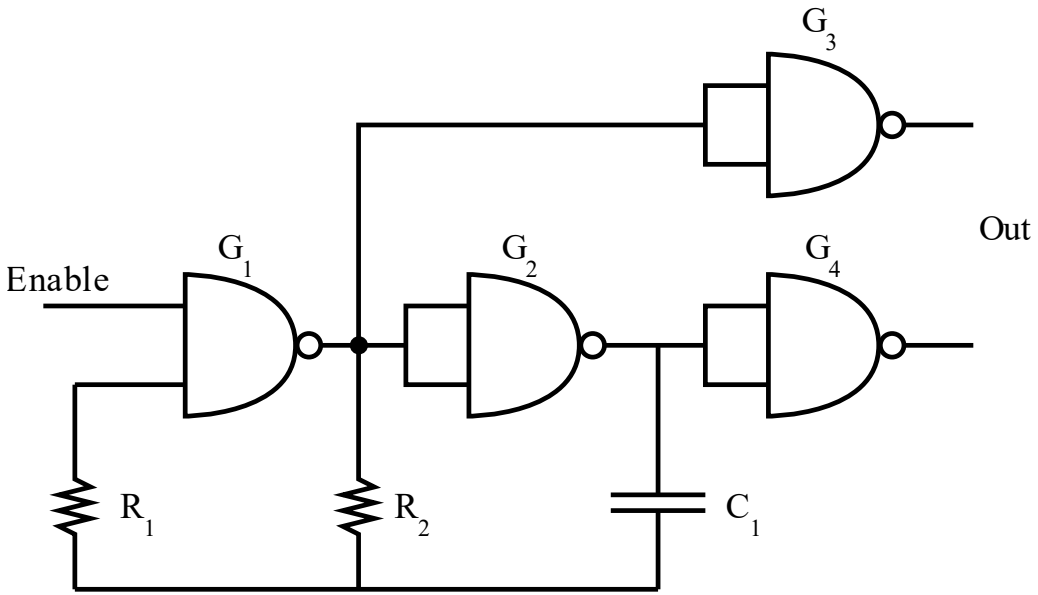
Realimentação negativa com atraso

- Usar circuito RC para atrasar a realimentação
- Tensão da saída vai ser “atrasada” antes de chegar na entrada
- Essa configuração não funciona tão bem



Realimentação negativa com atraso

- G_3 e G_4 apenas server de buffer
- G_1 e G_2 sempre em estados inversos
- C_1 carrega com a corrente das saídas de G_1 e G_2
- R_2 limita a corrente de carga
- $Período = 2{,}2\ R_2C_1$



Disclaimer

- Nos clocks são utilizados osciladores de cristal
- Muito mais precisos
- Muito mais estáveis ao ambiente
  - Temperatura
  - Pressão
  - Variações de tensão

Laboratório

- **Montar** um circuito para fazer um oscilador
  - Aprender o que é um osciloscópio

Circuito RC série

- “Mas professor, o que acontece entre o início e o fim da carga do capacitor??

$$V_C(t) = V(1 - e^{-\frac{t}{RC}})$$

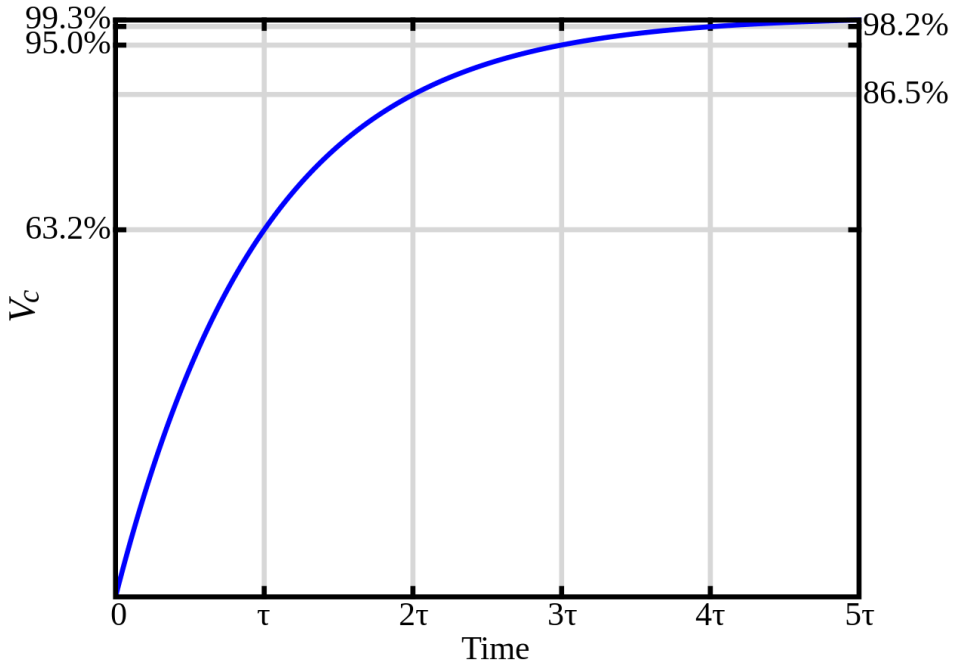


Imagem retirada da wikipedia

$$V_R(t) = Ve^{-\frac{t}{RC}}$$

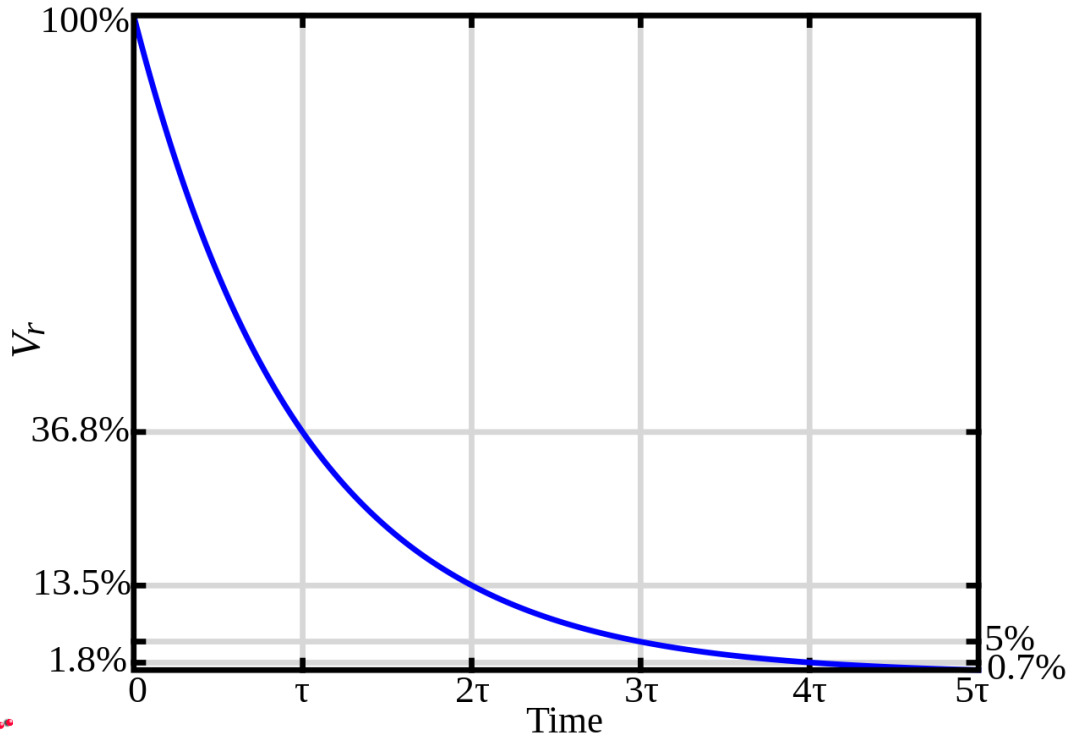
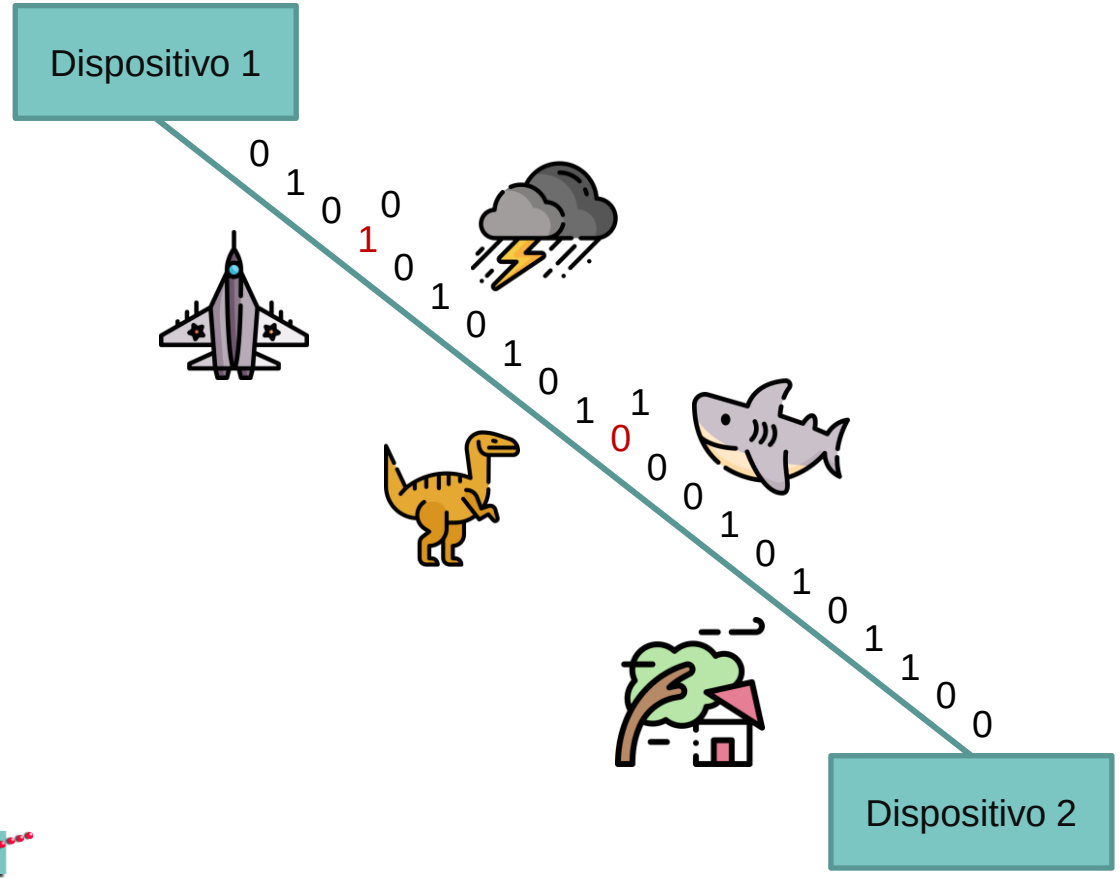


Imagem retirada da wikipedia

Erros

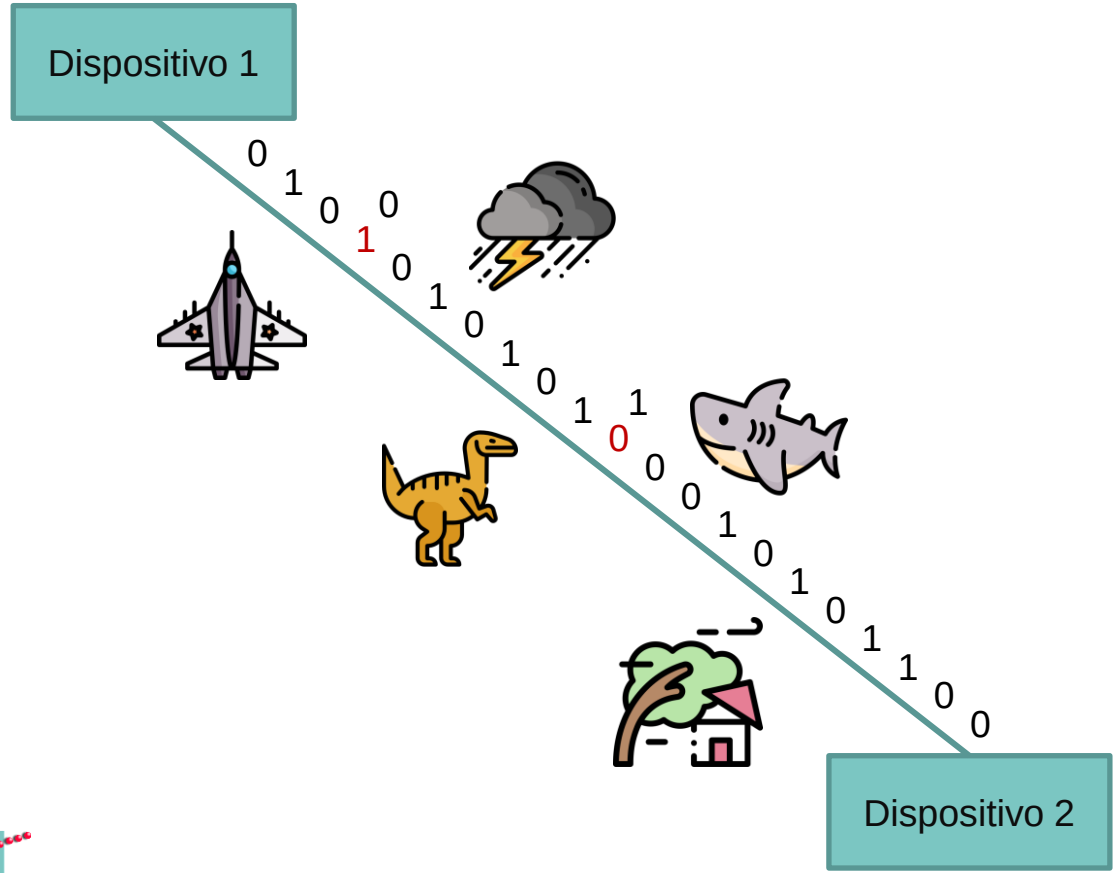
Cenário

- Bits são enviados entre dispositivos digitais
- Problemas podem acontecer
  - Pulso eletromagnético
  - Mau contato
  - ...
- Bits podem ser invertidos
  - O que era 1 vira 0
  - O que era 0 vira 1



Ideia

- Inserir redundâncias no envio
  - Informação a mais do que seria enviada originalmente
- Verificar redundância no recebimento
  - Se informação é redundante, ótimo
  - Se houver inconsistência, existe erro



Exemplos de redundância

- Enviar a mesma informação duas vezes independente de erros
  - Pouco eficiente
  - Dobro do custo

- Codificar a informação com mais bits do que o necessário
  - Símbolos inexistentes indicam erro

A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	0	0
1	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

Codificando 3 bits em 4 bits de forma a detectar vários tipos de erros

Paridade de um conjunto de bits

- A paridade do número de 1's da mensagem original
- Paridade ímpar
  - 0 quando o nº de 1's é par
  - 1 quando o nº de 1's é ímpar
- Paridade par
  - 1 quando o nº de 1's é par
  - 0 quando o nº de 1's é ímpar

P	A_2	A_1	A_0
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1

Paridade de cada um dos símbolos

Quase sempre usamos paridade ímpar

Redundância por bit de paridade

- Na transmissão
  - Calculamos a paridade
  - Adicionamos o bit de paridade à mensagem

- Na recepção
  - Calculamos a paridade da mensagem recebida
  - Verificamos se é a mesma indicada pelo bit de paridade

P	A_2	A_1	A_0
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1

Adicionando um bit de paridade

Redundância por bit de paridade

- A inversão de um único bit pode ser detectada!
  - Com um único bit adicionado
- E se houver 2 inversões?
  - E 3?
  - E n?

P	A_2	A_1	A_0
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1

Adicionando um bit de paridade

Modelo geral para detecção de erros

- Transmissor transmite mensagem *m* junto com algum *f(m)*

- Receptor recebe m_r e $f(m)_r$
  - Calcula $f(m_r)$
  - Compara $f(m_r)$ com $f(m)_r$
    - *Se iguais, não detecta erro*
    - *Se diferentes, detecta erro*

$$f(\text{✉}_r) = f(\text{✉})_r$$

Adendo importante

- Detecção de erros
  - Muitas maneiras além da mostrada
    - Repetição
    - Codificação
    - Funções (Paridade, *Checksum*, CRC)

Adendo importante – 2

- Detecção de erros
  - Mensagem é transmitida
    - Há erro na transmissão
  - Receptor é capaz de saber que houve erro
    - Receptor não é capaz de recuperar a mensagem original

- Correção de erros
  - Mensagem é transmitida
    - Há erro na transmissão
  - Receptor é capaz de saber *qual* erro aconteceu
    - Receptor é capaz de recuperar a mensagem original

Circuito de detecção de paridade

Tabela verdade

- Nº de 1s é par
  - P = 0

- Nº de 1s é ímpar
  - P = 1

A_2	A_1	A_0	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Mapa de Karnaugh

- Vizinhos são sempre diferentes!
- Meio óbvio
  - Queremos detectar inversão única de bits

	$\overline{A_0}$		A_0		A_0	
$\overline{A_2}$	0	0	1	1	3	0	2	1
A_2	4	1	5	0	7	1	6	0
	$\overline{A_1}$				A_1			

A_2	A_1	A_0	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Equações

$$P = A_0 \cdot \overline{A_1} \cdot \overline{A_2} + \overline{A_0} \cdot A_1 \cdot \overline{A_2} + \overline{A_0} \cdot \overline{A_1} \cdot A_2 + A_0 \cdot A_1 \cdot A_2$$

- XOR vai resolver nosso problema

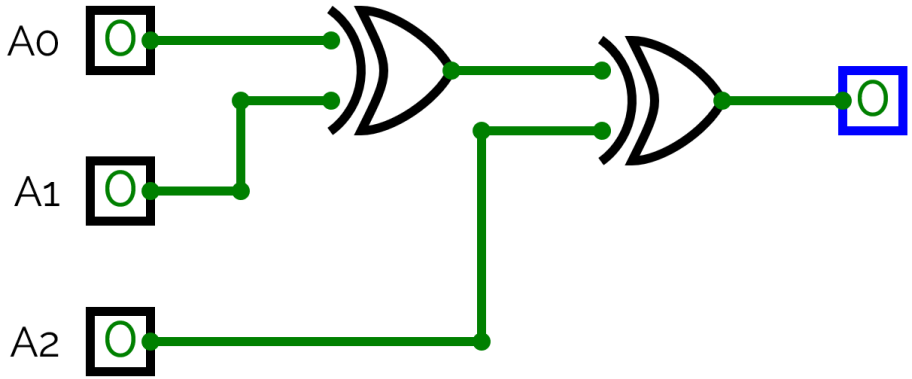
$$P = (A_0 \oplus A_1) \oplus A_2$$

  - Ordem dos parênteses não importam

	$\overline{A_0}$		A_0		$\overline{A_0}$	
$\overline{A_2}$	0	0	1	1	3	0	2	1
A_2	4	1	5	0	7	1	6	0
	$\overline{A_1}$				A_1			

Circuito

- Extremamente simples



Conclusão

- Comunicação pode ter erros
  - Detectar erros
  - Corrigir erros

- É importante saber contar o tempo
  - Geradores de clock

Próxima aula

- Comparação
- Comunicação com estados

Créditos

Os ícones desta apresentação foram feitos por Freepic e retirados de www.flaticon.com

GTA / UFRJ

GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

www.gta.ufrj.br