

Elementos de Sistemas

Aula 9 – Lógica sequencial

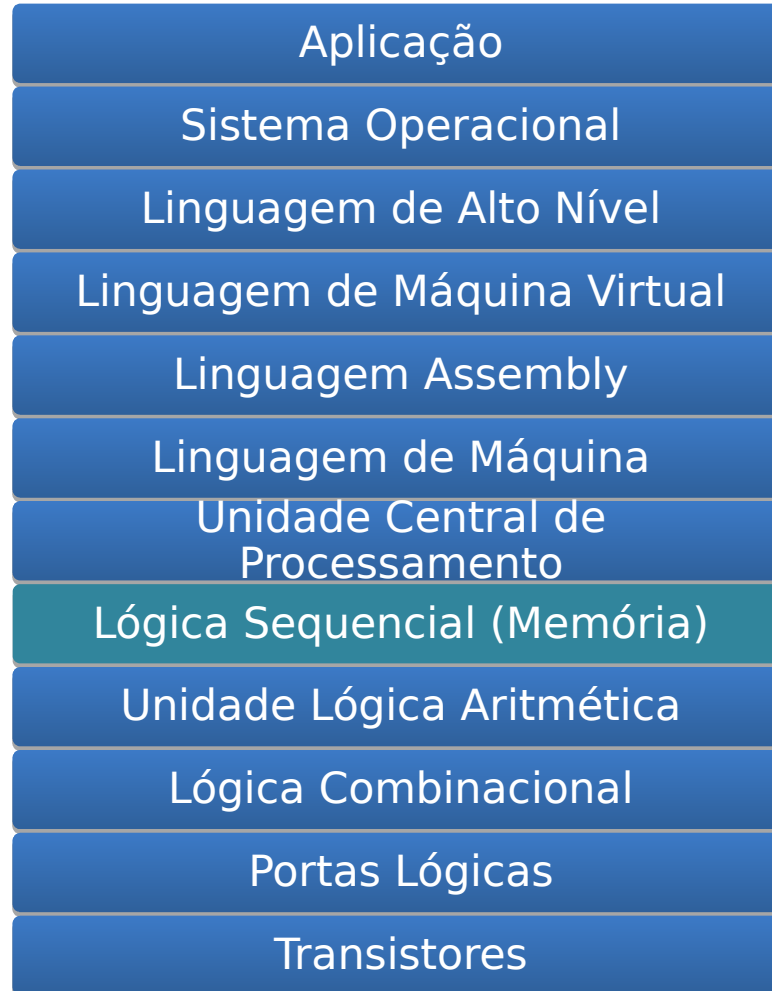
***"Se você diz a verdade, você não precisa lembrar de nada."
"If you tell the truth, you don't have to remember anything."
Mark Twain (1835-1910) escritor americano***

Objetivos de Aprendizizado da Aula

- Criar registradores a partir de flip-flops;
- Visualizar hierarquia de módulos

Conteúdos: Registradores; Registradores de Deslocamento;

Níveis de Abstração



Dúvidas ?

Lembrem-se:

Tragam suas dúvidas anotadas;

Verifiquem sites como:

- Google
- Stack Overflow
- Etc...

Use o Slack, para perguntar para seus colegas,
e o professor antes da aula

Você deveria saber:

- **Com base nos vídeos e na leitura do livro :**
 - Diferença entre lógica combinacional e sequencial
 - O que é um latch e como ele é feito
 - O que é um Flip Flop e como ele opera
 - O que é uma célula de memória (e como ela é feita) – binary bit
 - O que é um registrador e como ele funciona
 - O que é uma memória RAM
 - Program Counter

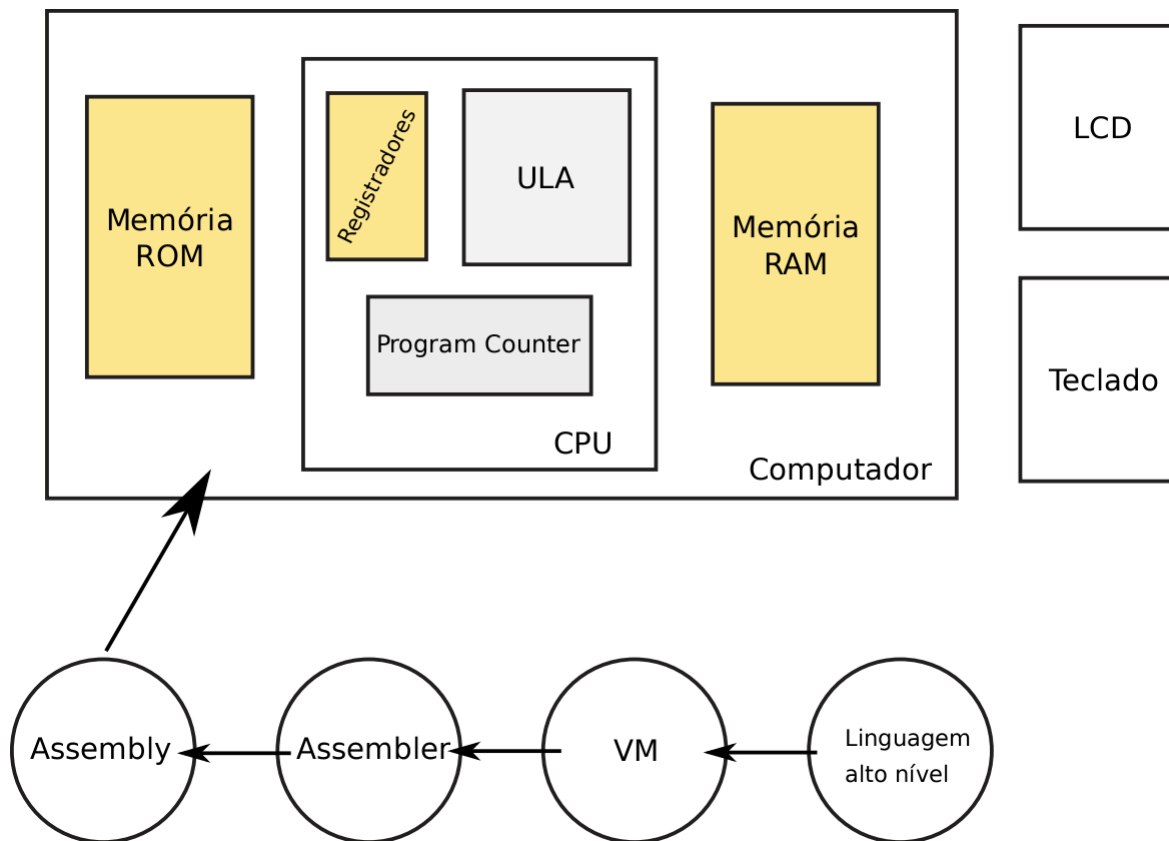
Trabalhando em Projeto

Objetivos no Curso:

(que os alunos sejam capazes de:)

- Trabalhar em equipes auto gerenciáveis;
- Desenvolver em fases “Sprints”;
 - Cada sprint um dos módulos principais.
- Respeitar a agenda do projeto e de reuniões;
 - Acompanhamento:
 - TODO
 - DOING
 - DONE

Projeto E – Chips Sequenciais

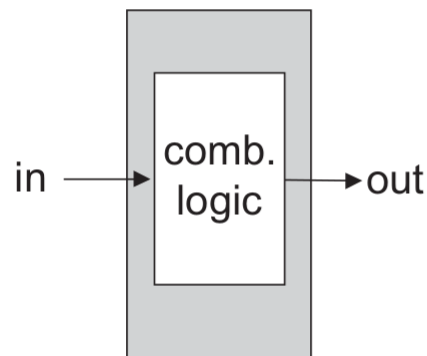


Projeto E – Chips Sequenciais

Desenvolver os diversos chips do projeto.

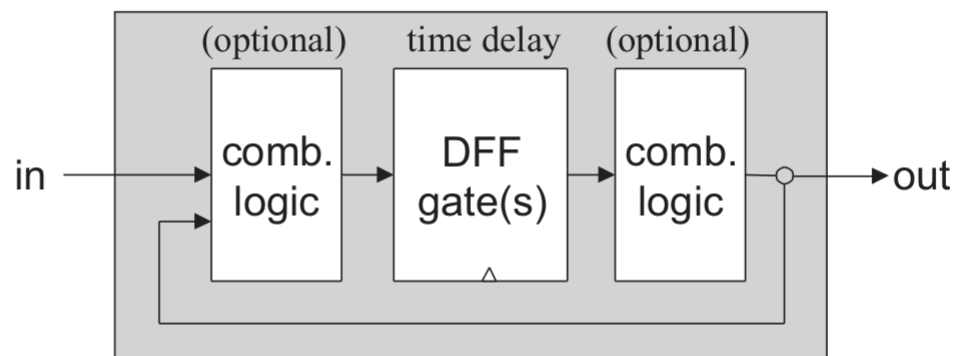
Chapter 3

Combinational chip




$out = \text{some function of } (in)$

Sequential chip



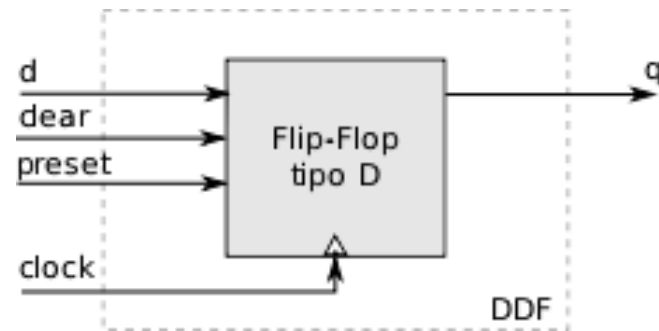
$out(t) = \text{some function of } (in(t-1), out(t-1))$

Figure 3.4 Combinational versus sequential logic (in and out stand for one or more input and output variables). Sequential chips always consist of a layer of DFFs sandwiched between optional combinational logic layers.



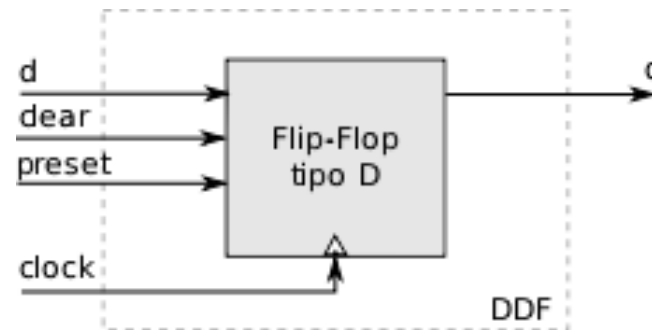
Sequential → Flip Flop

Flip Flop em VHDL



Flip Flop em VHDL

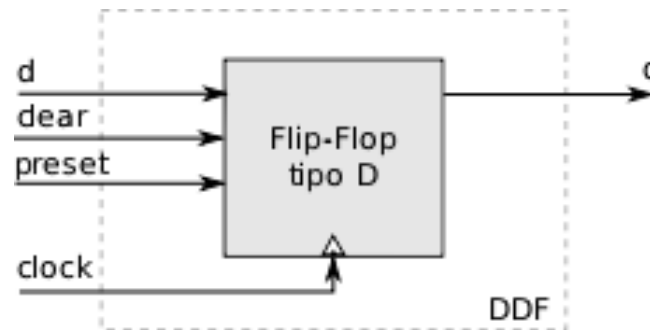
```
library ieee;  
use ieee.std_logic_1164.all;  
  
entity FlipFlopD is  
  port(  
    clock: in std_logic;  
    d:      in std_logic;  
    clear:  in std_logic;  
    preset: in std_logic;  
    q:      out std_logic  
  );  
end entity;
```



Flip Flop em VHDL

```
library ieee;
use ieee.std_logic_1164.all;

entity FlipFlopD is
  port(
    clock: in std_logic;
    d:      in std_logic;
    clear:  in std_logic;
    preset: in std_logic;
    q:      out std_logic
  );
end entity;
```



D Flip Flop Truth Table

CL (Note 1)	D	R	S	Q
↗	0	0	0	0
↘	1	0	0	1
↖	x	0	0	Q
x	x	1	0	0
x	x	0	1	1
x	x	1	1	1

No Change

x = Don't Care Case

Note 1: Level Change

Flip Flop em VHDL

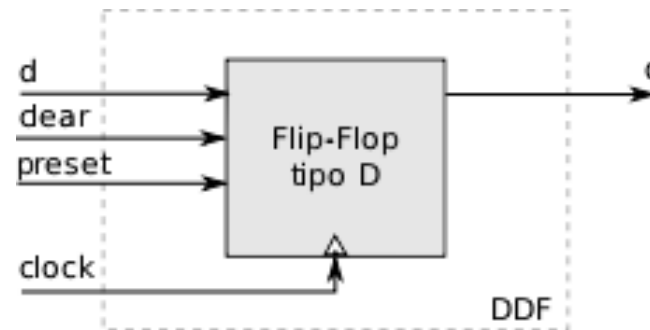
```

library ieee;
use ieee.std_logic_1164.all;

entity FlipFlopD is
  port(
    clock:  in std_logic;
    d:      in std_logic;
    clear:  in std_logic;
    preset: in std_logic;
    q:      out std_logic
  );
end entity;

architecture arch of FlipFlopD is
  begin
    process(clock, clear, preset) begin
      if (clear = '1') then
        q <= '0';
      elsif (preset = '1') then
        q <= '1';
      elsif (rising_edge(CLOCK)) then
        q <= d;
      end if;
    end process;
  end architecture;

```



D Flip Flop Truth Table

CL (Note 1)	D	R	S	Q
↗	0	0	0	0
↘	1	0	0	1
↖	x	0	0	Q
x	x	1	0	0
x	x	0	1	1
x	x	1	1	1

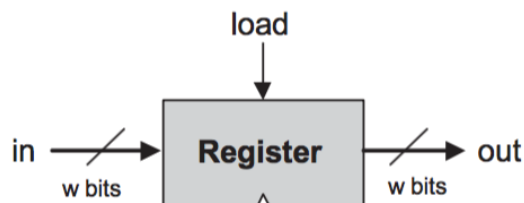
No Change
x = Don't Care Case

Note 1: Level Change

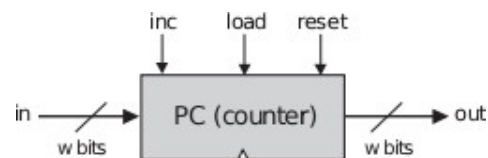
O que fazer

- Registrador

- 8bit
- 16bit
- 32bit
- 64bit

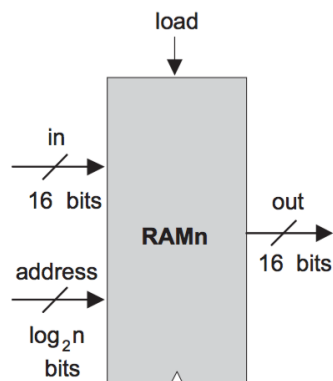


- ProgramCounter



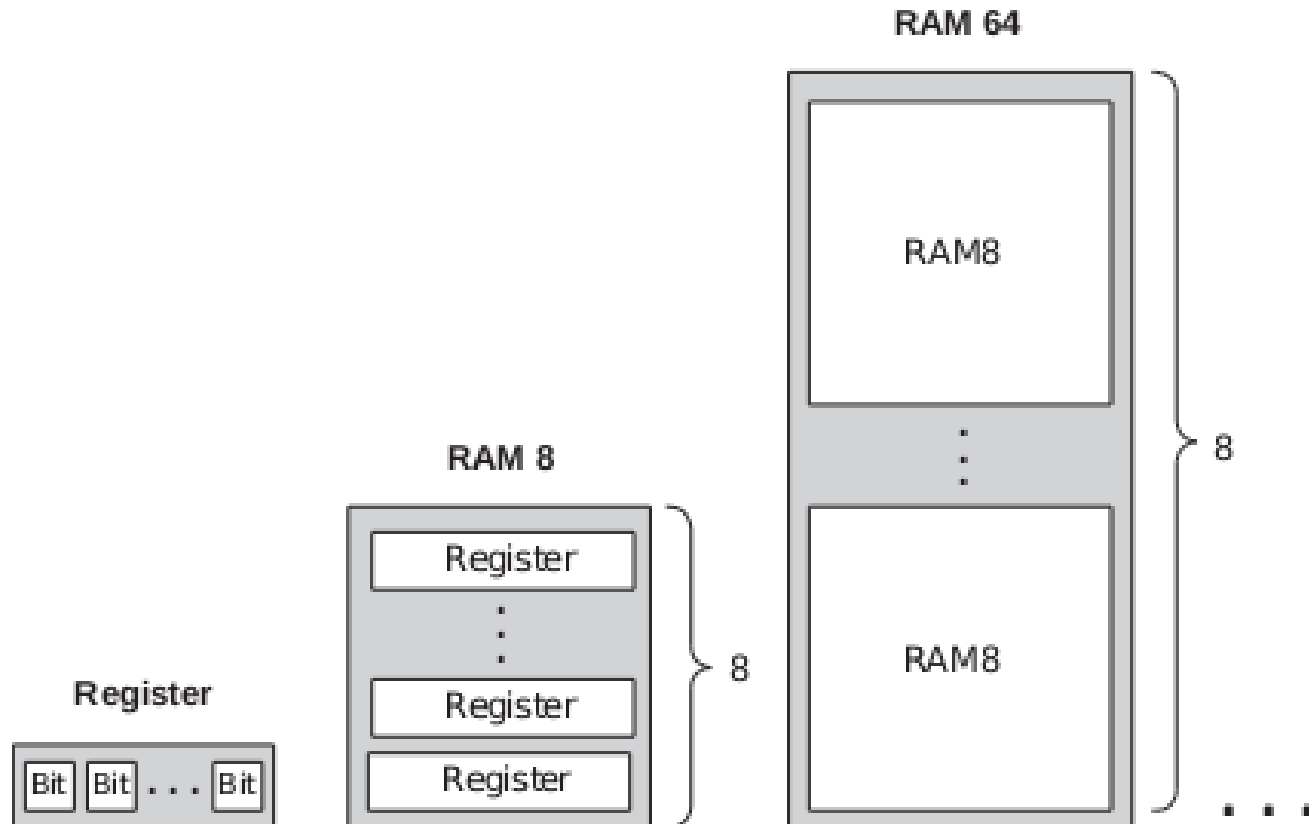
- Memórias:

- RAM 8
- RAM 64
- RAM 512
- RAM 4K



Memória RAM

Lembre que você pode combinar entidades para criar entidades mais completas.



Novas rubricas → Mais simples para todos !

Desenvolvedor

Conceito

Insatisfatório

- (I)
- Se comprometeu a fazer algum desenvolvimento (kanban) e não fez.
 - Criou o branch mas não fez o desenvolvimento completo.

Satisfatório

- (A)
- Desenvolveu as rotinas atribuídas pelo Facilitador para passarem nos testes!
 - Acompanhou o Kanban board (github project). Ex: Puxou tarefas, etc.
 - Submeteu alterações por pull requests.
-

Facilitador

Conceito

Insatisfatório

- (I)
- Não acompanhou o projeto, deixando os colegas sozinhos.
 - O relatório das atividades não é condizente com o real desempenho dos integrantes (analisado via git).

Satisfatório

- (A)
- Atualizou o repositório pelo Fork.
 - Fez a manutenção do Kanban board (GitHub project). Ex: cria cards, atribui tarefas, da feedback de insues.
 - Aceitou os pull-requests.
 - Resolveu conflitos de merge nos pull requests.
 - Acompanhou o desenvolvimento do grupo, dando o suporte sempre que necessário
 - **Entregou o branch master sem nenhum erro (passando no travis)**
 - Fez o relatório das atividades descrevendo o papel de cada integrante com clareza.
-

Elementos de Sistemas - Projeto E - Ferramental

Rafael Corsi - rafael.corsi@insper.edu.br

Março, 2018

Início do projeto

1. Atualizar arquivo ScramMaster.txt
2. Atualizar repositório com o upstram
3. Adicionar o novo script de teste ao travis : E-LogicaSequencial/script/testeLogicaSequencial.py
4. Criar projeto no github
5. Atribuir tarefas e acompanhar o desenvolvimento

Process VHDL

Em VHDL quando desejamos fazer algo sequencial é necessário usarmos uma estrutura chamada de process, que possui a declaração a seguir :

```
process (optional sensitivity list)
    declarations
begin
    sequential statements
end process;
```

Nesse **process** possuímos a lista de sensibilidade (*sensitivity list*) que indica quando o process será executado. Podemos pensar da seguinte maneira, sempre que algum sinal que está listado nessa lista de sensibilidade mudar de valor (0 -> 1, 1 -> 0) o processo será executado, vemos o exemplo a seguir :

```
process(A)
begin
    Q <= A;
end process;
```

Sempre que o sinal A (sinal ou porta) alterar de valor o sinal Q será atribuído com o seu valor

Insper

www.insper.edu.br