

# ELT134 – Laboratório de Arquitetura e Organização de Computadores

Método de Projeto de CPUs

Autor: Prof. Ricardo de Oliveira Duarte

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir de DE10-Lite\_User\_Manual.pdf encontrado em:  
[https://www.terasic.com.tw/cgi-bin/page/archive\\_download.pl?Language=English&No=1021&FID=a13a2782811152b477e60203d34b1baa](https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=1021&FID=a13a2782811152b477e60203d34b1baa)



## Atribuição-NãoComercial-Compartilha Igual 4.0 Internacional



BY

NC

SA

### Atribuição-NãoComercial-Compartilha Igual CC BY-NC-SA

Esta licença permite que outros remixem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que atribuem a você o devido crédito e que licenciem as novas criações sob termos idênticos.

Texto completo da licença (em Português): <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.pt>

# Assuntos

- Consideração inicial
- Modelos de *CPUs*
- Fluxo geral de projetos
- Detalhamento de cada etapa do fluxo
- Referências complementares

# Consideração inicial

- O método apresentado nessa disciplina tratará unicamente do:
  - Projeto, desenvolvimento e implementação de *CPUs* programáveis de um único núcleo (*single core*) sintetizáveis em *FPGAs*.

# Modelos de *CPUs*

- Modelo Ciclo Único
- Modelo Multiciclo
- Modelo *Pipeline* simples

Observação 1: Os métodos de projeto apresentados se restringem somente aos modelos acima apresentados.

Observação 2: Há implementações que combinam multiciclo e *pipeline*, mas não são tão comuns. E não são tratadas nessa Unidade da disciplina.

# Fluxo geral de projetos

- Etapa 1: Definição de requisitos do projeto.
- Etapa 2: Detalhamento (ou refinamento) dos requisitos do projeto.
- Etapa 3: Modelagem do sistema.
- Etapa 4: Continuação da modelagem do sistema.
- Etapa 5: Implementação do sistema
- Etapa 6: Verificação e validação do sistema

# Detalhamento de cada etapa do fluxo

- Etapa 1: Definição de requisitos do projeto
- Nessa etapa define-se:
  - Quais instruções sua CPU poderá processar, ou seja, o início da definição da *ISA*.
  - Tamanho(s) da instrução.
  - Tamanho do(s) dado(s) que a sua CPU será capaz de processar.
  - Capacidade de memória que a sua CPU será capaz de endereçar (tem a ver com o *PC*)
  - Formas de endereçamento que a sua CPU será capaz de tratar.
  - Formas de E/S que a sua CPU será capaz de tratar.
  - Priorizará ou não o uso de banco de registradores no processamento dos dados?
  - Modelo RISC ou Modelo CISC.
  - Modelo Von Neumann ou Modelo Harvard.
  - Modelo de *CPU*: ciclo único, multiciclo ou *pipeline* simples?
  - *Endianness* – ordenamento de *bytes* dentro da palavra (*little endian* ou *big endian*)

# Detalhamento de cada etapa do fluxo

- Etapa 2: Detalhamento (ou refinamento) dos requisitos do projeto.
- Detalhe cada instrução da *ISA* do seu processador (equivale ao detalhamento do *datasheet* do seu processador)
  - Mnemônico
  - Tamanho da instrução
  - Mapa de campos da sua instrução
  - *Opcode*: um código binário único associado a cada instrução.
  - Quantidade e tipos de operandos
  - Tamanho de cada operando
  - Tipo de operações que a instrução demanda
  - Tipo de endereçamento da instrução
- Adote como base para realizar o seu trabalho o *template* de discriminação de instrução disponibilizado pelo professor (*template\_discriminacao\_instrucao.docx*).



# Detalhamento de cada etapa do fluxo

- Etapa 3: Modelagem do sistema.
- Essa etapa depende do Modelo de *CPU* escolhido:
  - Para Modelo Ciclo Único e/ou Modelo *Pipeline* simples siga Etapa 3a
  - Para Modelo Multiciclo siga Etapa 3b

# Detalhamento de cada etapa do fluxo

- Etapa 3a: Modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Nessa etapa você desenhará o caminho de dados completo da sua *CPU*. Use papel, lápis e borracha.
- Para cada instrução detalhada na etapa 2, identifique as operações que cada uma delas demanda para ser implementada. Você precisa considerar o conceito de ciclo de instrução (ou ciclo de máquina) nesse passo!
- Associe cada operação a um ou mais componentes que aparecerão no caminho de dados (ou bloco operacional) da sua *CPU*.
- Construa (desenhe) o caminho de dados com todos os componentes.
- Interligue todos os componentes de forma que a *CPU* possa executar corretamente qualquer instrução da sua *ISA*.

# Detalhamento de cada etapa do fluxo

- Etapa 3a: Modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Cada componente (desenho) deve ter a ele associado um nome curto, mas representativo de sua(s) responsabilidades, além de ter uma identidade única.
- Cada componente desenhado deve conter todas as entradas e saídas de dados (posição horizontal) e entradas e saídas de controle (posição vertical).
- Multiplexadores deverão ser adicionados ao caminho de dados sempre que um dado sinal tenha proveniência de diferentes fontes de dados.
- Multiplexadores deverão também seguir as duas regras de caracterização citadas nesse slide.

# Detalhamento de cada etapa do fluxo

- Etapa 3a: Modelagem de uma CPU Ciclo Único e/ou Modelo *Pipeline* simples.
- Se o Modelo da sua CPU é Ciclo Único essa etapa do fluxo de projeto termina aqui.
- Se o Modelo da sua CPU é um *Pipeline* Simples, haverá a necessidade de inserir os registradores de pipeline no caminho de dados.
- Siga as duas primeiras regras de caracterização de componentes recomendadas no slide anterior para caracterizar os registradores de *pipeline* que você incluirá.
- A quantidade de registradores de *pipeline* no caminho de dados vai depender do número de estágios que você defina para implementar o *pipeline*.
- O tamanho em bits de cada registrador de *pipeline* vai variar em função da quantidade de dados (em *bits*) que você terá que encaminhar de um estágio de *pipeline* ao outro.

# Detalhamento de cada etapa do fluxo

- Etapa 3a: Modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Após ter inserido (desenhado) os registradores de *pipeline* e ter interligado os mesmos aos outros componentes do caminho de dados, essa etapa estará concluída.

# Detalhamento de cada etapa do fluxo

- Etapa 3b: Modelagem de uma *CPU* Multiciclo.
- Nessa etapa você desenhará o caminho de dados completo da sua *CPU*. Use papel, lápis e borracha.
- Para atingir o objetivo acima, você seguirá o método de projeto RTL detalhado no capítulo 5 do livro Digital Design do Frank Vahid.
- Para cada instrução detalhada na etapa 2, comece a construir o diagrama da Máquina de Estados Finitos de Alto Nível (*HLFSM*).
- As ações (ou a ação) em cada estado da *HLFSM* representará as operações que cada instrução demanda para ser implementada.
- O conceito de ciclo de instrução (ou ciclo de máquina) deverá ser respeitado para realização nessa etapa!

# Detalhamento de cada etapa do fluxo

- Etapa 3b: Modelagem de uma *CPU* Multiciclo.
- Repita o passo anterior para todas as instruções de sua ISA.
- Lembre que a transição de saída do último estado de cada instrução deve se dirigir para o estado da *HLFSM*, que tem por objetivo executar a busca de instrução (*fetch*).
- Cada ação existente em cada estado do diagrama da *HLFSM* representará cada operação que cada instrução demanda para ser implementada no caminho de dados.
- Cada ação vai demandar a inserção (desenho) e/ou adaptação de um componente no caminho de dados.

# Detalhamento de cada etapa do fluxo

- Etapa 3b: Modelagem de uma *CPU* Multiciclo.
- Quando todas as ações dos estados da *HLFSM* já estiverem sendo realizadas por um ou uma combinação de componentes no seu caminho de dados, para implementação de todas as instruções da *ISA* da sua *CPU*, essa etapa estará concluída para o modelo de *CPU* multiciclo.




# Detalhamento de cada etapa do fluxo

- Etapa 4: Continuação da modelagem do sistema.
- Essa etapa depende do Modelo de *CPU* escolhido:
  - Para Modelo Ciclo Único e/ou Modelo *Pipeline* simples siga Etapa 4a
  - Para Modelo Multiciclo siga Etapa 4b
- Essa etapa gera como produto a modelagem e a montagem da unidade de controle da sua CPU.

# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Nessa etapa você desenhará e projetará a unidade de controle completa da sua *CPU*. Papel, lápis e borracha.
- No caso do Modelo Ciclo Único, qualquer instrução será executada em um único ciclo de clock. O tamanho do período do ciclo de clock será definido pela instrução mais lenta da *ISA*.
- Nesse modelo a leitura dos elementos de memória é assíncrono e a escrita é síncrona (por ex.: na borda de subida do clock).
- No caso do Modelo *Pipeline* simples, a unidade de controle será composta por 2 partes:
  - O controle das instruções da *ISA*, que é o mesmo procedimento do Modelo Ciclo Único.
  - O controle de adiantamento (*forwarding*), congelamento (*stalling*) e despejo (*flushing*).

# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Para cada instrução detalhada na etapa 2, identifique os componentes do caminho de dados da etapa 3 que cada instrução demanda para ser implementada.
- Cada componente terá à ele associado um ou mais sinais de controle.
- Identifique o estado (nível lógico 1, 0 ou X) que cada sinal de controle de cada componente deve ficar, para que a instrução seja corretamente executada.
-  Construa e preencha uma tabela conforme o *template* dado pelo professor: *template\_discriminacao\_sinais\_ctrl.docx*

# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Desenhe um bloco acima do caminho dados da sua *CPU* e dê a ele o nome de Unidade de Controle Principal.
- As entradas da Unidade de Controle Principal da sua *CPU* serão:
  - O campo *opcode* e/ou o campo único identificador da instrução corrente.
  - Os sinais de controle porventura gerados por componentes de seu caminho da dados.
- As saídas da Unidade de Controle Principal da sua *CPU* serão as entradas de controle dos componentes do caminho de dados da sua *CPU*.
- Se a sua *CPU* tem por modelo Ciclo Único esta etapa termina aqui.


# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Analise o caminho de dados *pipeline* e registre em um documento, todos os conflitos de dados e conflitos de controle que poderão acontecer no seu *pipeline*, ou seja, todas as situações e instruções envolvidas na ocorrência de cada conflito.
- Analise qual ou quais técnicas precisarão ser empregadas na solução de cada conflito. Técnicas disponíveis:
  - Antecipação de desvios
  - Adiantamento (*forwarding*)
  - Congelamento (*stalling*)
  - Despacho (*flushing*)
- Altere o desenho do caminho de dados conformando todos os componentes para atender cada solução de conflito (incluindo o desenho de todos os sinais de controle dos registradores de *pipeline*, modificados para atender as situações de conflito).

# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
- Altere o arquivo *template\_discriminacao\_sinais\_ctrl.docx* de forma a atender as modificações que você realizou no passo anterior.
- Desenhe um bloco abaixo do caminho dados da sua *CPU* e dê a ele o nome de Unidade de Controle de Conflitos.
- As entradas da Unidade de Controle de Conflitos da sua *CPU* serão:
  - Os campos de endereços de registradores em estágios de seu interesse no pipeline.
  - Eventuais sinais de controle do *pipeline* (Ex.: sinais de habilita escrita).
  - Eventuais sinais de controle oriundos da Unidade de Controle Principal (Ex.: *branch*)
- As saídas da Unidade de Controle de Conflitos da sua *CPU* serão as entradas de controle de alguns componentes do caminho de dados da sua *CPU*. (Ex.: registradores de *pipeline*, multiplexadores, etc.)

# Detalhamento de cada etapa do fluxo

- Etapa 4a: Continuação da modelagem de uma *CPU* Ciclo Único e/ou Modelo *Pipeline* simples.
-  Escreva todas as funções booleanas da Unidade de Controle de Conflitos da sua *CPU* em um documento a parte. Você precisará delas para implementar sua Unidade de Controle de Conflitos em VHDL.
- Se a sua *CPU* tem por modelo *Pipeline* simples esta etapa termina aqui.

# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Essa etapa independe do Modelo de *CPU* escolhido.
- Método usado: *Component based design*
  - Cada componente da sua CPU, ou seja, cada componente do seu caminho de dados e de cada unidade de controle, deverá ter uma *entity*, uma *architecture* e um arquivo *vhd* com um nome único representativo.
  - Método de construção do caminho de dados: abstração *bottom-up*
- Descreva seus componentes na *HDL* usando modelos apropriados
  - Componentes combinacionais simples – descreva-os no modelo *dataflow*
  - Componentes que implementam funções booleanas – descreva-os no modelo *dataflow*
  - Componentes combinacionais complexos – descreva-os no modelo *comportamental*
  - Componentes sequenciais – descreva-os no modelo *comportamental*
  - Componentes integradores e/ou de mais alta hierarquia – descreva-os no modelo *estrutural*



# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Descreva seus componentes na *HDL* usando técnicas apropriadas
  - Indentação
  - Nome de *signals* and *variables* significativas.
  - Comentários relevantes.
  - Entidades somente com *signals std\_logic* e/ou *std\_logic\_vector*.
  - Use somente de estruturas e comandos da HDL que sejam sintetizáveis.
  - Use somente *packages* padrão: *std\_logic\_1164* e *numeric\_std*.
  - Use conversão de tipos apropriada quando necessário.
  - Inclua somente os *signals* indispensáveis na lista de sensibilidade de cada *process*.
  - Não introduza *latches* desnecessários.
  - Use somente *variables* quando for necessário.

# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Sugestão de uso de ferramentas:
  - Visual Studio Code (VS Code) com extensão TerosHDL
    - Razão: é uma interface bem mais leve que Quartus II e ModelSim, com recursos de indentação, reconhecimento de comandos da linguagem, templates e recursos de autocompletar. Permite a execução de scripts de compilação, simulação, visualização de formas de onda, etc. a partir de uma janela de terminal.
  - Intel-Altera Quartus II e ModelSim instalados.
    - Mas só usaremos os executáveis por linha de comandos dessas ferramentas. Não utilizaremos o ambiente gráfico interativo.
    - Razão: temos muitos componentes para compilar, sintetizar e simular. Fazer isso com uma ferramenta gráfica interativa será uma tarefa muito cansativa.

# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Sugestão de uso de ferramentas:
  - *MAX 10 FPGA device support da INTEL-Altera™* - biblioteca e *drivers* para gravação para os dispositivos (FPGAs) da família MAX 10 da INTEL-Altera™
    - Razão: é o device driver do kit de FPGA que temos no laboratório e que será usado para você fazer seus testes de funcionamento dos seus sistemas com material eletrônico.

# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Você criará arquivos executáveis pelo Terminal de comandos do MS-Windows ou Linux, contendo comandos para:
  - Criação de projetos
  - Compilação
  - Síntese
  - Geração de *testbenchs*
  - Simulação
  - Exibição de formas de ondas
  - Verificação de funcionamento
  - Gravação
- Como base preparei alguns modelos de arquivos de comandos que você pode usar para criar os seus. Todos os arquivos necessários estão compactados em um arquivo *somador.rar* Descompacte-o e as pastas serão automaticamente criadas.
- Veja no próximo slide a relação de *scripts* (arquivos) que eu disponibilizei para vocês.

# Detalhamento de cada etapa do fluxo

- Etapa 5: Implementação do sistema.
- Veja alguns exemplos de arquivos executáveis pelo Terminal de comandos do MS-Windows ou Linux, contendo comandos:
  - *batch\_script\_tb\_somador\_modelsim\_call.bat*
  - *batch\_task\_window\_script\_quartus\_sh.bat* que chama um arquivo TCL:  
*tcl\_somador\_quartus\_sh.tcl* ambos arquivos se encontram na subpasta: *somador\_quartus\_sh*
- Se você preferer trabalhar com *ghdl* e *gtkwave*, use o *batch*:
  - *batch\_script\_ghdl\_gtkwave\_call.bat* que chama um arquivo TCL: *gtkwave\_print.tcl*

# Detalhamento de cada etapa do fluxo

- Etapa 6: Verificação e validação do sistema
- Essa etapa independe do Modelo de *CPU* escolhido.
- Verifique se o seu componente e/ou o seu sistema apresenta o comportamento funcional que você espera por meio do método de verificação com *testbenchs* automatizados.
  - Para aprender como escrever um bom *testbench* consulte a referência [4] do planejamento de aulas dessa disciplina.
  - Disponibilizo à vocês um modelo de *testbench* automatizado *tb\_função.vhd*. Nesse caso o *DUT* é *função.vhd*. Ambos os arquivos se encontram compactados no arquivo: *LSD\_guia\_10\_ERE\_códigos.rar*

# Detalhamento de cada etapa do fluxo

- Etapa 6: Verificação e validação do sistema
- Descreva um *testbench* apropriado para validar cada componente (*DUT – Device Under Test*) que você escreveu em *HDL* na etapa 5.
- Verifique por simulação e visualização da forma de onda simulada cada conjunto de arquivos *vhd*:
  - *componente.vhd*
  - *tb\_componente.vhd*
- Use como base os modelos de arquivos de *script* de comandos que eu disponibilizei para vocês na pasta: somador para criar os seus próprios arquivos de *script*.

# Detalhamento de cada etapa do fluxo

- Etapa 6: Verificação e validação do sistema
- Uma vez verificado o funcionamento do seu componente e/ou sistema por simulação, valide-o agora em *kit* de FPGA.
  - Para aprender como gerar um arquivo sintetizável a partir do DUT que você verificou na etapa anterior, prepare o documento de *pin assignment* para o DE10-Lite.
  - Associe as entradas e saídas do seu componente/sistema às entradas e saídas do FPGA do kit DE10-Lite.
  - Cada pino do FPGA do kit DE10-Lite está associado a um recurso do kit.
  - Entende-se por recurso do kit: chaves (*switches*), botões (*push-buttons*), *leds*, *displays* de sete segmentos, sinais de *clock*, GPIOs nos *sockets* de interface, etc. Consulte a referência [7] do planejamento de aulas dessa disciplina.
  - Disponibilizei um arquivo .csv (comma separated value) com a associação de pinos padrão do kit DE10-Lite. O nome do arquivo é: *pinagem\_DE10Lite.csv*
  - Adapte o arquivo acima conforme as suas necessidades para a verificação.



# Detalhamento de cada etapa do fluxo

- Etapa 6: Verificação e validação do sistema
- Uma vez verificado o funcionamento do seu componente e/ou sistema por simulação, valide-o agora em *kit* de FPGA. (continuação)
  - Uma vez adaptado o arquivo acima conforme as suas necessidades para a verificação.
  - Faça um *copy and paste* dos *signals*/pinos de seu interesse e adapte o arquivo:
  - *tcl\_somador\_quartus\_sh.tcl*
  - Você vai precisar adaptar o *batch\_task\_window\_script\_quartus\_sh.bat* que chama um arquivo TCL: *tcl\_somador\_quartus\_sh.tcl*
  - Descomente a última linha do *batch\_task\_window\_script\_quartus\_sh.bat* para fazer a gravação do arquivo *.sof* no *FPGA* do *kit DE2-Lite*.

# Referências complementares

- *DE10-Lite\_User\_Manual.pdf*
- *Tcl Script File (.tcl) Definition.pdf*
- *Creating and Running Tcl Scripts.pdf*
- *Quartus Linux Programming the FPGA with command-line.pdf*
- *Programming using linux command line.pdf*
- *QuartusII Command Line Scripting.pdf*
- *Using the Quartus II Executables in Shell Scripts.pdf*
- *Intel Quartus Prime Pro Edition - Shell Scripting User Guide.pdf*
- *std\_logic vs std\_ulogic – VHDLwhiz.pdf*