Universidade Federal do Rio Grande do Sul
Instituto de Informática
Departamento de Informática Aplicada

Disciplina: INF01058 - Circuitos Digitais
Professor: Mateus Grellert

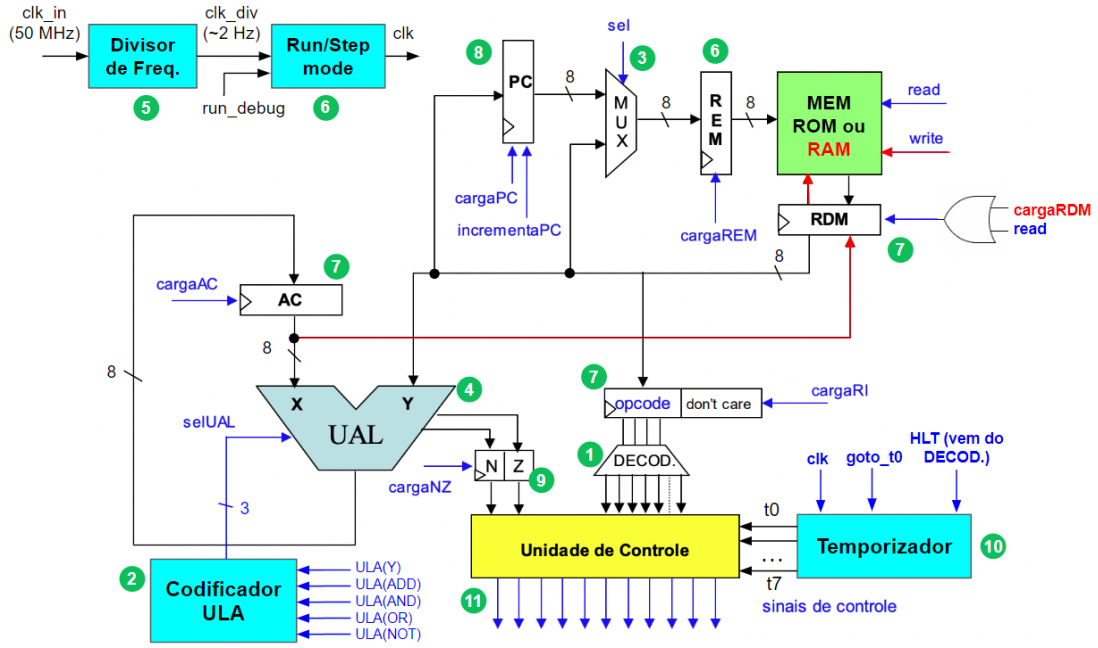
Projeto Final: Neander Completo com Memória

Objetivo:

Projetar e simular um Neander Completo com controle baseado em expressões lógicas (sem uso explícito de FSM), assim como uma memória de programa e de dados.

Instruções:

O processador hipotético Neander é uma versão bastante simplificada, porém com diversas funcionalidades presentes em processadores atuais. Portanto, ele é uma ótima alternativa para ensino de Circuitos Digitais e de Arquitetura de Computadores. A Fig. 1 apresenta a organização do Neander, enquanto que a Tab. I apresenta sua arquitetura. Cada componente possui um número identificador que será utilizado no planejamento proposto a seguir. Note que os sinais de clock e reset foram omitidos para facilitar a visualização.



Fig, 1 - Organização do Neander, com sinais de controle indicados em azul. Símbolos em vermelho são opcionais e fazem parte do desafio (ponto extra no projeto).

Como podemos observar na figura, o Neander possui os seguintes elementos principais: registradores (RDM, REM, RI, AC e PC), uma ULA, uma unidade de decodificação, multiplexadores e uma memória onde estão os dados e as instruções do programa.

Sua tarefa neste projeto é gerar uma versão completa do Neander utilizando uma memória (ROM ou RAM) que irá guardar o programa (*assembly*) e os dados. Todas as unidades (exceto a memória e o controle) serão discutidas e implementadas

(parcialmente ou integralmente) em sala de aula.

Planejamento dos Componentes

Recomenda-se que os componentes da ULA sejam desenvolvidos durante o semestre, de forma que haja tempo suficiente para sua integração e prototipação na placa FPGA até o prazo de entrega. A Tab. I a seguir apresentada um planejamento para implementação destes componentes (utilizando os identificadores da Fig. 1), organizados de acordo com a ordem dos conteúdos dados em aula:

Tab. I - Componentes a serem implementados e as aulas habilitadores para seu desenvolvimento

ID	Circuito	Descrição	Aula
1	Decodificador	Recebe o opcode de 4 bits na entrada e gera na saída um bit para cada instrução do NEANDER (LDA, STA,...). Estes bits serão utilizados no controle para identificar a instrução atualmente sendo executada	4
2	Codific. ULA	Recebe os bits relativos às operações da ULA (`ULA_Y`, `ULA_ADD`, ...) e gera o seletor de operação de 3 bits.	4
3	MUX	MUX para seleção da fonte do REM (PC ou RDM)	4
4	ULA	ULA do Neander (Y, ADD, AND, OR, NOT)	6
5	Divisor de Clock	Recebe o clock de 50 Mhz da FPGA e gera um clock de aproximadamente 2 Hz.	11
6	Run/Step Mode	Circuito que ajusta o sinal de clock do divisor para o clock efetivo do circuito. No modo RUN, o clock opera conforme o clock de entrada. No modo DEBUG,	11
7	AC, RDM, REM	Registradores especiais do NEANDER de 8 bits com habilitador e reset assíncrono.	13
8	PC	Registradores de 8 bits contador com habilitador para carga e para contagem e reset assíncrono.	13
9	N, Z	Registradores de 2 bits com habilitador para as flags de estado do NEANDER e reset assíncrono.	13
10	Temporizador	Conta os passos das instruções. Será utilizado pelo controle. Possui um sinal de reset assíncrono e um reset síncrono (`goto_t0`).	13
11	UC	Unidade de Controle implementada como uma FSM ou como uma máquina de controle hardcoded.	18

Implementação da ROM/RAM

A memória ROM do Neander será implementada com base no código em VHDL disponibilizado neste LINK. Esta memória suporta programas de até 256 posições (pois

possui endereçamento de 8 bits). O exemplo disponibilizado implementa um código simples que pode ser usado para teste.

Para utilizar esta unidade no seu esquemático, você deve gerar um símbolo (Create Symbol File) seguindo os mesmos passos que utilizamos em atividades anteriores.

A memória RAM não será disponibilizada pois sua implementação será considerada um desafio que irá proporcionar um ponto extra no trabalho. É importante ressaltar que memórias RAM normalmente necessitam de dois ciclos para acesso ao dado (um para setar o endereço, outro para registrar na saída), portanto as duplas que decidirem utilizar essa implementação precisarão alterar o módulo de controle.

Implementação do Controle

O controle desta versão do Neander será implementado utilizando expressões lógicas para cada sinal de controle. As instruções do Neander não são implementadas em um único ciclo, portanto é necessário separar sua execução em passos. Esses passos serão contados pelo circuito temporizador implementado em um laboratório anterior.

A Fig. 2 apresenta as expressões booleanas de cada sinal de controle do Neander. Note que para saber o resultado de cada sinal é preciso saber qual instrução está presente no Registrador de Instrução (RI). Para isso, usamos o circuito de decodificação (DECOD.) apresentado na Fig. 1. Note também que precisamos saber em que etapa da execução da instrução estamos. Essa informação virá do circuito temporizador.

carga REM = t0 + t3.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t5.(STA+LDA+ADD+OR+AND)

incrementa PC = t1 + t4.(STA+LDA+ADD+OR+AND) + t3.(JN.N' + JZ.Z')

carga RI = t2

sel = t5.(STA+LDA+ADD+OR+AND)

carga RDM = t6.STA

Read = t1 + t4.(STA+LDA+ADD+OR+AND+JMP+JN.N+JZ.Z) + t6.(LDA+ADD+OR+AND)

Write = t7.STA

UAL(Y) = t7.LDA

UAL(ADD) = t7.ADD

UAL(OR) = t7.OR

UAL(AND) = t7.AND

UAL(NOT) = t3.NOT

carga AC = t7.(LDA+ADD+OR+AND) + t3.NOT

carga NZ = t7.(LDA+ADD+OR+AND) + t3.NOT = carga AC

carga PC = t5.(JMP+JN.N+JZ.Z)

goto t0 = t7.(STA+LDA+ADD+OR+AND) + t3.(NOP+NOT+JN.N'+JZ.Z') + t5.(JMP+JN.N+JZ.Z)

Fig. 2 - Expressões lógicas dos sinais de controle do Neander. Os sinais t0, t1…,t7 vêm do temporizador.

Implementação do Temporizador

O temporizador (ilustrado na Fig. 3) é necessário para marcar os passos do controle e deve ser implementado como um registrador contador de 3 bits (conta de 0 a 7) com um decodificador acoplado à saída. A ideia é que cada passo tenha um bit de saída associado, sendo que o bit B_i deve ser `'1'` se o `contador = i` e `'0'` caso contrário. Este contador também possui dois sinais de reset: um reset assíncrono,

controlado pelo reset global do sistema, e um reset síncrono, controlado pela entrada `goto_t0`. Além disso, o temporizador possui uma entrada HLT, que para a contagem.

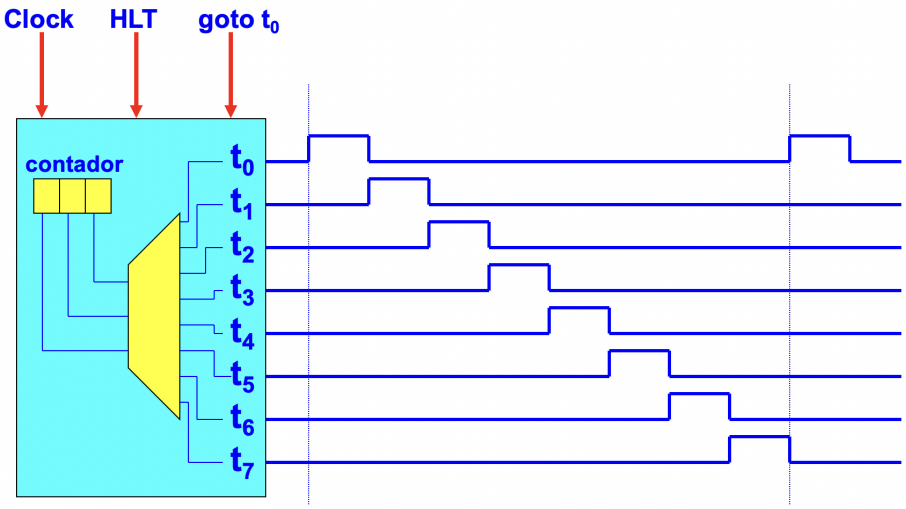


Fig. 3 - Visão *top level* do temporizador.

Implementação em FPGA (Modelo: EP3C16F484C6)

A Fig. 3 apresenta a forma como o circuito deve ser mapeado para FPGA. O valor dos registradores **PC** e **AC** devem ser representados no display de 7 segmentos na base hexadecimal, sendo os dois primeiros displays para o PC, e os dois últimos para o AC. As *flags* de estado **N** e **Z** da ULA devem ser representadas nos LEDs da placa. O circuito deverá ter dois tipos de execução: ***run mode***, em que o clock avança normalmente a cada período, assim como o ***debug mode***, em que o passo de clock é dado quando um botão é pressionado (***step up***). Esse controle do modo de operação será realizado por um pino do tipo *switch*.

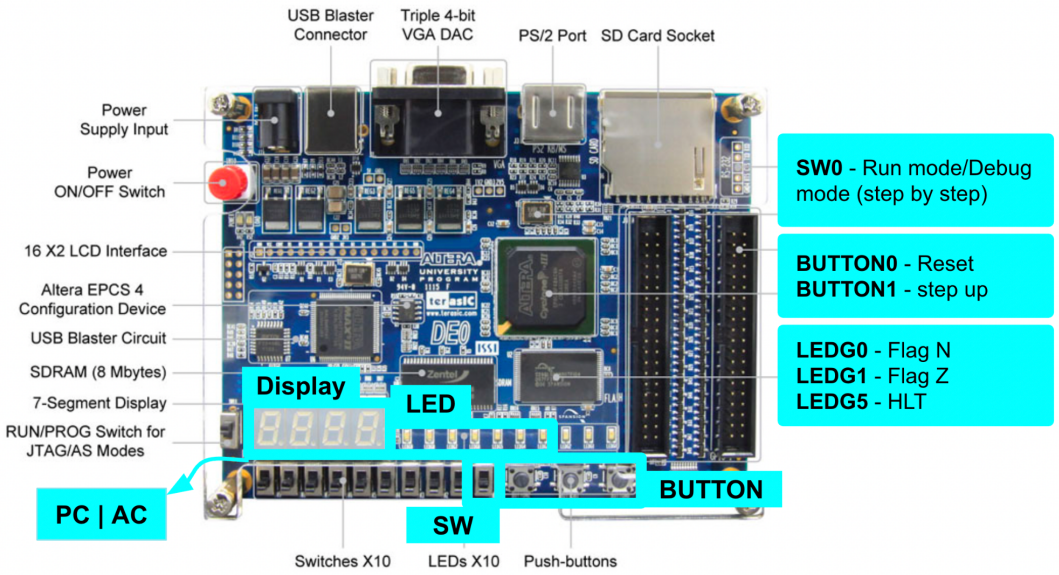


Fig 3 - Mapeamento do circuito para os pinos da FPGA.

Avaliação:

No dia da apresentação, os alunos devem saber:

- Demonstrar circuito funcionando na FPGA

Entrega no Moodle:

Arquivo ZIP com padrão de nome **cartao1_cartao2.zip** contendo SOMENTE:

- Pasta com projeto do Quartus (incluindo arquivos e diretórios criados pela ferramenta).
- Captura de tela da simulação em forma de onda (pode ficar dentro da pasta do projeto Quartus).

Acesso à FPGA fora dos horários de aula:

Quatro FPGAs serão disponibilizadas para os alunos no laboratório 215 do prédio 43413 (mesmo prédio das aulas). Os alunos podem solicitar a placa a um dos alunos presentes no laboratório. Se o laboratório estiver fechado, procure o professor na sala 222 do prédio 43424.