Trabalho de conclusão de disciplina, projeto K&S

Professor: João Fragoso

Disciplina: Sistemas Digitais

Data 24/11/2022

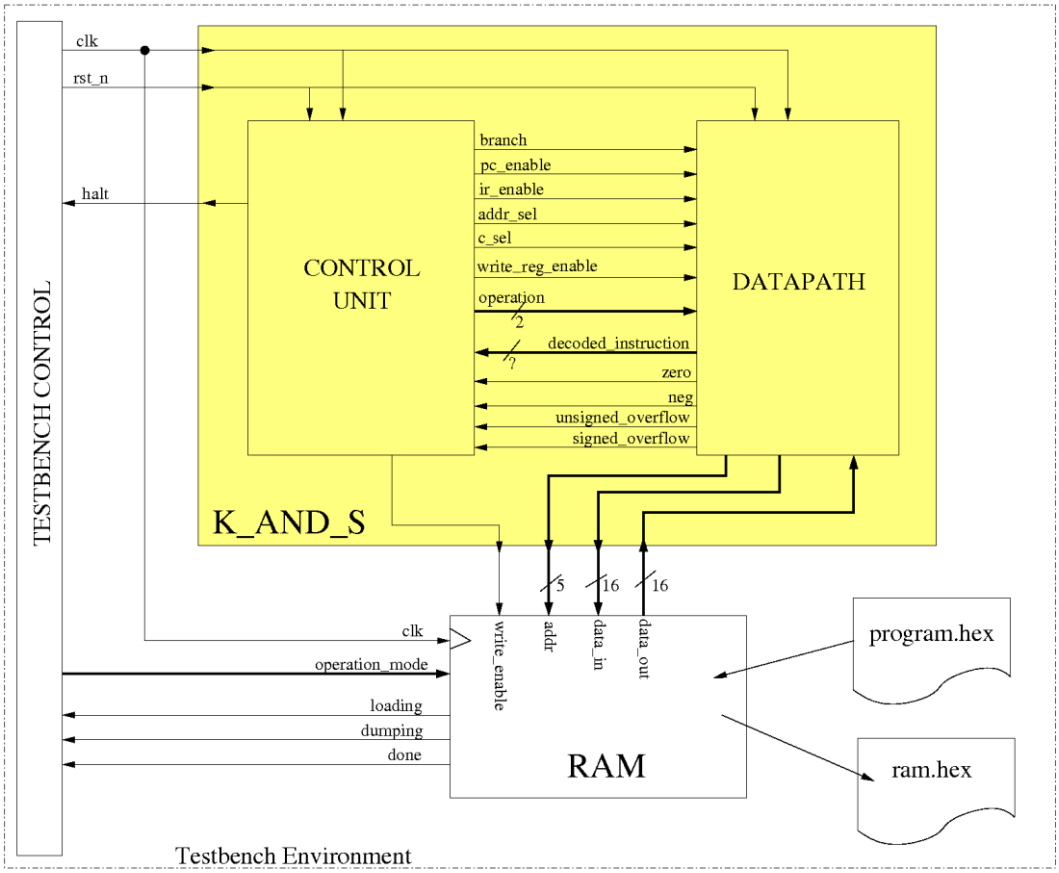
Nomes: Maikon Michel de Almeida, Gabriel Garcia

Introdução

A proposta do trabalho é criar um microcomputador, sendo ele programado em nível RTL. Uma máquina que precisa conter dois módulos: o “data_path” que faz as operações e o “control” que controla a primeira por meio do acionamento do fluxo dos dados operativa.

Outra característica importante sobre esse microcomputador, é que ele deve ser capaz de executar programas de um módulo de memória Ram que viria pronta, além de conter dentro dela os programas em hexadecimal a serem executados.

Portanto, ele deve se apresentar na seguinte configuração, incluindo os sinais do diagrama:



Estratégia de equipe de trabalho

O método utilizado neste projeto se mostrou bastante eficiente. Foram realizados vários revezamentos entre o programador e o apoiador que trabalharam sempre em reuniões (virtuais) invés de aplicar a divisão por tarefas individuais.

O apoiador acompanhava ao vivo o desenvolvimento do projeto e dava suporte com a procura de informações, investigação dos erros e analise dos sinais além da realização de testes enquanto o outro trabalhava no desenvolvimento mais direto das linhas de código e, assim, debates para a resolução de problemas foram constantes. Apesar desse projeto ter similaridades com outros projetos de programação, haviam várias características sobre a propagação de sinais, sintaxe e detalhes do sistema que combinados geraram bastante dificuldades para quem não estava muito familiarizado com o nível RTL.

Por isso, foi fortemente sentida a importância da cooperação entre os participantes neste projeto.

Em outras palavras, a dinâmica do trabalho em equipe foi fundamental para a rapidez e sucesso na conclusão do trabalho.

Vale destacar também, que foi aplicada a sugestão do mentor para que não dividir o trabalho em múltiplo períodos para não prejudicar o fluxo do desenvolvimento. Isso ajudou ainda mais no trabalho mais rápido além de uma grande imersão dos participantes no problema.

Início do projeto

O início do projeto foi uma das partes difíceis. Era necessário assimilar vários tópicos apresentados em aula. Apesar haver bastante estudo de um microcomputador mais simplificado, isso não foi suficiente para evitar muitas dúvidas.

Mas foram aplicados inicialmente alguns módulos e blocos de código com as ideias já estudadas além de código desenvolvidos com os esclarecimentos de dúvidas do professor.

E, desta forma, o projeto que começou devagar, criou forma até a conclusão da primeira versão do “percurso dos dados”, mas, claro, cheio de erros.

Desenvolvimento do “Controle”

A parte interessante da atividade que envolveu um maior grau de liberdade para a criatividade.

Ao analisar o diagrama da proposta do trabalho, foi percebido que a cada diferente instrução podia um tipo diferente de ciclo de execução de tarefas pela máquina. Ou seja, foi o encontro com a com máquina de estados.

Foi decidido que alguns estados poderiam ser reaproveitados de estados pela máquina, principalmente a parte inicial para carregar e decodificar a nova instrução.

E, a cada nova instrução, a máquina reiniciava um novo ciclo desde o início.

Depois foi concluído que a máquina precisava assumir comportamentos diferentes para o tipo de instrução decodificada. E, então, ela foi bastante dividida nessa etapa.

Mas com basicamente foram classificadas por quatro categorias de instruções em relação à máquina, além do “HALT”:
Load, Store, operação que passam pela ula (inclusive o Move) e “branches”.

Então foi elaborado o “control” para a lista de instruções a seguir:

I_NOP, I_LOAD, I_STORE, I_MOVE, I_ADD, I_SUB, I_AND, I_OR, I_BRANCH, I_BZERO, I_BNZERO, I_BNEG, I_BNNEG, I_BOV, I_BNOV, I_HALT

A máquina de estados de control

O algoritmo:

A máquina busca a instrução

Decodifica

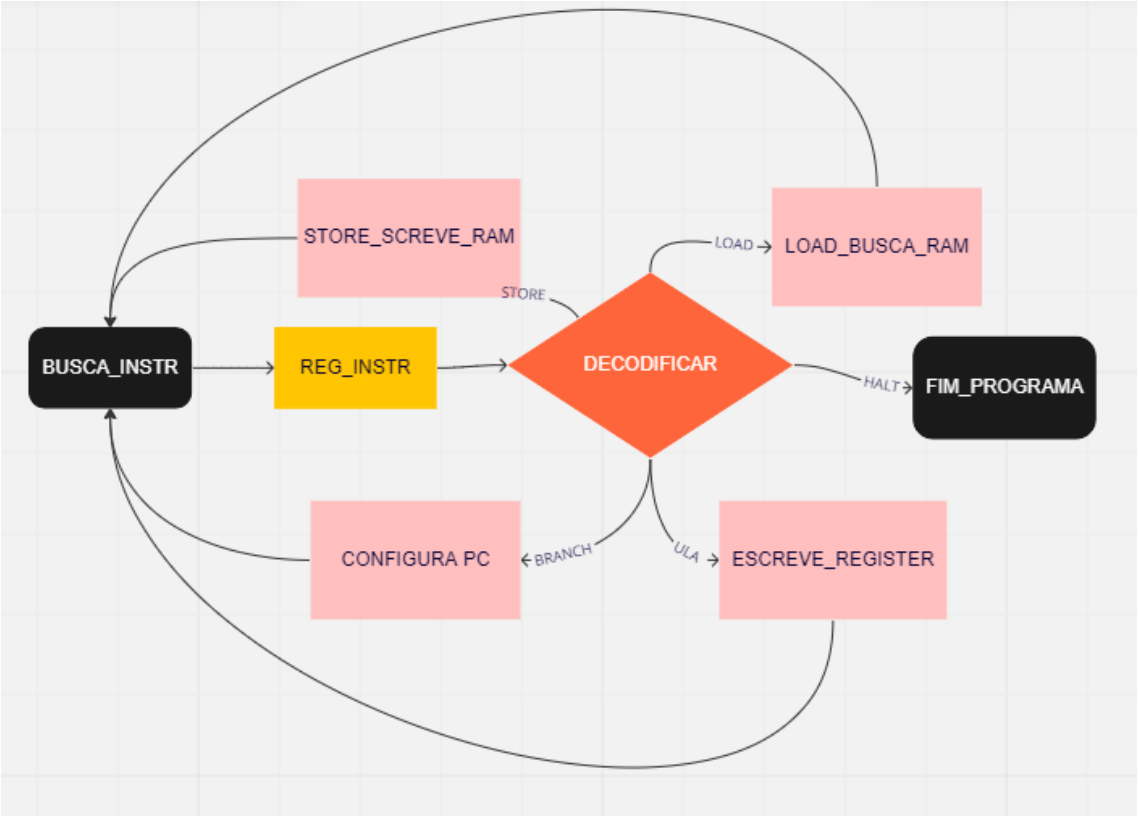
Executa o caminho do sinal, de acordo com a instrução

Repete até encontrar o HALT

Abaixo os detalhes do algoritmo estão mais detalhados:

BUSCA_INSTR	Início da máquina que dá partida à máquina. O sinal rst_n pode forçar a máquina a entrar no início
REG_INSTR,	Caminho inicial para uso de todas instruções para carregar a instrução
DECODIFICAR,	Etapa para descodificar o sinal que para a máquina saiba o que fazer com a instrução lida. É com essa informação que esse control depende para definir o comportamento. Depois da próxima etapa, se volta ao estado inicial para executar outra instrução
LOAD_BUSCA_RAM,	Instrução para o caso de LOAD que configura os fluxos para carregar valores da ram
STORE_ESCREVE_RAM,	I Instrução para o caso de STORE que configura os fluxos para salvar valores da ram
ESCREVE_REGISTER,	São operação que precisam passar pela ULA e ser armazenadas no banco de registradores. Esse estado é ramificado em outras 4 possibilidades de estados, dependentes da operação da instrução (ADD, SUB, OR, AND, MOVE)
FIM_PROGRAMA	Caminho que ativa o halt do microcomputador, reseta a máquina para encerrar tudo.

VERSÃO SIMPLIFICADA DA MÁQUINA APLICADA NO ALGORITMO:



Para a figura ficar mais enxuta, foram resumidas algumas codificações como (ADD, SUB, OR, AND e MOVE) por “ULA” e todas operações de BRANCH por “CONFIGURA PC”. Que são ramificações muito similares entre si, de acordo com o resultado da decodificação.

Escreve Register depende do sinal comando “case” em operation enquanto os branches faziam uma condicional de acordo com a flag relacionada com o Branch para atualizar o PC.

Essas ramificações foram simplificadas no código foram tratadas também com o comando “case” no descoded_instruction (DESCODIFICAR).

Ao aplicar tudo isso e correções de erros, a máquina fica pronta.

Principais erros e dificuldades:

Até fazer as primeiras configurações da máquina de controle não tem como verificar os erros. Então alguns erros da ULA apareceram só depois de testar os primeiros comandos de controle. Foi necessário tempo para se adaptar a usar o painel de sinais e muita paciência para olhar passo a passo da execução de cada instrução e comparar com o funcionamento do K&S da versão web e, com isso, foi preferido pelo time não utilizar o Perl.

A correção de erros foi feita principalmente pelo colaborador do momento por meio de dicas ou revezamento.

O mais difícil foi a correção do STORE, uma das primeiras máquinas a ser escritas.

Houve muito erro em esquecer que os valores de default da máquina eram ativados a cada troca de estado.

Por razão parecida era esquecido de voltar o estado inicial da máquina no fim de um caminho da máquina.

O Store precisou de um estado intermediário para funcionar corretamente configurando addr_sel = 1. IR e descoded_instruction ficavam com valores estranhos desconhecidos pela equipe.

Mas com a projeção do fluxo de sinais nos diagramas foram feitos os estados de máquina corretamente e acertadas as configurações dos sinais e, assim, esses erros foram aos poucos desfeitos.

O erro nos Branches foi tentar usar diretamente o sinal calculado pelo circuito combinacional. Ao olhar os sinais, foi percebida a necessidade do uso de um pequeno banco para as flags. O tipo de detalhe citado acima que dificultou o trabalho em relação à “programação comum”.

Há outro erro importante de ressaltar. Após acertar as falhas de cada instrução diferente, todas pareciam estar funcionando com microprogramas de teste. Mas, de modo geral houve muita dificuldade para executar um programa completo. Foi feito um oneroso trabalho de executar passo a passo da execução do microcomputador e comparar cada mudança de sinal com o K&S do HTML.

Em dado momento da análise, descobriu-se uma fala a instrução SUB. De primeira vista dava resultados estranhos. Ao alterar valores de teste com programas mais simples, depois de muito trabalho, chegou-se à conclusão que os operandos estavam invertidos.

De início era feito a – b com o complemento de 2 em b.
Então para corrigir foram invertidos os sinais e feito o complemento em a e a operação se mostrou correta.

E, por fim, havia a necessidade de fazer uma configuração na simulação que tinha como limite 1000ns. No começo se pensou que era algum estado incorreto da máquina, mas ao ajustar para 100ms a simulação acorreu bem.

Resultado
O microcomputador passou a achar o resultado correto de alguns programas como de calculo de fatorial e MDC além de pequenos programas de teste de desvio (caso positivo e negativo).

Na entrega do trabalho há os programas simulados na máquina (fatorial, MDC e teste) dentro da pasta sim

Em MDC.hex, a máquina carrega 100 (posição 29) e 30 (posição 30) e armazena o resultado 10 (posição 31).

Em FATORIAL.hex, a máquina carrega 5 (posição 31) e armazena o resultado 120 (posição 29).

E teste mais particular dos desvios:

Programa feito para testar as novas instruções, BOV, BNOV, BNNEG E BNZERO, onde a condição para fazer essas branchs é atingida por somas e subtrações simples e após cada instrução nova de desvio, há um branch 0 (identificado por 0100 em hexa) assim, caso alguma instrução nova não funcione, o programa retorna à primeira instrução e entra em loop infinito.

Conclusão

O trabalho foi uma ótima oportunidade de desenvolver o trabalho em equipe, caso que foi contra as expectativas de ambos participantes.

Os erros foram grandes gatilhos para olhar um projeto em nível RTL com a visão mais a fundo, minuciosa para consolidar o aprendizado da disciplina de Sistemas Digitais, especialmente sobre a máquina de estados e a linguagem verilog.

Foi notória a transformação de perspectiva sobre os microcontroladoes e praticado, além de tudo, a busca de informação (internet, professor, livro).

Sobretudo a importância da prática sobre analiso sobre o fluxo dos dados com o uso de diagramas.