Programa para Excelência em Microeletrônica

Módulo: Nivelamento

Matéria: Sistemas Digitais

Desenvolvimento de um Microcomputador

Aluno: Alfredo Rodrigo Sousa da Silva

Sumário

1 – Introdução..... 3
2 – Teoria..... 3
3 – Desenvolvimento do projeto..... 4
3.1 – Contador de programa.....4
3.2 – Registrador de instruções..... 7
3.3 – Acumulador A.....9
3.4 – Somador/subtrator.....9
3.5 – Registrador B.....11
3.6 – Registrador de saída.....12
3.7 – Controlador/sequencializador.....12
3.8 – Microcomputador completo.....13
4 – Conclusão.....17
5 – Referências.....18
6 – Anexos.....19

1. Introdução

Neste documento será abordado as etapas de desenvolvimento de um microprocessador utilizando a ferramenta Quartus Prime, da Altera. O desenvolvimento desse microprocessador foi realizado dividido em blocos, os quais serão explanados e detalhados neste documento.

2. Teoria

O microprocessador está presente em todos os microcomputadores. Ele é responsável por realizar cálculos e pela tomada de decisões em um computador. Coração da máquina e extremamente importante para o seu funcionamento, um microprocessador é constituído de três partes principais:

- Unidade de Lógica e Aritmética (ULA);
- Unidade de controle; e
- Unidade de memória.

Há também diversas unidades de entrada e de saída, por onde entram os dados a serem processados, e por onde saem os dados ao final do processamento.

O processador é composto por diversos blocos (Figura 1) responsáveis por realizar tarefas distintas. São eles o contador de programa, o registrador de endereço de memória (REM), memória RAM, registrador de instruções, controlador/sequenciador, acumulador A, somador/subtrator, registrador B e registrador de saída. Todos esses blocos são interconectados pelo barramento W e gerenciados pelo controlador/sequenciador através de doze variáveis que ditam o funcionamento do microprocessador e garantem o sincronismo entre os blocos.

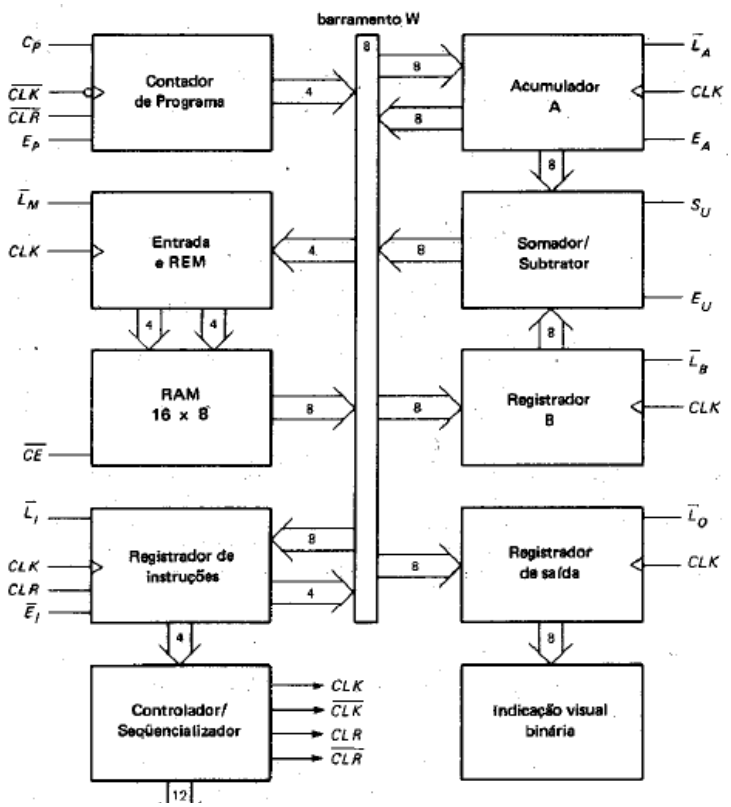


Figura 1 – Diagrama de blocos do microprocessador

A seguir, veremos, detalhadamente, o processo de construção deste microprocessador utilizando a ferramenta Quartus Prime, da Altera.

3. Desenvolvimento do Projeto

A construção do microprocessador foi feita por blocos, como pode ser visto na Figura 1. Veremos a seguir como foi feito o desenvolvimento de cada bloco e como foi feita a junção de todos os blocos para formar o microprocessador.

3.1. Contador de programa

O contador de programa é um contador de quatro bits responsável por endereçar a posição de memória a ser acessada para executar uma instrução. Composto por quatro *flip-flops* interligados de forma assíncrona e por portas lógicas do tipo *tri-state* em suas saídas, o contador de programa conta em ordem crescente e em binário de 0000 (0_{10}) até 1111 (15_{10}).

Utilizando o software Quartus, da Altera, foi criado um novo projeto, intitulado de *program_counter*. Logo em seguida, um novo arquivo de esquemático de circuito foi criado com o mesmo nome, e foram adicionados todos os componentes necessários. Os primeiros componentes a serem adicionados foram os flip-flops do tipo JK, como pode ser visto na Figura 2:

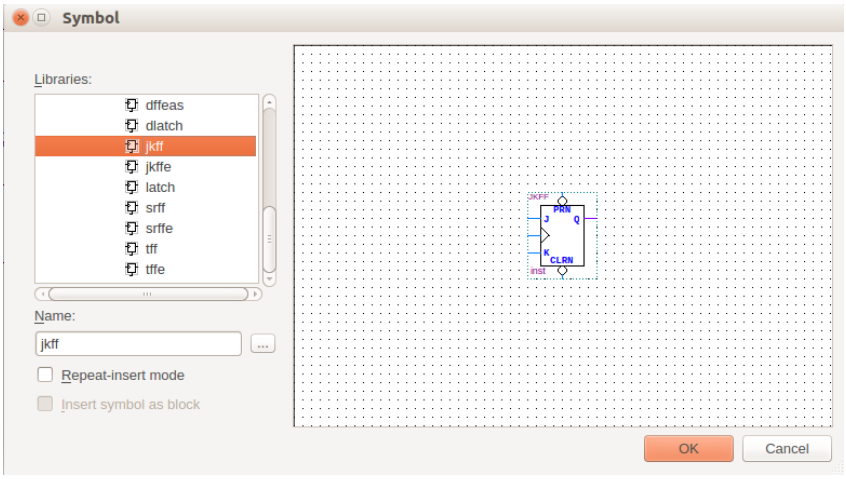


Figura 2 – Flip-flop do tipo JK

Para o contador de programa foram utilizados quatro *flip-flops* JK interligados de forma assíncrona. Como no software Quartus não foi encontrado o flip-flop JK com o *clock* sensível a borda de descida, foram utilizadas portas not nas entradas dos mesmos para realizar tal adaptação. Como também no esquemático original não é utilizado o pino de PRN ($\overline{Preset}$), neste projeto estes pinos foram interligados e conectados ao VCC, para que fosse garantido sempre o nível lógico baixo no mesmo.

Logo em seguida, foram feitas todas as ligações dos pinos restantes. Neste esquema, os pinos J e K de todos os *flip-flops* foram interligados. Nesta configuração, poderia

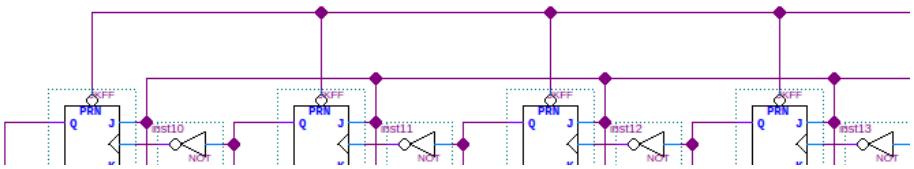


Figura 3 – Interligação dos *flip-flops*

também ser utilizado o *flip-flop* tipo T. Com as ligações obtivemos as entradas C_p, $\overline{CLK}$ e $\overline{CLR}$. As ligações foram feitas da forma como pode ser vista na Figura 3:

Como eles foram utilizados de forma assíncrona, o sinal de saída que saia do pino Q dos *flip-flops* foram interligados também ao *clock* do seguinte *flip-flop*.

Depois de realizadas as devidas ligações, foram adicionadas portas lógicas do tipo *tri-state*[1] nas saídas do circuito, com suas entradas adicionais *enable* interligadas entre si, formando a entrada E_p. Neste circuito, as portas *tri-state* são utilizadas para habilitar ou desabilitar a passagem do sinal advindo da saída do *flip-flop*. Ao término dessas ligações, o circuito ficou da seguinte forma (Figura 4):

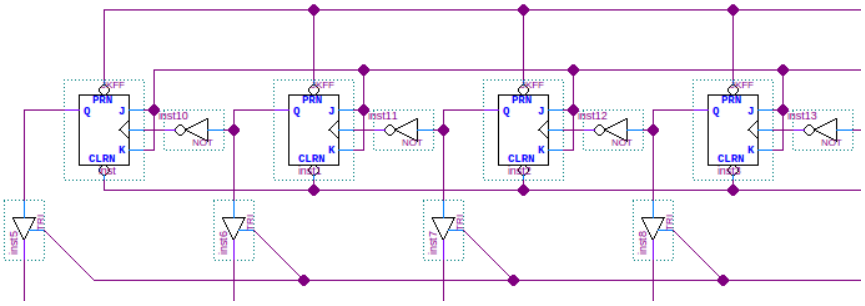


Figura 4 – Ligações das portas *tri-state*

Em seguida, foram adicionados pinos de *input* e de *output* nos fios de entrada e de saída do circuito. As saídas das *tri-states*, da direita para a esquerda, foram ligadas a pinos de *outputs* que foram ligados posteriormente, no projeto de topo, ao *W bus* (barramento W) aos fios A, B, C e D do barramento, respectivamente, onde A representa o bit menos significativo da palavra (esse esquema pode ser melhor entendido ao observar a Figura 5). Já nos fios referentes as instruções C_p, $\overline{CLK}$, $\overline{CLR}$ e E_p foram ligados pinos de *input*, os quais foram ligados posteriormente, também no projeto de topo, às saídas correspondentes do controlador/sequencializador (este bloco será explicado mais a frente).

Todos os pinos foram devidamente instanciados, para que a organização fosse mantida e as ligações se fizessem de forma mais fácil no projeto de topo. Neste momento, todas as ligações do contador de programa foram finalizadas. As ligações e os pinos podem ser observados na Figura 5:

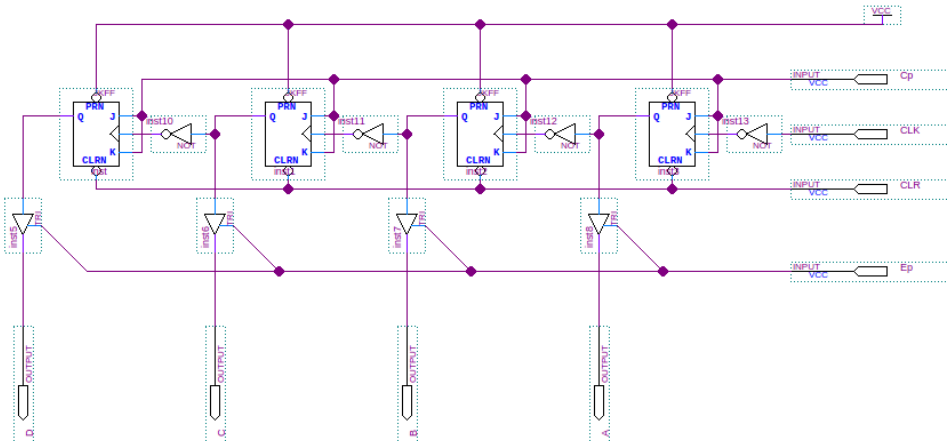


Figura 5 – Contador de programa finalizado

[1] A porta *tri-state* recebe esse nome dado a razão de tal função permitir três possíveis estados na saída do circuito: nível lógico alto, nível lógico baixo e alta-impedância (Z).

Após terminadas as ligações, o processo seguinte foi a compilação do esquemático. Após a compilação, é possível saber se todas as ligações e configurações do circuito são válidas. Também são gerados vários relatórios acerca da compilação, nos quais o usuário pode consultar e observar cada etapa da compilação com detalhes. O resultado da compilação é essencial para que ocorra a posterior simulação, pois caso o circuito não compile, a simulação não pode de ser efetuada.

No caso do contador de programa, sua compilação apontou 11 alertas, (Figura 6), mas nenhum erro:

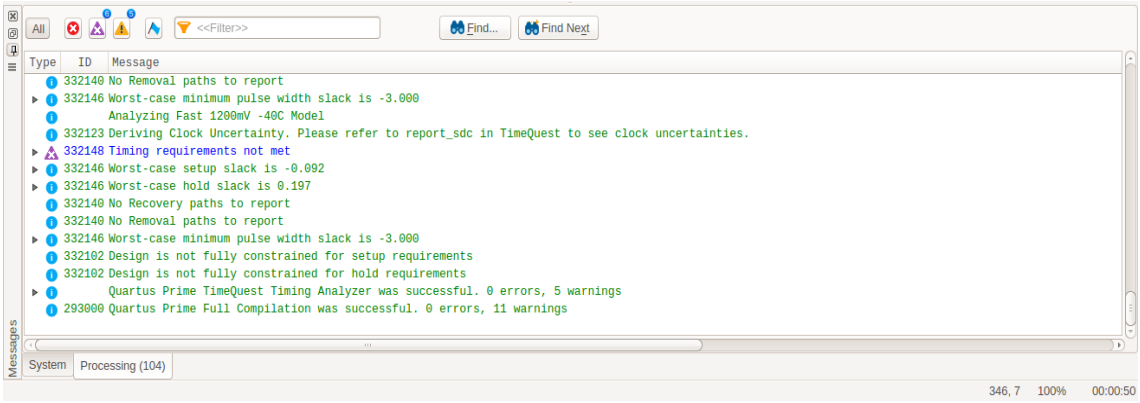


Figura 6 – Relatório gerado pela compilação do contador de programa

Em seguida, foi criado um arquivo do tipo vwf para a efetuação da simulação do circuito. Neste arquivo, foram adicionados todos os pinos de entrada e de saída do circuito, e foram feitas as devidas configurações de tempo de execução e de sinal nos referidos pinos. O arquivo criado pode ser visto na Figura 7 e sua simulação na Figura 8:

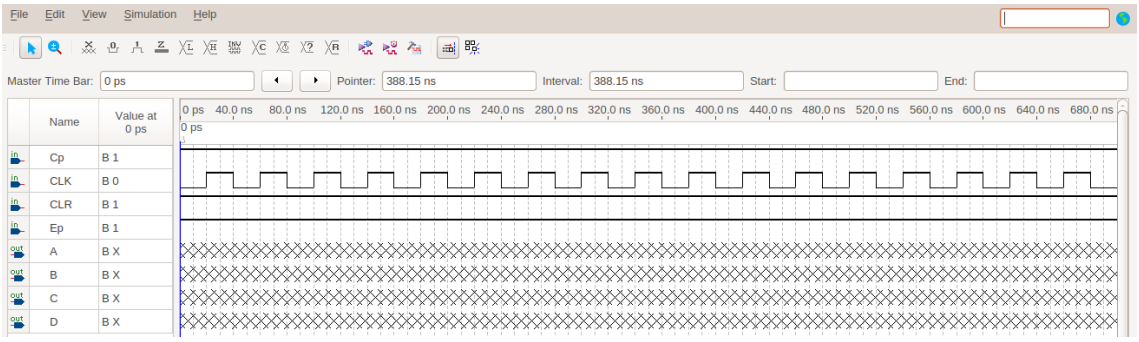


Figura 7 – Arquivo de simulação vwf

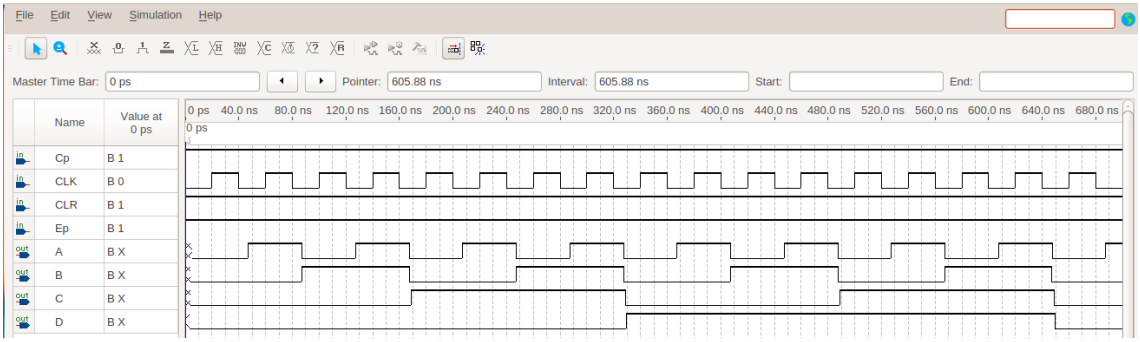


Figura 8 - Resultado da simulação

No resultado da simulação, observou-se que, nos pinos de saída A, B, C e D do contador de programa, a cada descida de borda do *clock* o contador incrementava 1, contando assim de 0000 até 1111, reiniciando o contagem ao término de um ciclo de 16 pulsos de *clock*. Com essa observação foi chegado à conclusão de que o contador de programa estava finalizado e funcionando perfeitamente. Vale salientar que este foi o único bloco simulado durante o seu desenvolvimento. O motivo para não terem sido feitas simulações nos blocos seguintes foi o de que eles dependiam de outras variáveis e de resultados advindos de outros blocos. Portanto, suas simulações só foram feitas em conjunto, quando todos estavam interligados, no projeto de topo.

Um arquivo de símbolo foi gerado para o contador de programa, tornando possível a sua utilização no projeto de topo, tendo em vista que, nele, basta apenas importar os arquivos referentes aos blocos e efetuar as devidas ligações. Seu símbolo gerado foi o que pode ser observado na Figura 9:

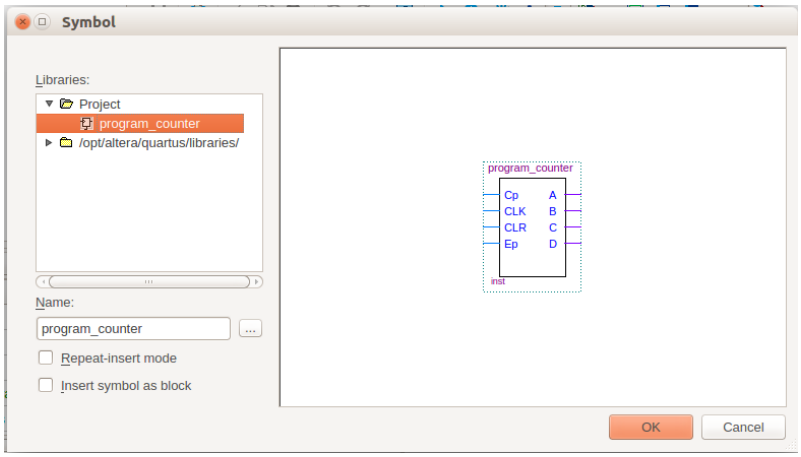


Figura 9 – Símbolo do contador de programa

3.2. Registrador de instruções

O próximo bloco construído foi o registrador de instruções. Para ele foi criado um novo projeto com o nome *instruction_register*, no qual foram realizadas todas as etapas do bloco anterior de forma semelhante.

O registrador de instruções é constituído de dois circuitos integrados (CI's) cuja referência é 74LS173. Sua função é guardar, por um período de tempo, a instrução endereçada na memória e que foi disponibilizada no *W bus*.

A primeira etapa do desenvolvimento deste bloco constituiu-se na pesquisa pelo datasheet[2] do CI utilizado no registrador de instruções, a fim de se obter o seu diagrama lógico, ou seja, todas as portas lógicas, ligações e demais características internas do mesmo. O diagrama lógico utilizado encontra-se disponível no "ANEXO A" deste documento.

O que continha no diagrama lógico do CI 74LS173 foi reproduzido no Quartus. Ao término da montagem, o 74LS173 ficou da seguinte forma (Figura 10), onde a esquerda temos as entradas do circuito, e a direita as saídas:

[2] Datasheet (que na tradução literal significa folha de dados) é um documento que apresenta de forma resumida, todos os dados e características técnicas de um equipamento ou produto. Neste caso, do circuito integrado (CI) 74LS173.

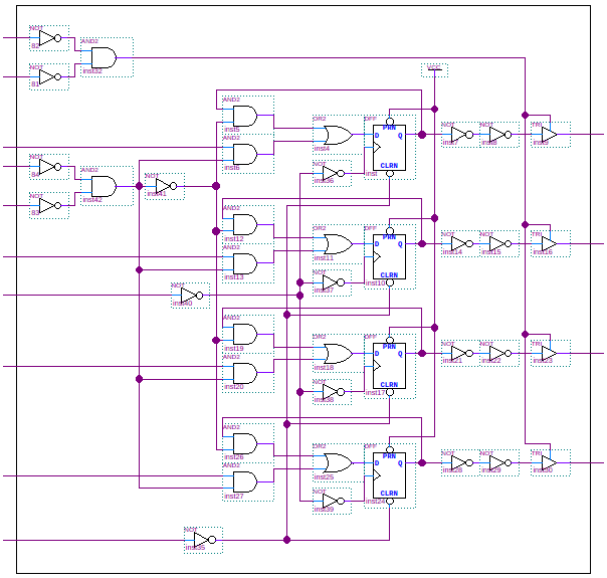


Figura 10 – Circuito integrado 74LS173

Após montado, todo o diagrama lógico do CI foi duplicado (colado) no mesmo arquivo, já que o registrador de instruções possui dois 74LS173. Em seguida, todas as devidas interligações foram feitas.

Neste circuito, temos entradas e saídas que vão para o *W bus*. Devido a isto, essas entradas e saídas foram interligadas, e nelas foram postos pinos bidirecionais, ou seja, que podem ser usados tanto como pinos de entrada como pinos de saída. Também foram postos os pinos de entrada das instruções $\bar{L}_i$, $\bar{E}_i$ e CLK. Este circuito conta com as saídas I_4, I_5, I_6 e I_7 no segundo CI. O esquemático do registrador de instruções completo pode ser visto na Figura 11:

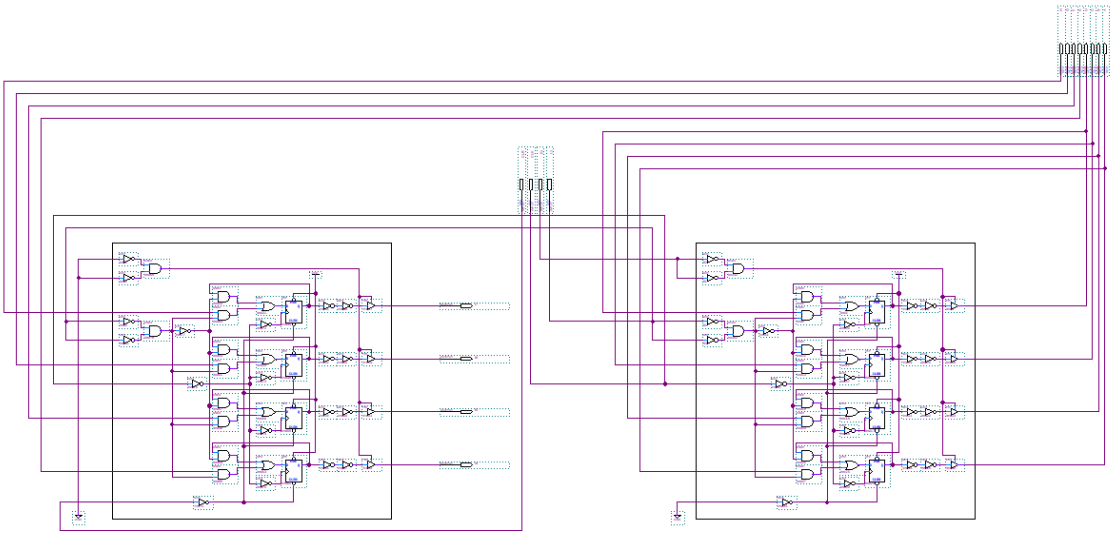


Figura 11 – Registrador de instruções

Logo após o término, foi feita a compilação (de modo semelhante a compilação que foi feita no bloco anterior) e nenhum erro foi detectado. Como explicado anteriormente, este e os blocos seguintes não foram simulados ao término do seu desenvolvimento, pois eles dependiam de outras variáveis e de resultados advindos de outros blocos.

Ao término, foi gerado um arquivo de símbolo semelhante ao gerado no contador de programa e que se encontra ilustrado na Figura 9.

3.3. Acumulador A

O acumulador A é um registrador de oito bits e tem como finalidade armazenar os resultados intermediários calculados pelo processador.

O desenvolvimento do acumulador A foi extremamente semelhante ao desenvolvimento do registrador de instruções. As diferenças se encontram nas saídas dos CI's 74LS173, nas quais estas encontram-se ligadas ao *W bus* por meio de *tri-states*, e também possuem pinos de saída que são direcionados ao seguinte bloco (somador/subtrator), além das entradas de instruções, que passam a ser agora $\bar{L}_a$, CLK e E_a.

Este projeto recebeu o nome de *accumulator_a* e ficou da seguinte forma (Figura 12):

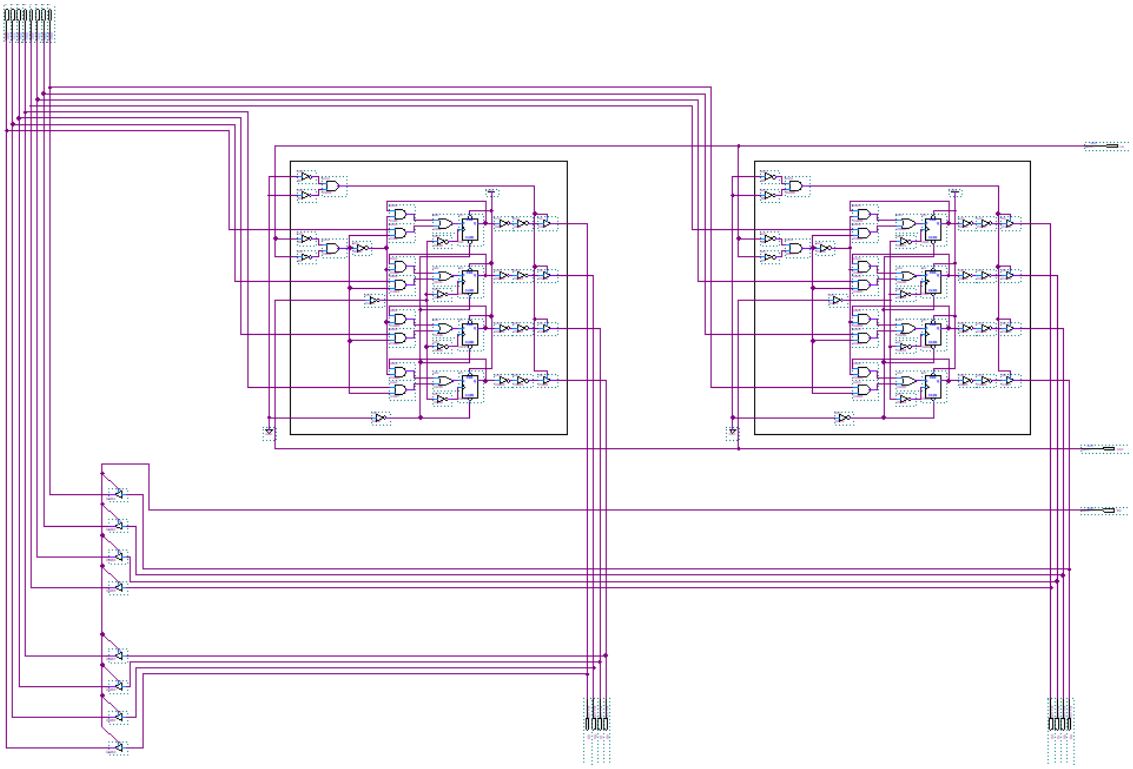


Figura 12 – Acumulador A

O projeto foi compilado e seu arquivo de símbolo foi gerado.

3.4. Somador/subtrator

Este foi um dos blocos mais complexos do processador.

Ele é constituído de dois CI's 74LS83 que realizam a função de soma e de subtração. Ele ainda possui oito portas xor responsáveis, em conjunto com o *carry in* do primeiro CI, pela seleção da operação a ser realizada pelo circuito (soma ou subtração).

O projeto criado recebeu o nome de *adder_and_subtractor*.

Semelhante ao registrador de instruções, para o somador/subtrator foi necessário realizar uma pesquisa para observar o diagrama lógico do CI utilizado neste bloco e poder reproduzi-lo. O diagrama lógico utilizado encontra-se disponível no "ANEXO B" deste documento.

O CI ficou da seguinte forma (Figura 13):

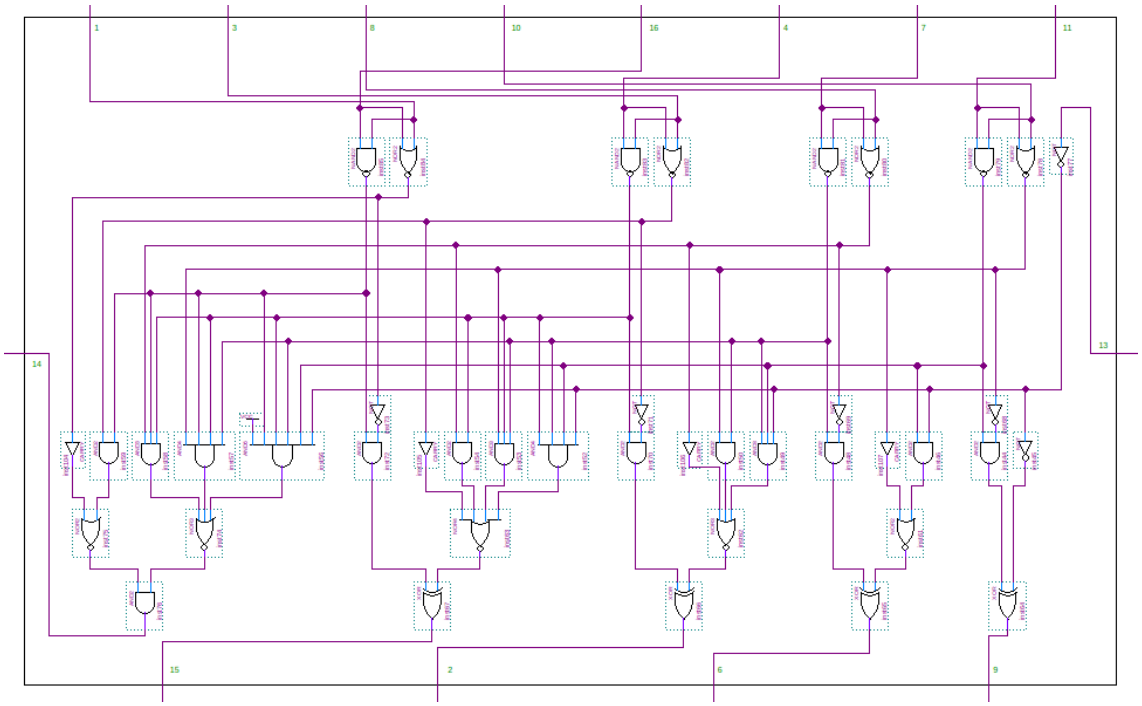
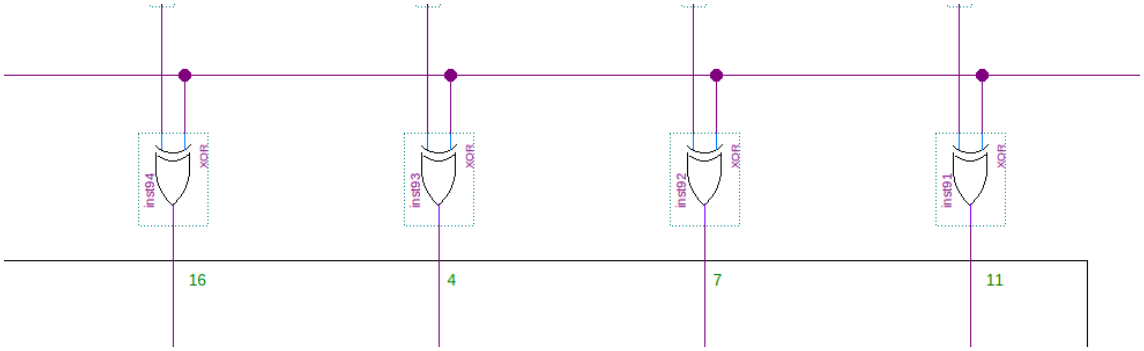
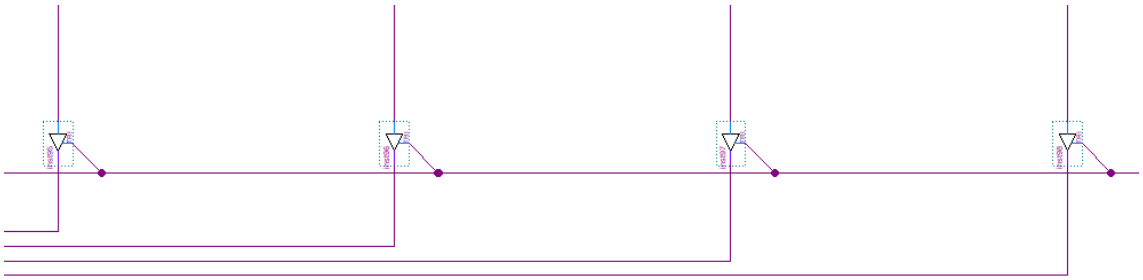


Figura 13 – Circuito integrado 74LS83

Uma vez reproduzido, o circuito integrado foi duplicado (colado) no mesmo arquivo, e as devidas ligações foram realizadas. Foram acrescentadas também as portas xor nas entradas de uma das palavras de cada CI, e portas *tri-state* em todas as saídas dos dois CI's, como pode ser observado na Figura 14a e 14b:



(a) Portas xor nas entradas do CI



(b) Portas *tri-state* nas saídas do CI

Figura 14 – Entradas e saídas dos CI's 74LS83

Após terminadas todas as ligações, foram inseridos os pinos de entrada e de saída do circuito, atentando para a nomenclatura dos mesmos, tendo em vista que este bloco se comunicará diretamente com mais dois blocos, fazendo-se necessária a organização para melhor entendimento na montagem final do projeto de topo. Juntamente com estes pinos, foram postos também os pinos referentes as funções S_u e E_u. Ao final, o circuito foi compilado e não foi encontrado nenhum erro, e seu símbolo foi gerado. O circuito completo ficou da forma como pode ser vista na Figura 15:

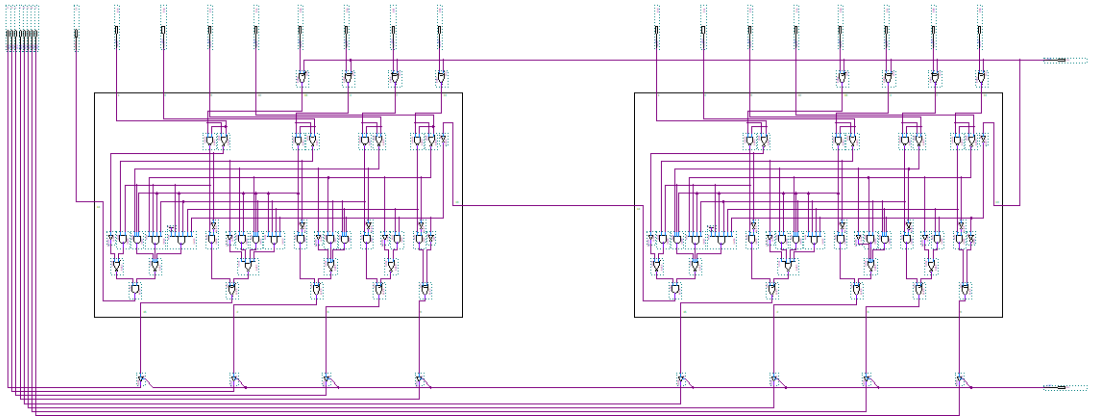


Figura 15 – Somador/subtrator

3.5. Registrador B

O registrador B é um registrador de oito bits responsável por fornecer o número em operações aritméticas.

O desenvolvimento deste bloco foi bastante semelhante ao desenvolvimento do acumulador A. As diferenças estão nas saídas, que neste não são utilizadas portas *tri-states*, e nas entradas de instruções, que passam a ser agora $\overline{L}_b$ e CLK.

O projeto recebeu o nome de *b_register* e todo o circuito ficou da seguinte forma (Figura 16):

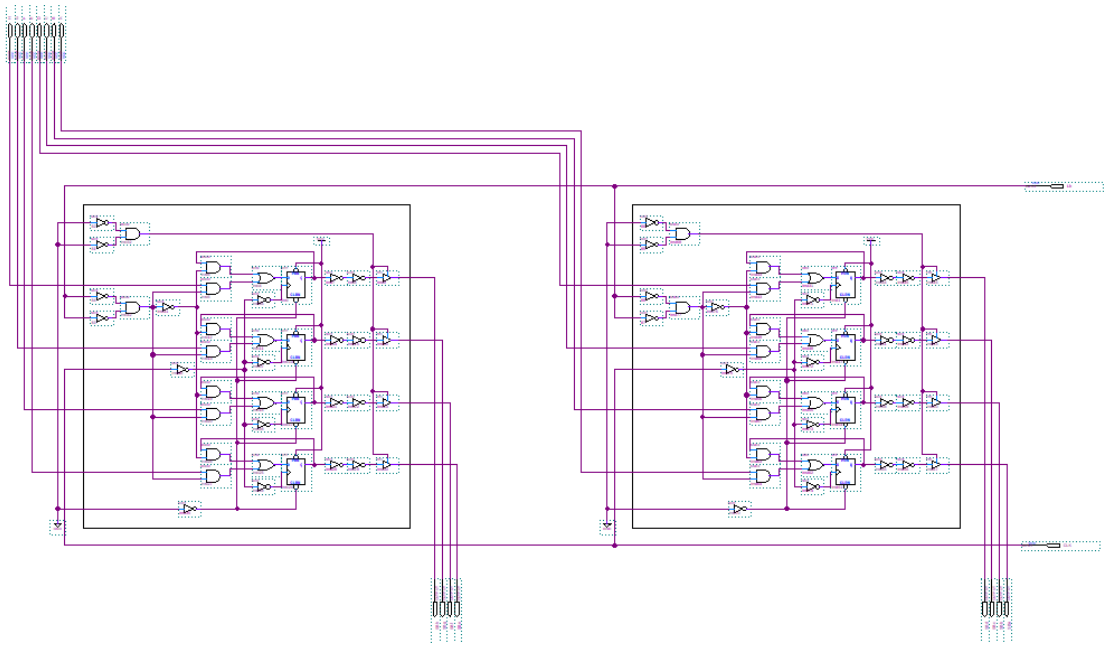


Figura 16 – Registrador B

O registrador B foi compilado e nenhum erro foi encontrado. Em seguida, seu símbolo foi gerado.

3.6. Registrador de saída

Sua função é transmitir os dados processados pelo processador para o mundo externo.

O registrador de saída também é extremamente semelhante ao acumulador A, e seu desenvolvimento foi praticamente o mesmo. As diferenças estão, mais uma vez, nas saídas, pois estas não são mais direcionadas a nenhum outro bloco, sendo elas as saídas finais de todo o processador, e nas entradas de instruções, que passam a ser agora $\overline{L}_o$ e CLK.

Abaixo, na Figura 17, podemos ver como ficou todo o circuito, intitulado de *output_register*:

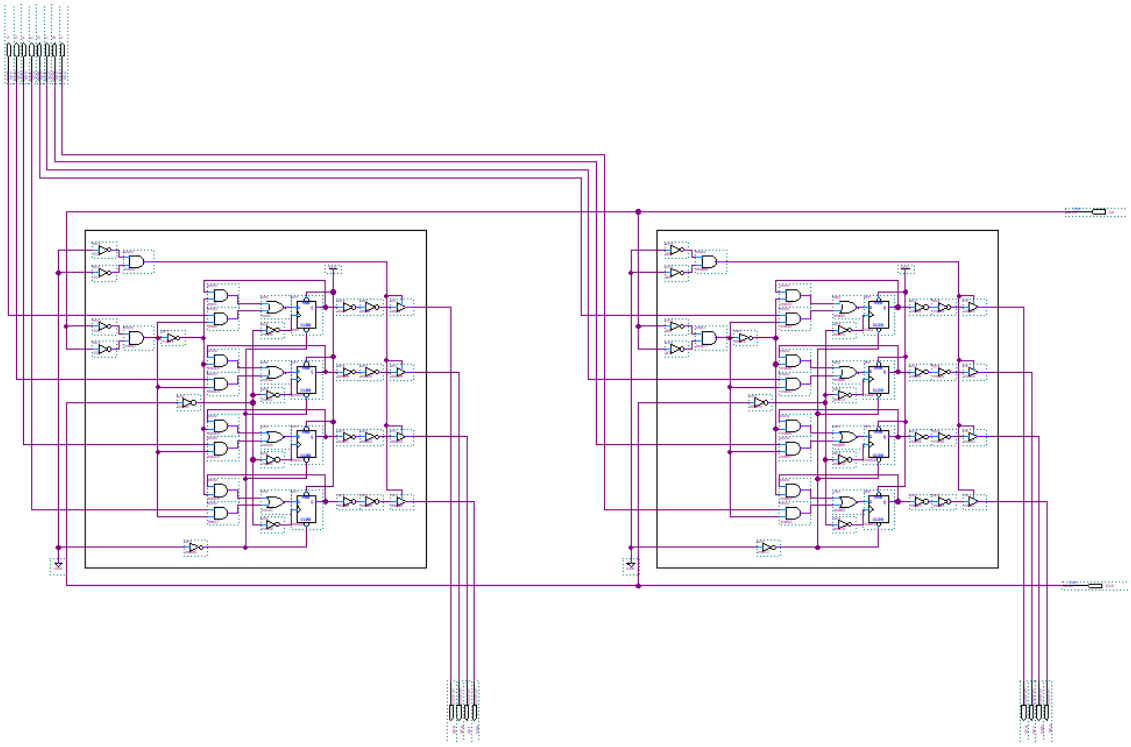


Figura 17 – Registrador de saída

Novamente, o projeto foi compilado e nenhum erro foi encontrado. Após a compilação, seu arquivo de símbolo foi gerado.

3.7. Controlador/sequencializador

Este bloco é responsável por gerar os doze sinais que controlam todos os blocos do processador, sincronizando suas operações no devido tempo, além de gerar o *clock*, o $\overline{clock}$, o *clear* e o $\overline{clear}$. Os doze sinais ou instruções geradas por ele são C_p, E_p, $\overline{L}_m$, $\overline{CE}$, $\overline{L}_i$, $\overline{E}_i$, $\overline{L}_a$, E_a, S_u, E_u, $\overline{L}_b$ e $\overline{L}_o$. O circuito montado, que recebeu o nome de *sequencer*, pode ser visto na Figura 18. Ele é dividido em três partes que juntas geram esses sinais (ver Figura 18). São elas o contador em anel (em vermelho), o decodificador de instruções (em verde) e a matriz de controle (em azul):

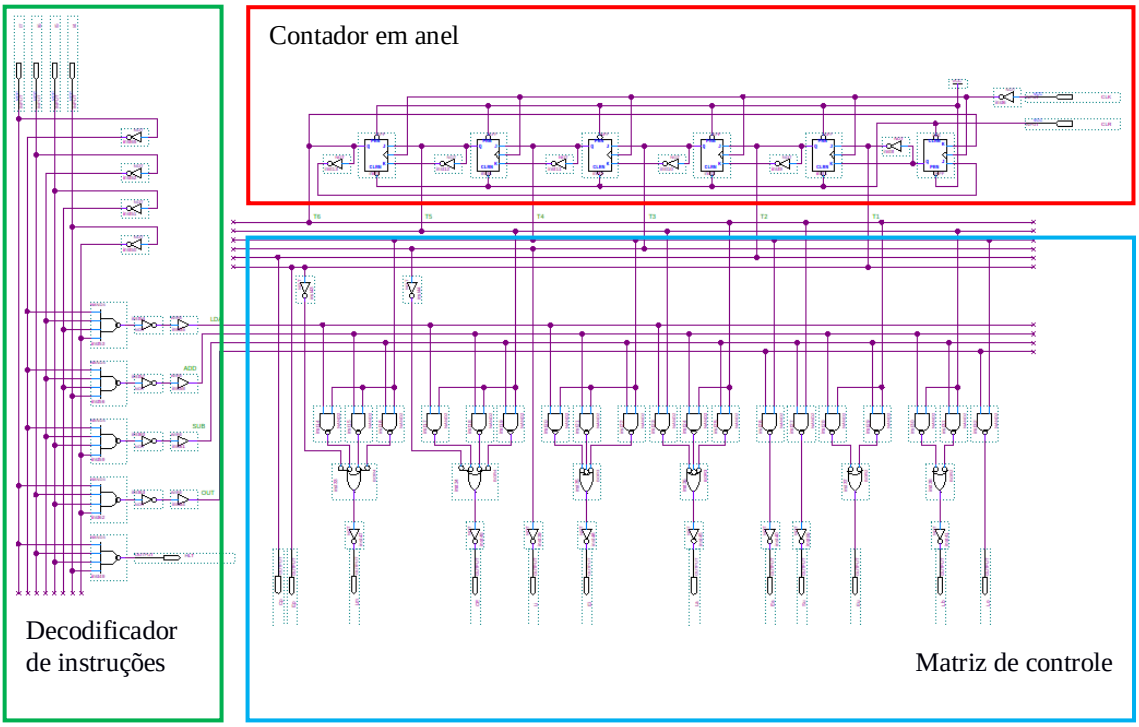


Figura 18 – Controlador/sequencializador

3.8. Microprocessador completo

Após o término da montagem de todos os blocos, foi criado um novo projeto intitulado de *microprocessor*, no qual foram introduzidos todos os blocos anteriormente criados. Ao criar esse projeto, os arquivos dos blocos anteriores foram importados e, quando um novo arquivo de esquemático foi criado, todos os símbolos dos blocos foram adicionados ao novo arquivo.

Pode-se notar que, de acordo com a Figura 1, os únicos blocos que não foram montados foram a memória RAM e o registrador de endereço de memória (REM). O motivo disto se deu ao fato de o Quartus possuir uma funcionalidade de criação de bloco de memória, no qual o desenvolvedor pode escolher, dentre diversas funcionalidades, a que mais se aplica ao seu projeto. Com isso, o bloco de memória RAM foi inserido no projeto através dessa funcionalidade. Já o bloco do REM não foi necessário sua implementação, devido ao fato da memória já realizar, também, a função que seria exercida por ele: manter o valor do endereço apontado pelo contador de programa por um período de tempo.

Na figura seguinte, podemos ver o bloco de memória já instalado no microprocessador (Figura 19):

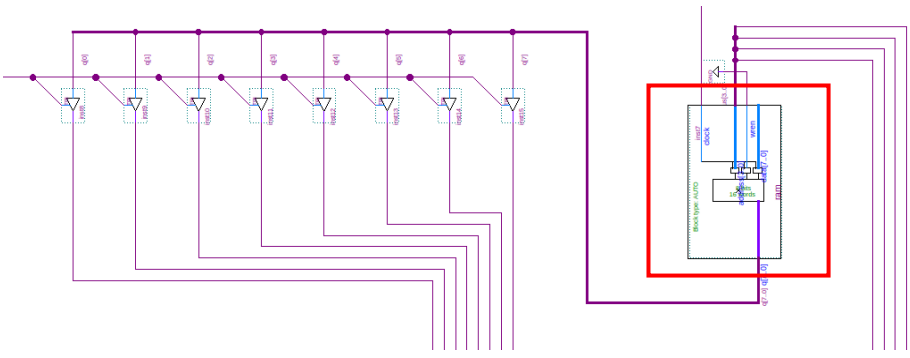


Figura 19 – Memória RAM (destacada em vermelho)

Para criar a memória RAM, foi utilizada a opção “RAM: 1-PORT”, encontrada navegado nos menus da seção IP Catalog, mais precisamente em Library > Basic Functions > On Chip Memory (Figura 20):

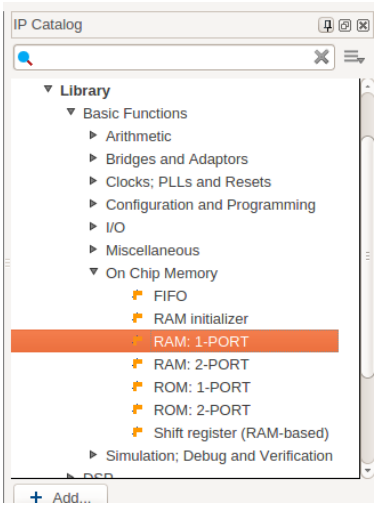


Figura 20 – Onde se encontra a memória RAM

Após clicar na opção, uma janela com um assistente de configuração é aberta, e o desenvolvedor pode configurar diversas características da memória, como pode ser visto na Figura 21:

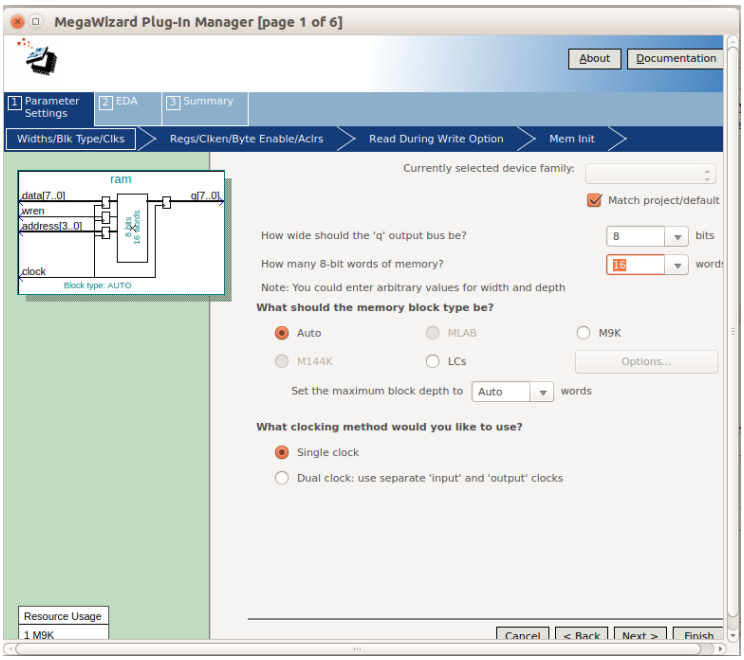


Figura 21 – Assistente de configuração da memória RAM

Em suas configurações é possível, inclusive, inserir um arquivo hexadecimal com as instruções a serem realizadas pela memória.

Foi disponibilizado pelo coach Antônio Agripino um arquivo desse tipo que foi utilizado como instrução para a memória. Neste arquivo continham instruções de adição e de subtração, mais respectivamente, uma rotina que faria com que o microprocessador realizasse a seguinte operação matemática:

$$(20 - 16 + 24 - 32)_{10}$$

Após a configuração, a memória RAM foi importada ao projeto e instalada conforme mostra a Figura 19.

Ao término da montagem de todos os blocos, o circuito completo do microprocessador ficou da forma como pode ser vista na Figura 22:

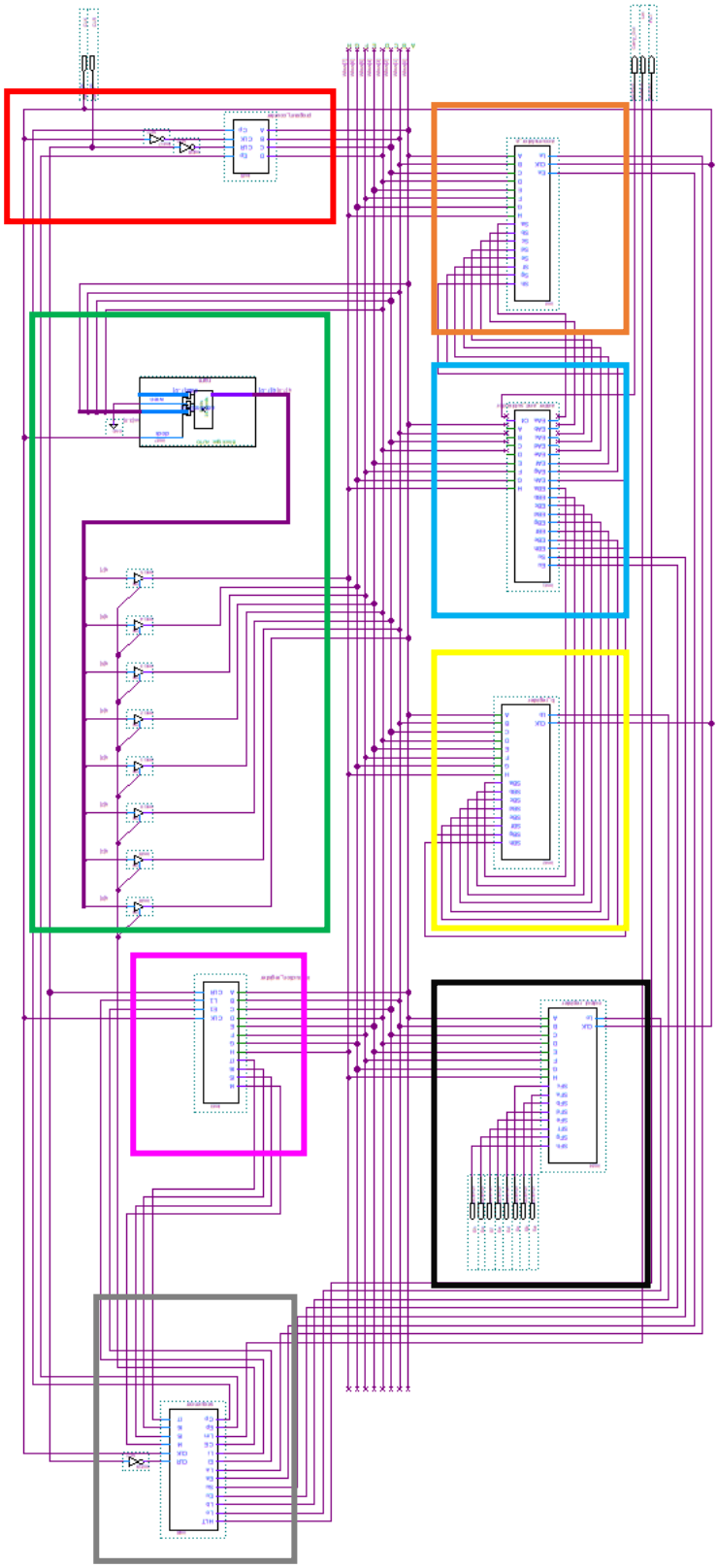


Figura 22 – Microprocessador completo

É possível ver, na imagem anterior (Figura 22) as diferentes partes do microprocessador. Vemos destacado de:

- Vermelho – Contador de programa.
- Verde – Memória RAM com saídas *tri-states*;
- Rosa – Registrador de instruções;
- Cinza – Controlador/sequencializador;
- Laranja – Acumulador A;
- Azul – Somador/subtrator;
- Amarelo – Registrador B; e
- Preto – Registrador de saída.

Ao término de toda a montagem, o processador foi compilado, e nenhum erro foi encontrado (Figura 23):

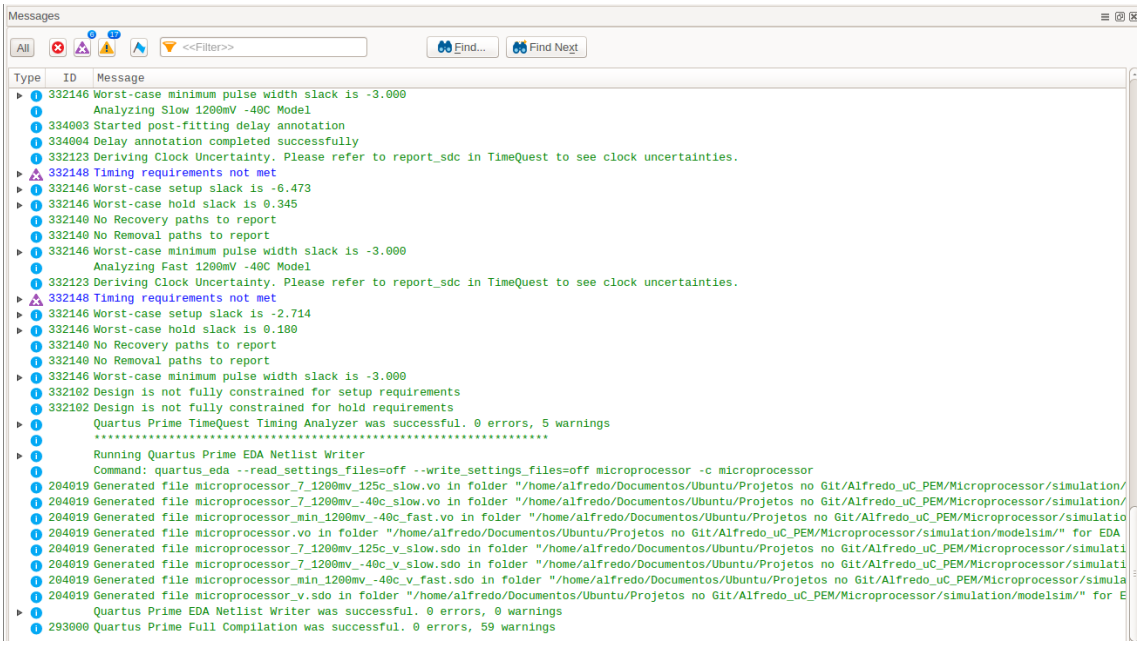


Figura 23 – Relatório gerado pela compilação do microprocessador

Em seguida, foi criado um arquivo de simulação vwf, onde foram inseridos os pinos de entrada e de saída do processador. As formas de onda definidas podem ser vistas na Figura 24, e o resultado da simulação na Figura 25:

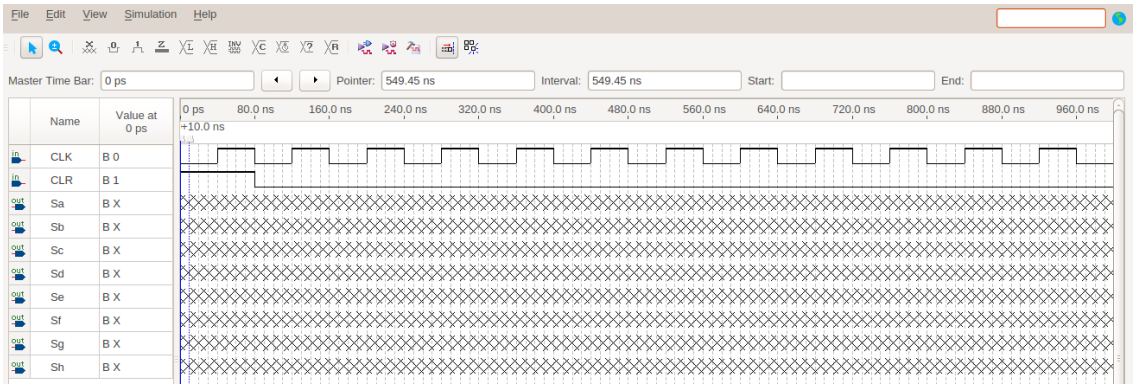


Figura 24 – Arquivo de simulação vwf

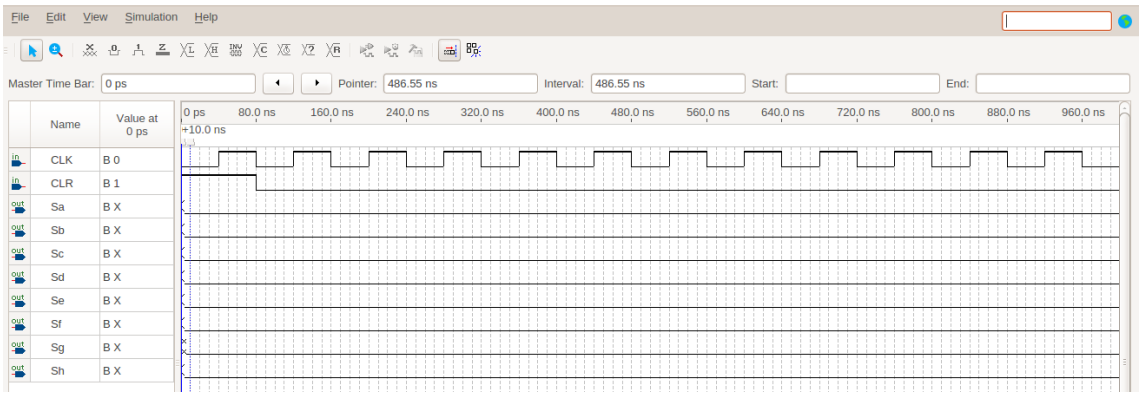


Figura 25 – Resultado da simulação do microprocessador

Ao término da simulação, pode ser visto que o microprocessador não se comportou da forma esperada. Como dito anteriormente, o microprocessador deveria realizar a seguinte equação matemática:

$$(20 - 16 + 24 - 32)_{10}$$

O resultado que deveria ser obtido ao final da simulação era $1111\ 1100_{c2}$, que corresponde a -4_{10}. Porém, o resultado obtido foi $0000\ 0000_2$, que corresponde a 0_{10}.

Com isso, o projeto do microprocessador foi finalizado, apesar de incompleto.

4. Conclusões

Desde o início do desenvolvimento até o término – ou quase – o aprendizado foi imenso. No final do projeto, a conclusão que se chegou foi que um microprocessador, apesar de complexo, pode ser simples se visto com outros olhos. Além disso, houve o aprendizado da utilização da ferramenta Quartus, da Altera. Uma ferramenta poderosa e que pode auxiliar bastante um desenvolvedor durante o seu projeto.

Apesar de o microprocessador não ter funcionado como esperado no final, seu desenvolvimento trouxe aprendizados importantíssimos. Ficou esclarecido como se dá o funcionamento de um processador, além de se ter tido uma noção da arquitetura de computadores/processadores. Mais especificamente, foi aprendido que um processador possui ciclos de execução, e que ele é dividido em vários blocos responsáveis por atividades diferentes, em que cada bloco é executado em sincronia com os demais, garantindo assim a confiabilidade dos resultados obtidos.

5. Referências

ALLDATASHEET.COM**. 74LS173 Datasheet (PDF) - Texas Instruments.** Disponível em: <http://www.alldatasheet.com/datasheet-pdf/pdf/27388/TI/74LS173.html> Acesso em: 27 de Outubro de 2016.

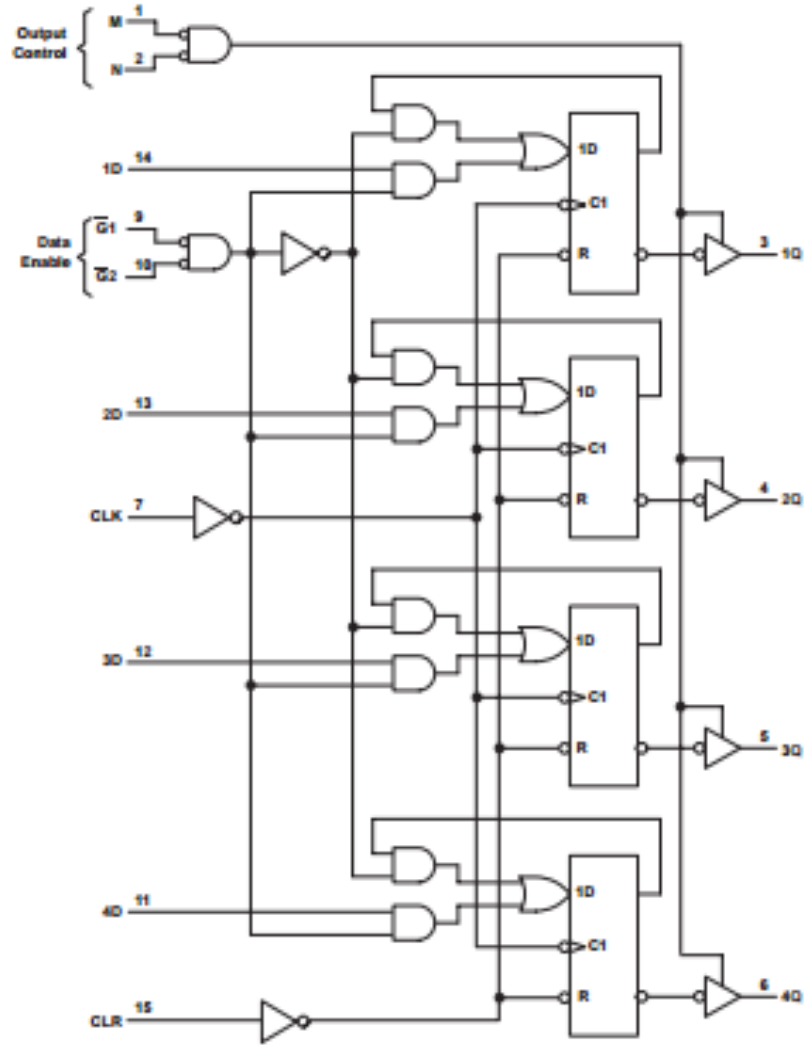
FUTURLEC.COM. **technical Information - Fairchild Semiconductor 74LS83 Datasheet.** Disponível em: <http://www.futurlec.com/74LS/74LS83.shtml> Acesso em: 31 de Outubro de 2016.

FUTURLEC.COM. **Technical Information - Texas Instruments 74LS173 Datasheet.** Disponível em: <http://www.futurlec.com/74LS/74LS173.shtml> Acesso em: 28 de Outubro de 2016.

6. Anexos

ANEXO A – Diagrama lógico do CI 74LS173

logic diagram (positive logic)



Pin numbers shown are for D, J, N, and W packages.

DM74LS83A

Truth Table

Inputs				Outputs					
				When C0 = L			When C0 = H		
						When C2 = L			When C2 = H
A1 A3	B1 B3	A2 A4	B2 B4	Σ1 Σ3	Σ2 Σ4	C2 C4	Σ1 Σ3	Σ2 Σ4	C2 C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

H = HIGH Level, L = LOW Level

Input conditions at A1, B1, A2, B2, and C0 are used to determine outputs Σ1 and Σ2 and the value of the internal carry C2. The values at C2, A3, B3, A4, and B4 are then used to determine outputs Σ3, Σ4, and C4.

Logic Diagram