Relatório do laboratório 11 - Unidade de Controle do Neander

Nome: Arthur Ferreira Ely

Cartão: 338434

Data: 19/03/2023

Número da sala: 103

Foi realizado, neste laboratório, a implementação da parte de controle do Neander. A parte de controle serve para mandar sinais para o resto do computador, para que assim possamos deixa-lo completamente funcional. A metodologia seguida foi a do livro do professor Weber, fazendo toda a lógica utilizando um contador de três bits e dois decodificadores. Os Flip-Flops utilizados foram do tipo D.

Células da unidade de controle (UC)

Para o desenvolvimento da UC, foi adotado a técnica *bottom-up,* fazendo pequenos componentes lógicos e os juntando na UC completa posteriormente. A parte de controle é formada por:

- Decodificador das instruções;
- Contador de três bits;
- Decodificador dos sinais de controle.

Decodificador das instruções

Serve para dizer qual instrução está vindo da memória conforme os 4 bits mais significativos do valor que está no registrador de instruções (RI). A seguir, a figura 1 mostra o diagrama esquemático do decodificador, baseado na figura 2, do livro do professor Weber. As entradas que não resultam em nenhumas instruções foram conectadas à *ground* (GND).

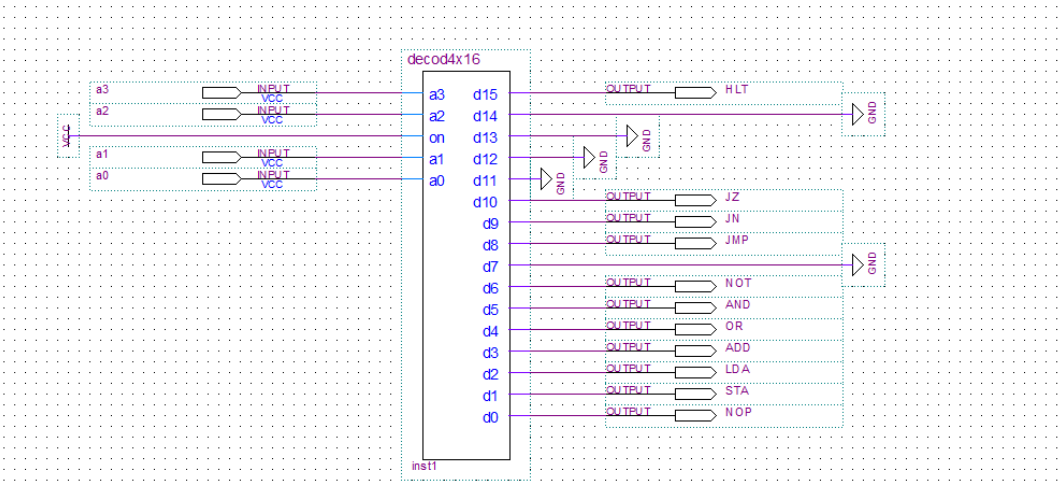

Figura 1: decodificador das instruções do Neander.

10.2 Fluxo de dados

O conjunto de instruções do NEANDER fornece mais detalhes sobre o uso e as interconexões necessárias entre estes elementos:

Código	Instrução	Execução
0000	NOP	nenhuma operação
0001	STA end	MEM(end) ← AC
0010	LDA end	AC← MEM(end)
0011	ADD end	AC← MEM(end) + AC
0100	OR end	AC← MEM(end) OR AC
0101	AND end	AC← MEM(end) AND AC
0110	NOT	AC← NOT AC
1000	JMP end	PC← end
1001	JN end	IF N=1 THEN PC ← end
1010	JZ end	IF Z=1 THEN PC ← end
1111	HLT	término de execução - (halt)

Figura 2: tabela com os códigos das instruções do Neander.

A seguir, a simulação desse decodificador:

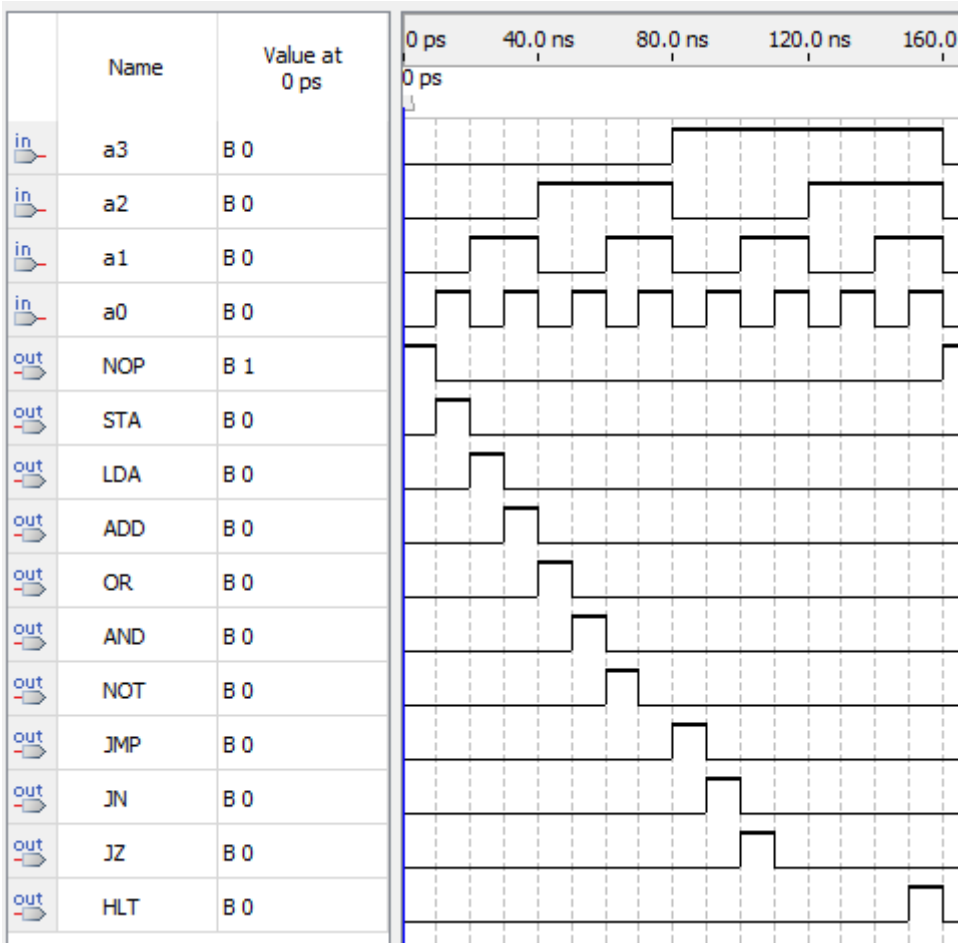

Figura 3: simulação do decodificador das instruções.

Contador de três bits

Serve para temporizar toda a lógica de controle. A cada borda de subida do clock, o contador vai ser somado em 1. Cada tempo fará algo diferente, seguindo o modelo do livro do professor Weber. A seguir, a figura 1 mostra o diagrama esquemático do contador. Um adicional desse contador de três bits é o *goto_t0,* que serve para reiniciar o contador em 0 e é uma das saídas do decodificador de sinais de controle.

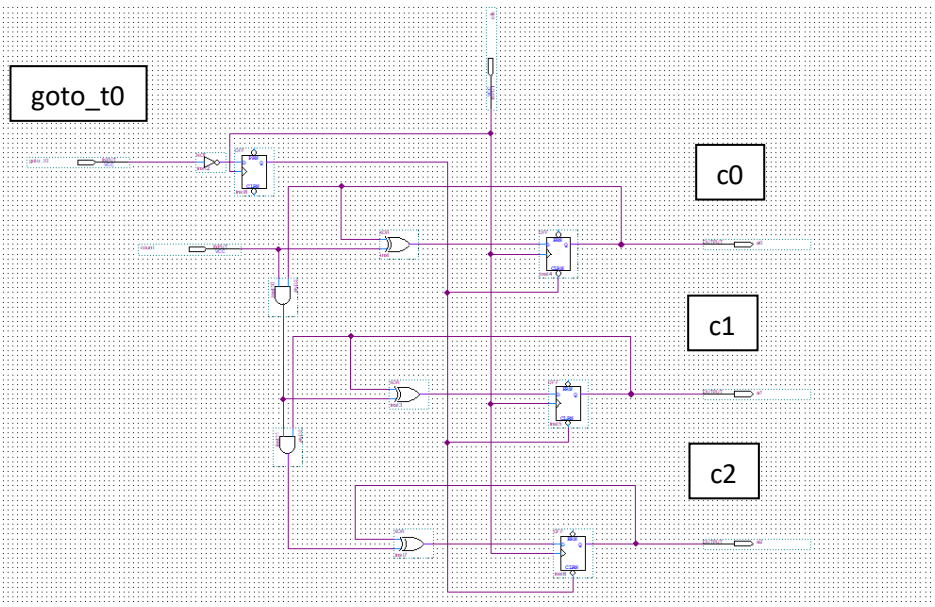

Figura 4: decodificador das instruções do Neander.

A seguir, a simulação desse contador:

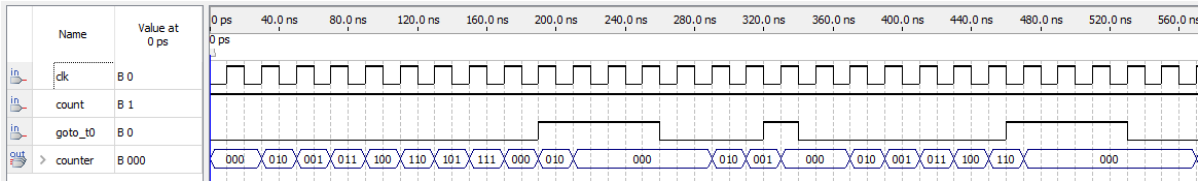

Figura 5: simulação do contador de três bits, com o *goto_t0*.

Decodificador dos sinais de controle

Serve para dizer qual sinal de controle deverá ser ativado no determinado ciclo de clock. Recebe como parâmetros o decodificador de instruções do Neander, o contador de três bits e, adicionalmente, os sinais de N e Z, que indicam quando o operando a ULA é negativo ou zero, respectivamente. A figura 6 a seguir mostra esse decodificador com as suas entradas.

Note que há um componente adicional entre o contador e o decodificador de sinais. Foi optado por colocar um decodificador 3x8 para o valor do contador, para trabalhar com menos fios dentro do decodificador de sinais.

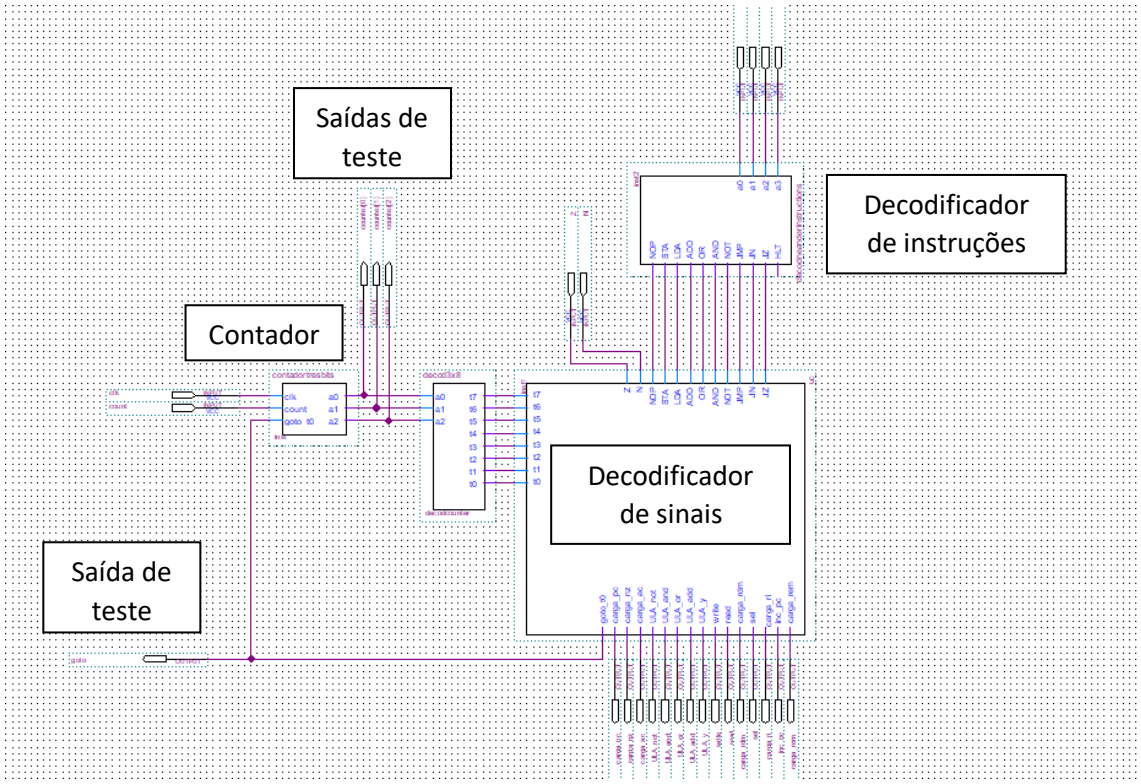

Figura 6: estrutura da unidade de controle completa

A seguir, na figura 7, o diagrama esquemático do decodificador de sinais. Esse diagrama foi feito com base no livro do professor Weber, como pode-se ver na figura 8. Além disso, algumas simplificações nos sinais foram feitas para diminuir mais a quantidade de fios no circuito. Essas simplificações estarão destacadas nas imagens.

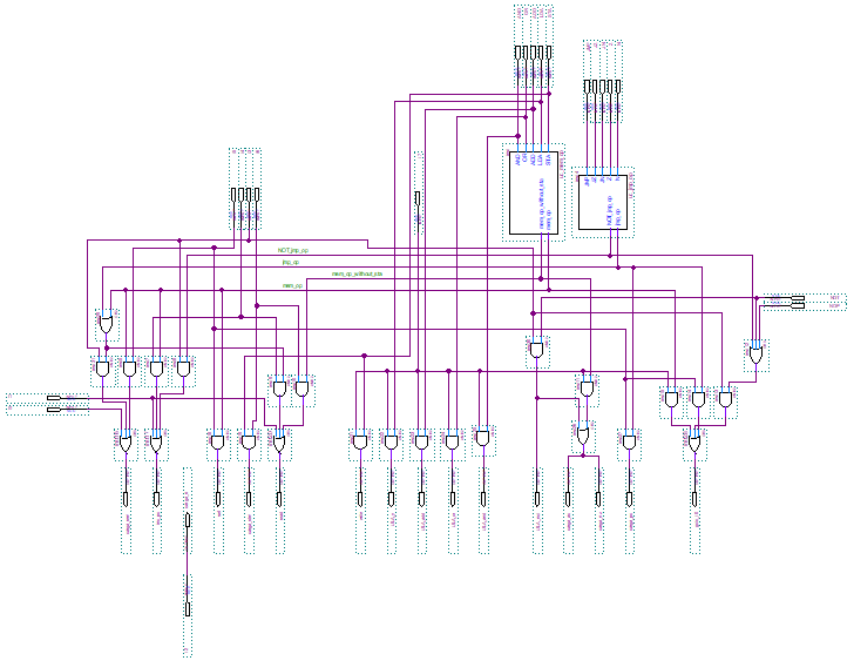

Figura 7: diagrama esquemático do decodificador de instruções do Neander

Para o sequenciamento de todas as instruções são necessários 8 tempos distintos, numerados de t0 a t7. Os três primeiros, t0, t1 e t2, servem para a fase de busca das instruções. Os demais (t3 a t7) são necessários para a fase de execução. Observe-se que, quando termina a execução de uma instrução, existe um sinal de controle explícito para voltar ao tempo t0 (goto t0). Das Tabelas 10.3 e 10.4 podem ser derivadas as equações booleanas para cada um dos sinais de controle. Estas equações são indicadas a seguir, em forma não necessariamente otimizada:

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)

Figura 8: funções para cada sinais de controle.

A seguir, as simplificações no circuito. As simplificações estão nos dois pequenos decodificadores presentes na parte superior do circuito, conforme mostra a figura 9.

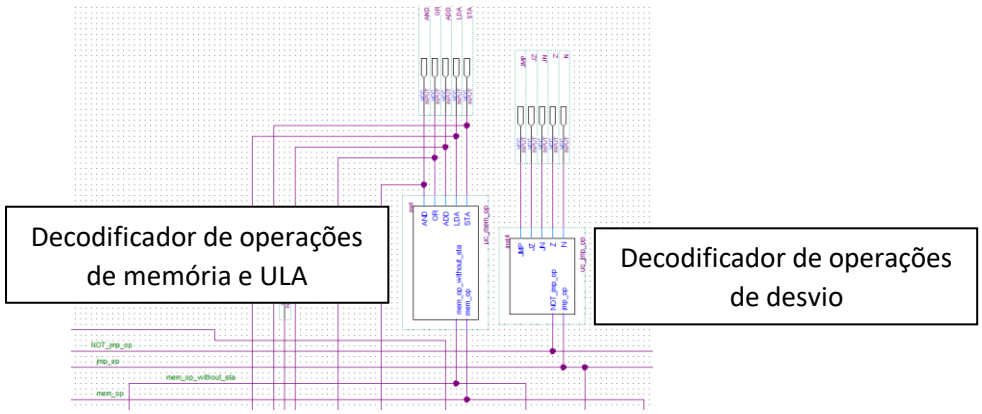

Figura 9: parte superior do decodificador de sinais citada.

Nela, há dois decodificadores. Um relacionado as operações que lidam com a memória e a ULA (não estando presente a função NOT), e outro relacionado as operações de desvio.

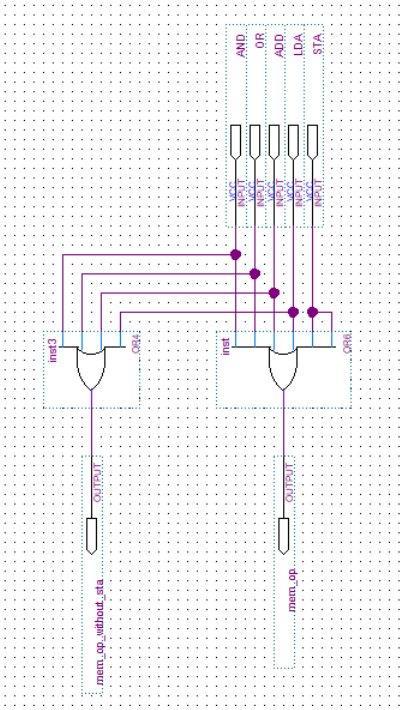

Figura 10: decodificador das operações de memória e ULA.

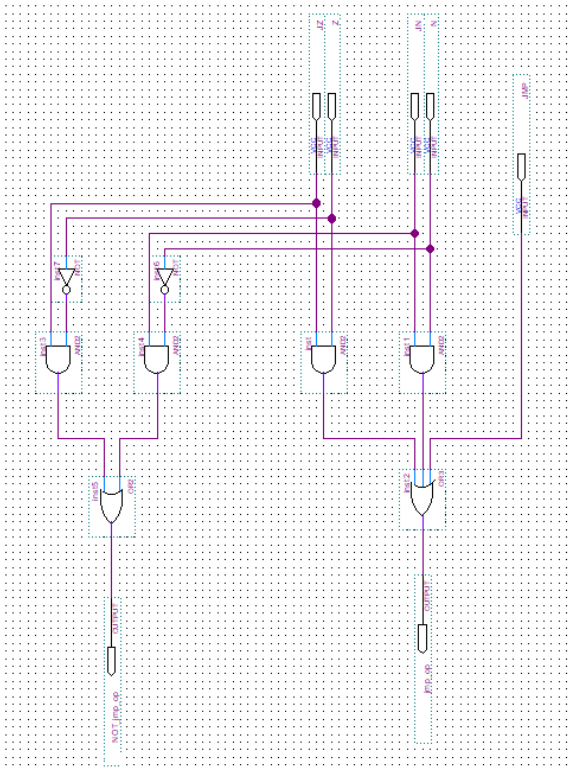

Figura 11: decodificador das operações de desvio.

Conclusão:

A seguir, toda a simulação da unidade de controle, na figura 12:

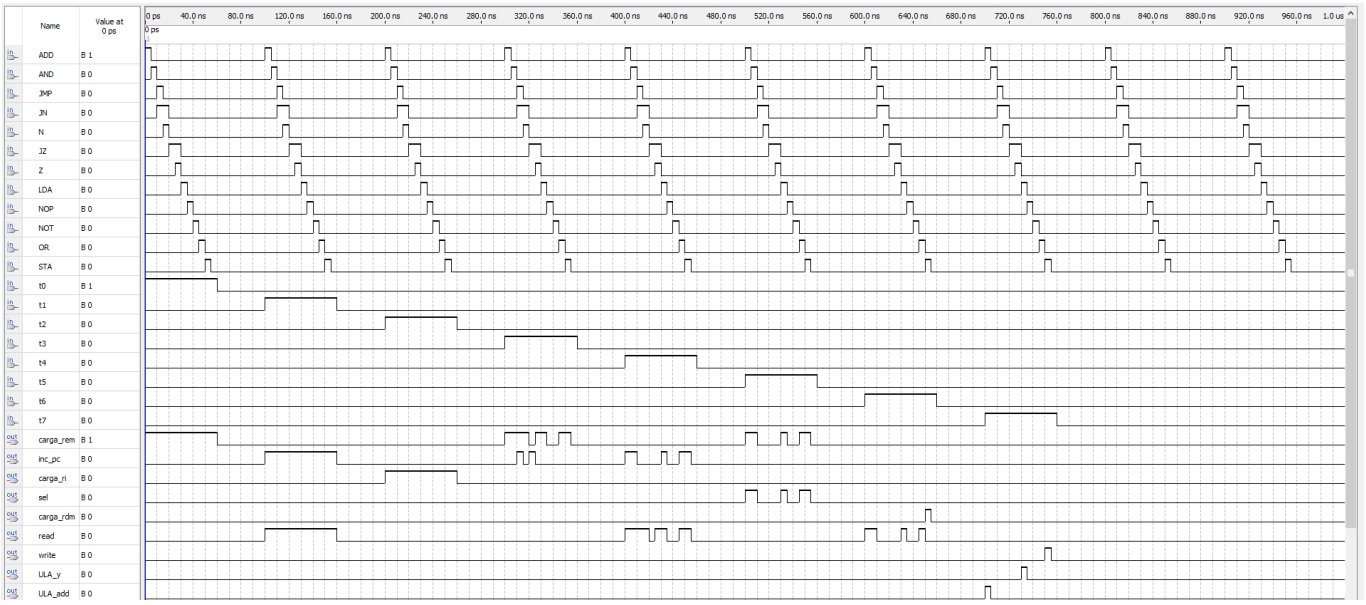

Figura 12: simulação completa da UC.

Lembrando que todos os sinais de controle foram baseados na tabela 10.2 do livro do professor Weber, como mostra a figura 13 a seguir:

10.3 Sinais de controle

Todos os sinais de controle da Figura 10.1 são gerados nos instantes de tempo adequados pela Unidade de Controle. A Tabela 10.2 mostra a equivalência entre as transferências realizadas e os sinais de controle a serem ativados.

Transferência	Sinais de controle
REM ← PC	sel=0, carga REM
PC ← PC + 1	incrementa PC
RI ← RDM	carga RI
REM ← RDM	sel =1, carga REM
RDM ← AC	carga RDM
AC ← RDM; Atualiza N e Z	UAL(Y), carga AC, carga NZ
AC ← AC + RDM; Atualiza N e Z	UAL(ADD), carga AC, carga NZ
AC ← AC or RDM; Atualiza N e Z	UAL(OR), carga AC, carga NZ
AC ← AC and RDM; Atualiza N e Z	UAL(AND), carga AC, carga NZ
AC ← not(AC); Atualiza N e Z	UAL(NOT), carga AC, carga NZ
PC ← RDM	carga PC

Tabela 10.2 - Sinais de controle para as transferências do NEANDER

Figura 13: tabela de sinais de controle do Neander

Com isso, temos nossa parte de controle feita e funcionando. A seguir, duas figuras que mostram onde essa parte se encaixa no *datapath* do Neander.

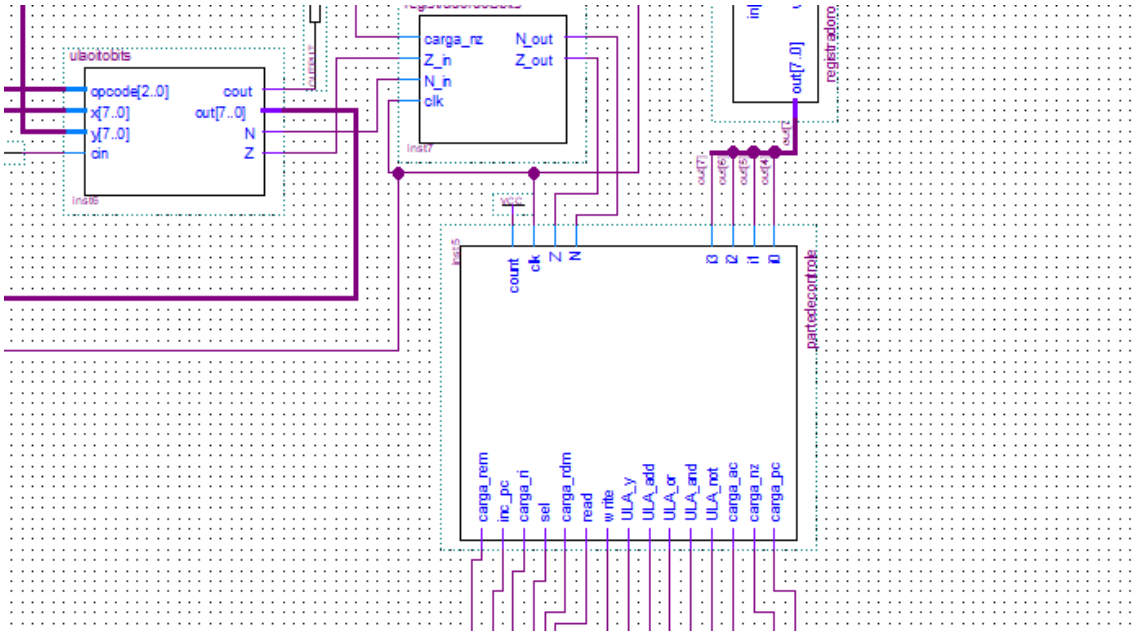

Figura 13: parte de controle conectada ao resto do Neander.

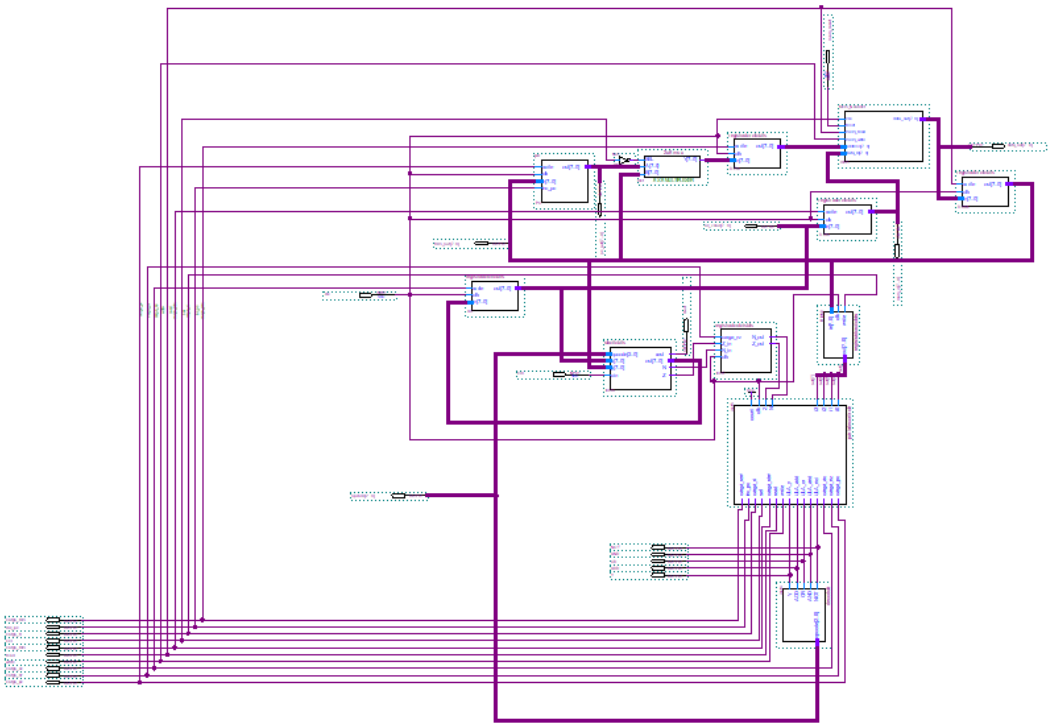

Figura 14: Neander e seu *datapath* completo e funcional.

Dificuldade encontrada no desenvolvimento do decodificador de sinais:

Ao desenvolver a parte de controle, quando o teste do decodificador de sinais estava sendo feito, foi percebido que quando o sinal de *goto_t0* era ativado, o contador retornava imediatamente ao *t0,* não realizando as outras operações do tempo do estado atual do contador.

Após muito tempo, foi constado que a entrada *goto_t0* não estava conectada a um Flip-Flop do tipo D (DFF) antes de ser conectado a porta CLRN do DFF. Então, quando o *goto_t0* era ativado, imediatamente o contador era zerado e não dava tempo de ativar os outros sinais, visto que estes seriam ativados na próxima subida do Clock. Bastou conectar a entrada em um DFF e o decodificador de sinais funcionou corretamente.

Isso mostrou o quão importante é a barreira temporal que os Flip-Flops proporcionam e o quão estável um circuito síncrono é, porque tudo vai acontecer exatamente como se espera, uma instrução após a outra e um sinal sendo ativado após o outro.