Universidade de Brasília
Departamento de Ciência da Computação

Laboratório 2

- CPU RISC-V UNICICLO -

Alunos:

Dyesi Montagner - Matrícula: 212008544
Gabriel Castro - Matrícula: 202066571
Lucas Santana - Matrícula: 211028097
Maria Vitória Monteiro - Matrícula: 222035652
Jackson Marques - Matrícula: 17001367

Grupo B3 – OAC Unificado – 2025/1
Link do repositório GitHub: Repositório Grupo B3
Playlist no youtube com os testes em vídeo: Playlist Grupo B3

1) (10.0) Implemente o processador Uniciclo com ISA Reduzida com as instruções: add, sub, and, or, slt, lw, sw, beq, jal, e ainda as instruções jalr e addi.

1.1) (1.0) Escreva um programa de1.s que teste a corretude da implementação de todas as 9 + 2 instruções e teste no Rars. Dica: Use o registrador t0 para visualizar resultados!

Tabela 1: Tabela de Testes.

Instrução	Código	Entrada	Saída (t0)	Endereço Memória (gp)
LW	`lw t0, 0(gp)`	0x12345678	0x12345678	0x10010000
ADDI	`addi t0, zero, 1`	-	1	-
ADD	`add t0, t0, t1`	t0=1, t1=5	6	-
SUB	`sub t0, t0, t1`	t0=6, t1=5	1	-
AND	`and t0, t0, t1`	t0=0xF0, t1=0x0F	0x00	-
OR	`or t0, t0, t1`	t0=0x00, t1=0x0F	0x0F	-
SLT	`slt t0, t0, t1`	t0=5, t1=10	1	-
SW	`sw t0, 4(gp)`	t0=0x55	-	0x10010004 → 0x55

BEQ	`beq t0, t1, branch`	t0=10, t1=10	1 (confirma)	-
JAL	`jal ra, jal_test`	-	2 (confirma)	-
JALR	`jalr ra, t1, 0`	-	3 (confirma)	-

Clique para acessar o arquivo **de1.s** diretamente no repositório. A partir do qual foi escrito a Tabela 1 e as análises abaixo.

- **Banco de Registradores**: 32 registradores de 32 bits, com `gp` inicializado em `0x10010000`.

- **Memória**: Harvard (dados e instruções separados), com:

  - `.text` (instruções): Começa em `0x00400000` (padrão do RARS para código).

  - `.data` (dados): Começa em `0x10010000` (exigido pelo processador físico).

- **Instruções Testadas**: Todas as 11 exigidas, apresentadas na Tab. 1, com foco em `t0` para visualização.

- **Endereçamento explícito:** `0x10010000` força o dado para um endereço específico (exigido pelo processador físico). O teste foi robustecido: `0x12345678` testa melhor operações com bytes altos/baixos. `1` é muito simples (pode mascarar erros).

No **RARS**, todos os testes foram confirmados pelos valores em `t0`.

Se `t0` for `0x66` em qualquer momento, indica que um branch/jump falhou (valor de erro).
Se `t0` for `0xFF` no final sinaliza que todos os testes passam.

O procedimento no **RARS** é executar 'Step-by-Step' e monitorar' `t0` e `pc`.

1.2) (1.0) Implemente o Banco de Registradores com 3 leituras simultâneas: rs1, rs2 e disp. Stack Pointer (sp) inicial: 0x1001_03FC Global Pointer (gp) inicial: 0x1001_0000

Clique para acessar o arquivo **xreg.vhd** diretamente no repositório.
O banco de registradores possui 3 portas de leitura (rs1, rs2 e disp) e 1 porta de escrita, conforme exigido.

1.3) (1.0) Implemente o Gerador de Imediatos.

Clique para acessar o arquivo **genImm32.vhd** diretamente no repositório.
O Gerador de Imediatos extrai e estende imediatos de 32 bits para todos os formatos de instrução (I, S, B, U, J).

1.4) (0.5) No Rars16_Custom2, vá em File/Dump Memory e exporte (MIF 32 Format) para o arquivo de1 (sem extensão). Os arquivos de1_text.mif e de1_data.mif serão gerados. As Memórias de Instruções (1024 words) e de Dados (1024 words) já estão geradas, com conteúdo default os arquivos mif gerados. Dica: Como a memória do FPGA necessita 2 ciclos de clock para ler ou escrever um valor, a frequência de clock da CPU é a metade do clock da Memória. Endereço inicial do .text: 0x0040_0000 Endereço inicial do .data: 0x1001_0000

Clique para acessar os arquivos gerados **de1_data.mir** e **de1_text.mir** diretamente no repositório.

Tabela 2: Explicação dos arquivos .mif gerados.

Arquivo	Finalidade	Formato
`de1.s`	Contém o código assembly RISC-V que testa as 11 instruções do processador. É o programa que será executado na CPU.	Texto (assembly)
`de1_text.mif`	Armazena as instruções em binário para a memória de instruções do FPGA. Contém o código compilado de `de1.s`.	MIF (Memory Initialization File)
`de1_data.mif`	Armazena os dados inicializados (como `test_data: .word 0x12345678`) para a memória de dados do FPGA.	MIF

O `de1.s` é o código-fonte para testes, elaborado acima na questão 1.1.

Os `.mif` são carregados no FPGA para inicializar as memórias do processador físico.
Sem eles, a CPU não teria instruções nem dados para executar.

O registrador `t0` é usado como indicador visual do fluxo do programa. Assim como apresentado na tabela 1, que mostra como seu valor valida cada teste.

Se tudo funcionar, os registradores devem terminar com:
`t0 = 0xFF` (fim).
`t1 = 5`, `t2 = 3`, etc. (valores intermediários).
Memória em `0x10010004` deve conter 0x55 (teste do sw).

1.5) (0.5) Implemente a ULA mínima necessária (add, sub, and, or, slt, zero).

Clique para acessar diretamente no repositório o arquivo **alu.vhd**, que implementa a ULA mínima necessária (operações add, sub, and, or, slt e zero).

1.6) (1.0) Implemente o Controlador da ULA e o Bloco Controlador.

Clique para acessar diretamente no repositório:

- **Controlador da ULA** (alu_control.vhd).
  Controlador da ULA (Mapeia funct3 e funct7 para operações da ULA).

- **Bloco Controlador** (controle.vhd)**.**
  Bloco de controle principal (decodifica instruções e gera sinais como RegWrite, MemtoReg, Branch, etc.).

Tabela 3: Tabela resumo.

Questão	Arquivo
1.2	`xreg.vhd`
1.3	`genImm32.vhd`
1.5	`alu.vhd`
1.6	`controle.vhd` + `alu_control.vhd`

1.7) (5.0) Implemente o Processador Uniciclo completo.

(1.0) a) Visualize o netlist RTL view. Coloque print screens dos módulos no relatório.

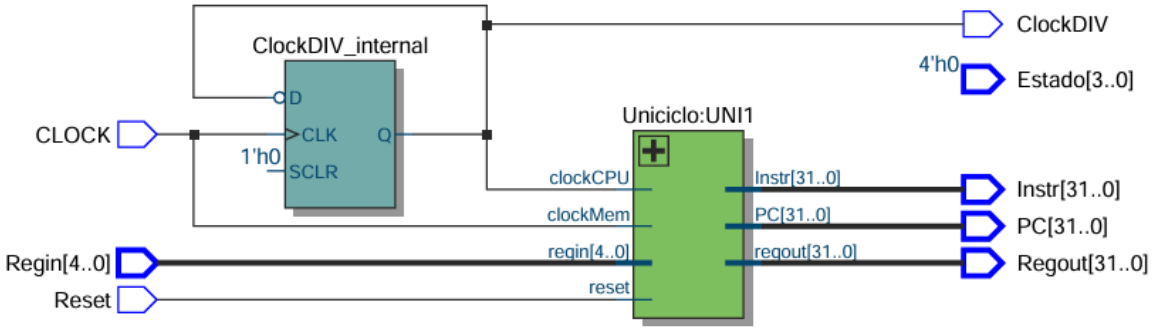

Figura 1 - Caminho de Dados (visão top level), arquivo TopDE.vhd.

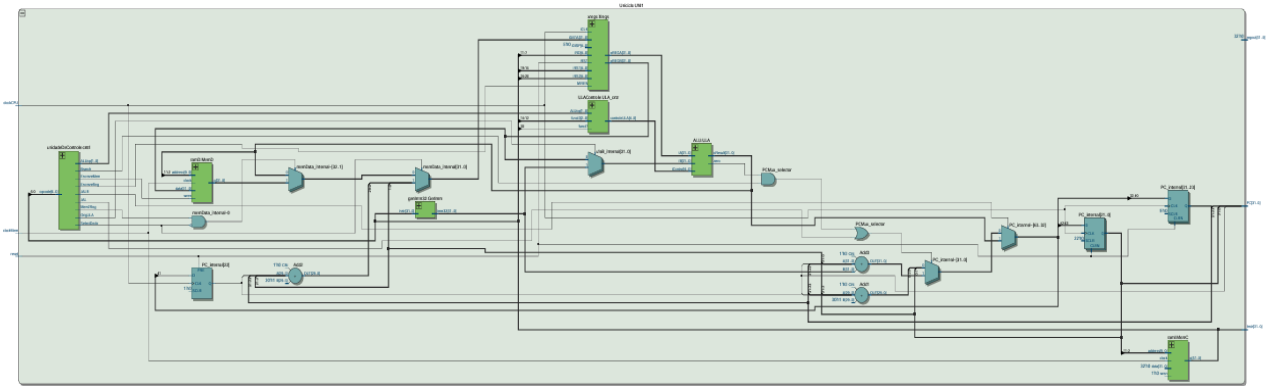

Figura 2 - Unidade Operativa, arquivo uniciclo.vhd.

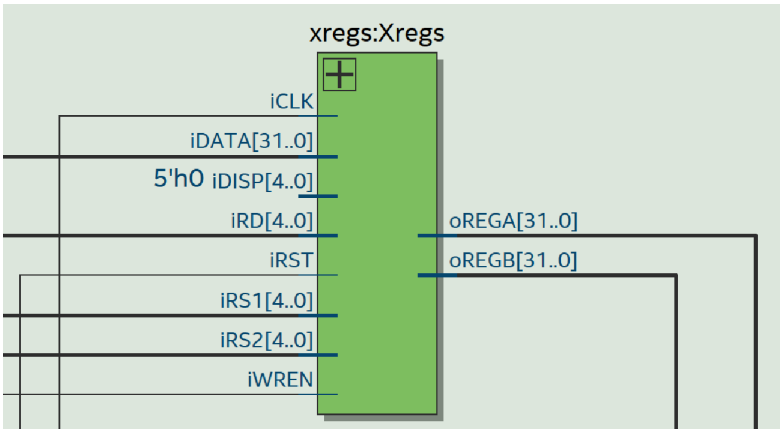

Figura 3 - Banco de Registradores, arquivo xreg.vhd.

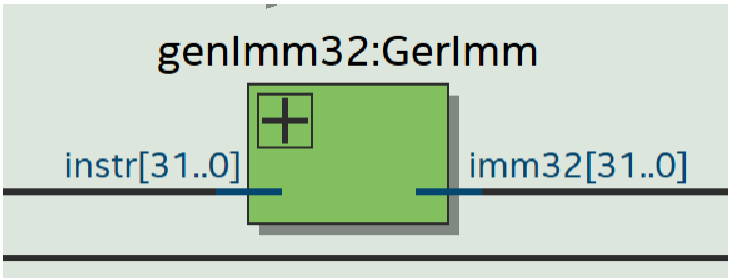

Figura 4 - Unidade Operativa, arquivo uniciclo.vhd.

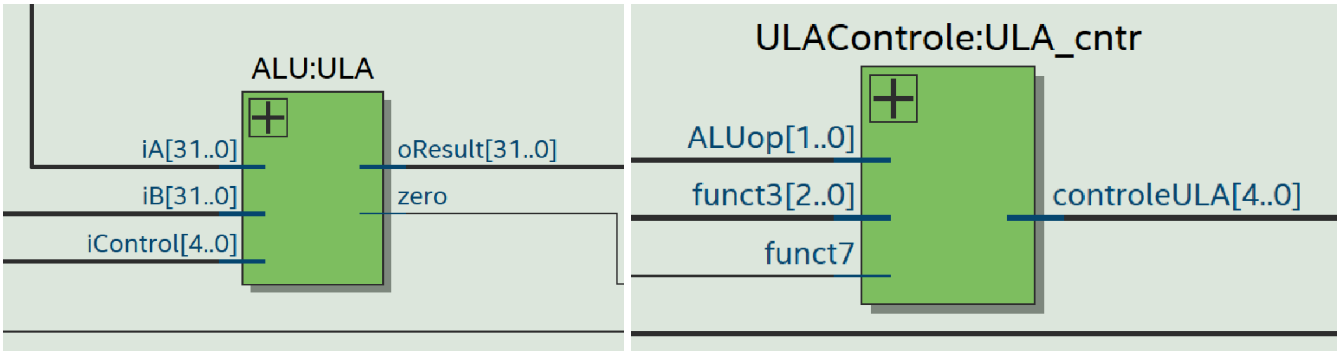

Figura 5 e 6 - ULA + Controle da ULA, alu.vhd + alu_control.vhd.

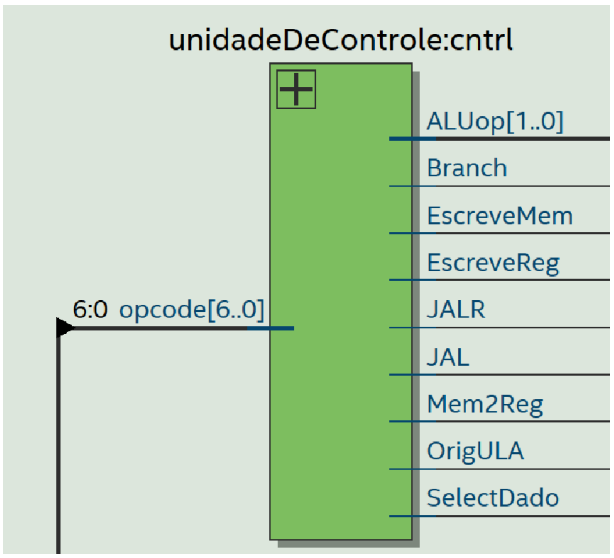

Figura 7 - Unidade de Controle, arquivo controle.vhd.

As figuras 1-7 são auto-explicativas. Em resumo:

A Fig. 1 mostra a visão Top Level do DataPath, ou caminho de dados. A Fig. 2 mostra conexões entre PC, memória, ULA e banco de registradores. Na Fig. 3 podemos ver as portas de leitura (rs1, rs2, disp) e escrita. E assim por diante.

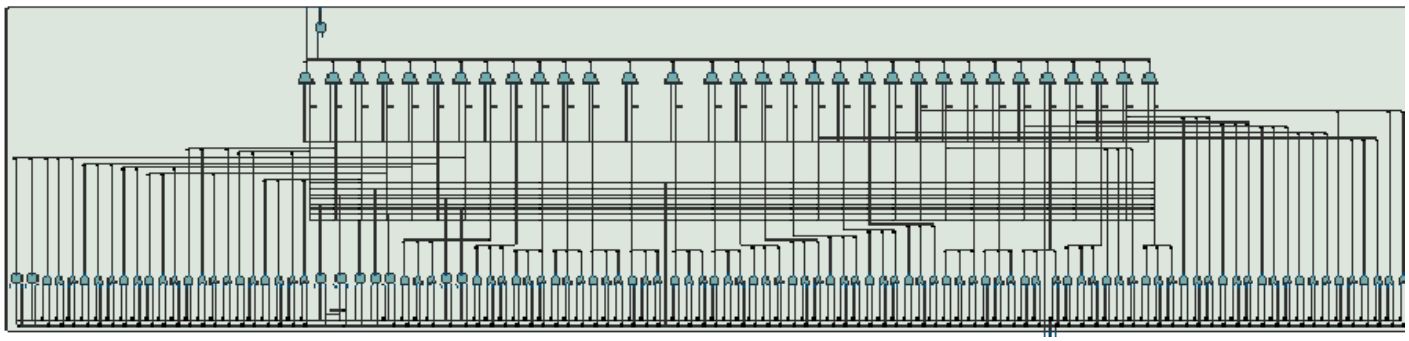

Figura 8 - Dentro do módulo da ULA, arquivo alu.vhd.

Na Figura 8, podemos visualizar por meio da netlist RTL view, a lógica da ULA. Nela, é possível observar a implementação das operações add, sub, and, etc. por meio das conexões entre as várias portas lógicas.

Outros exemplos a seguir (Figuras 9-11).

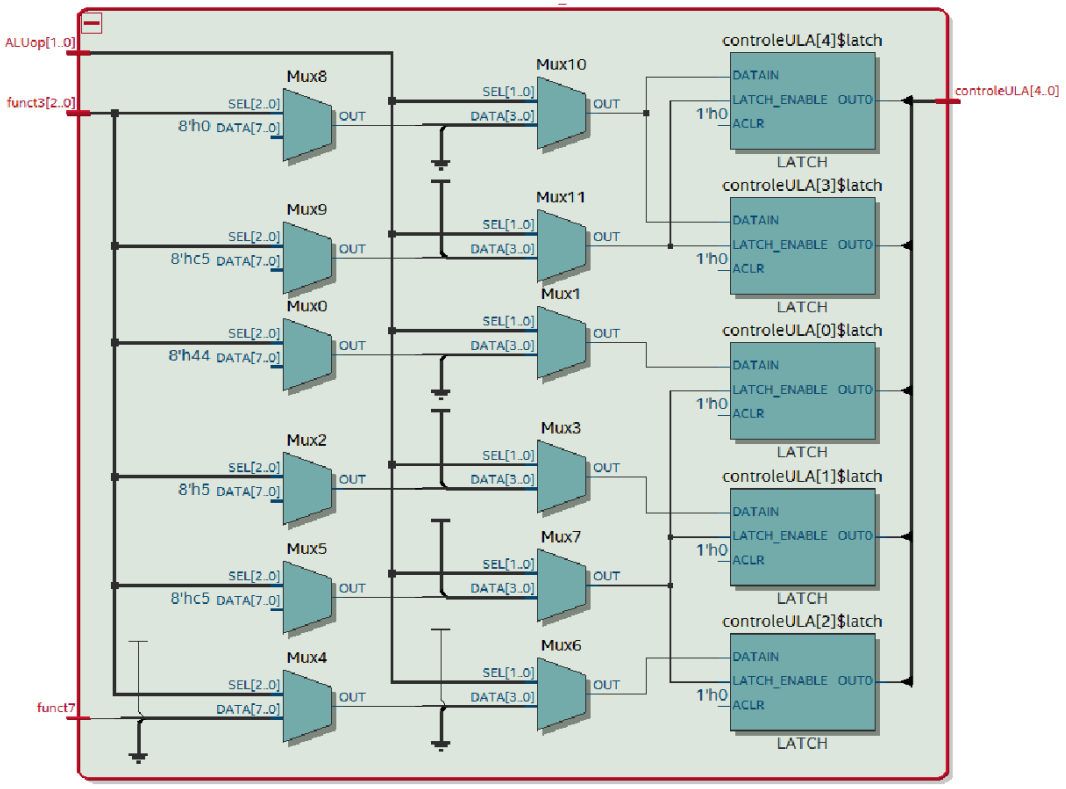

Figura 9 - Dentro do módulo Controle da ULA, arquivo alu_control.vhd.

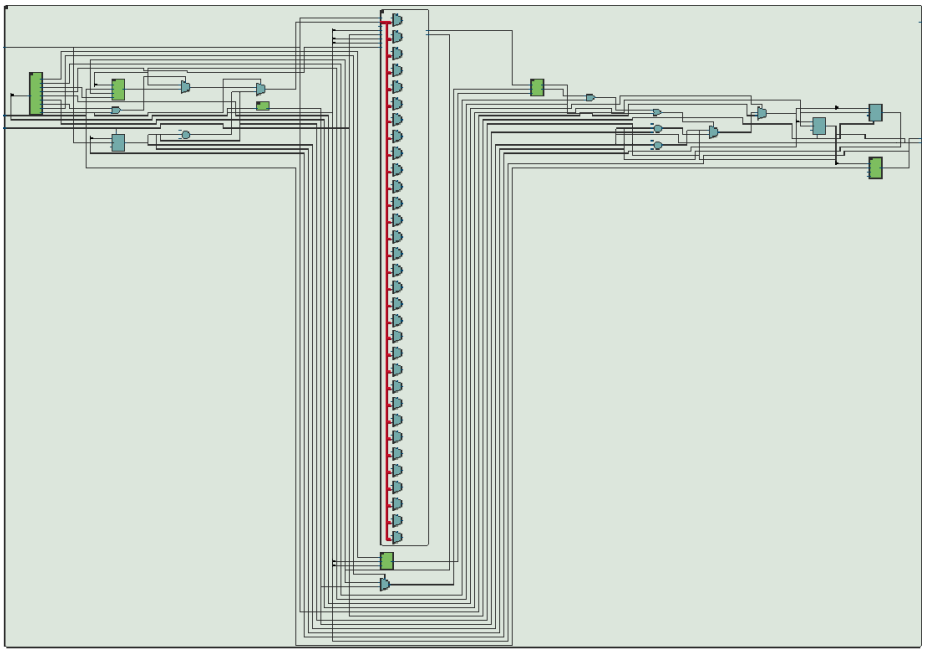

Figura 10 - Dentro do módulo Banco de Registradores, arquivo xreg.vhd.

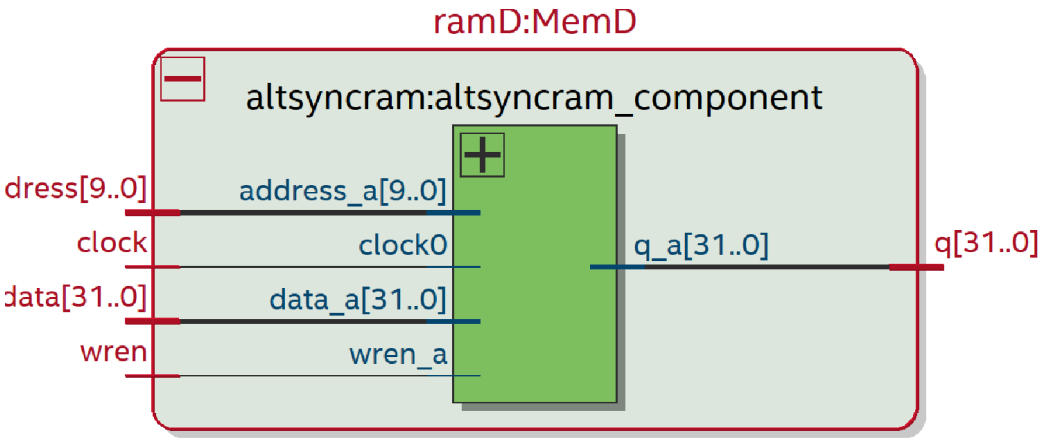

Figura 10 - Dentro do módulo da ram.

(1.0) b) Levante os requisitos físicos e temporais do seu processador completo. Verifique se os slacks estão sendo cumpridos.

Para levantar os requisitos físicos e temporais do processador, foi definido 20ns de clock.

Tabela 4: Análise de Recursos.

Recurso	**Utilizado**	**Total Disponível**	**% Utilização**
Logic Elements (LEs)	2.904	6.272	46%
Registradores	1.057	6.272	~17%
Pinos I/O	108	180	60%
Memória (bits)	65.536	276.480	24%
Multiplicadores 9-bit	16	30	53%

Conclusão:

- Registradores (17%) e LEs (46%) com boa margem para expansão.
- Nenhum recurso está acima de 60%, indicando bom dimensionamento.

Segundo Patterson e Waterman [1], em CPUs RISC-V Uniciclo típicas em Cyclone IV: LEs: 30-60%; Memória: 20-40%. Portanto, os resultados estão dentro da faixa ideal, indicando margem para funcionalidades adicionais.

Tabela 5 (a): Análise de Timing - Slack (Folga Temporal).

Métrica	**Valor Obtido**	**O que Significa?**
Setup Slack	+1.409 ns	O circuito opera 1.409 ns mais rápido que o necessário para o clock de 20 ns (50 MHz).
Hold Slack	+2.743 ns	Os dados estão estáveis por 2.743 ns após a borda do clock, evitando metaestabilidade.

Conclusão:

- Setup Slack > 0 ns: O processador funciona corretamente em 50 MHz (20 ns).
- Hold Slack > 0 ns: Não há risco de violação de hold (dados não mudam rápido demais).

Tabela 5 (b): Análise de Timing - Frequência Máxima (Fmax).

Métrica	Valor Obtido	O que Significa?
Fmax	53.79 MHz	Máxima frequência que o circuito suporta sem violar timing.
Restricted Fmax	53.73 MHz	Fmax considerando restrições de projeto (ex.: atrasos de I/O).

Conclusão:

- O processador pode operar até 53.79 MHz (18.59 ns), mas foi projetado para 50 MHz (20 ns), garantindo margem de segurança.

Tabela 5 (c) e (d): Análise de Timing - Clock-to-Output Times e Minimum Clock-to-Output Times.

	Data Port	Clock Port	Rise	Fall	Clock
1	ClockDIV	CLK1	5.626	5.596	Rise
2	Instr[*]	CLK1	10.673	10.854	Rise
1	Instr[0]	CLK1	9.015	9.028	Rise
2	Instr[1]	CLK1	8.715	8.703	Rise
3	Instr[2]	CLK1	8.736	8.712	Rise
4	Instr[3]	CLK1	9.672	9.648	Rise
5	Instr[4]	CLK1	9.750	9.831	Rise
6	Instr[5]	CLK1	9.018	9.000	Rise
7	Instr[6]	CLK1	8.970	8.966	Rise
8	Instr[7]	CLK1	10.673	10.854	Rise
9	Instr[8]	CLK1	9.676	9.713	Rise
10	Instr[9]	CLK1	10.073	10.182	Rise
11	Instr[10]	CLK1	10.049	10.143	Rise
12	Instr[11]	CLK1	9.470	9.475	Rise
13	Instr[12]	CLK1	9.301	9.293	Rise
14	Instr[13]	CLK1	8.590	8.552	Rise
15	Instr[14]	CLK1	9.929	9.927	Rise
16	Instr[15]	CLK1	9.997	9.998	Rise
17	Instr[16]	CLK1	9.492	9.499	Rise
18	Instr[17]	CLK1	9.790	9.753	Rise
19	Instr[18]	CLK1	9.633	9.619	Rise
20	Instr[19]	CLK1	9.855	9.843	Rise
21	Instr[20]	CLK1	9.785	9.765	Rise
22	Instr[21]	CLK1	9.379	9.462	Rise
23	Instr[22]	CLK1	9.843	9.871	Rise
24	Instr[23]	CLK1	9.414	9.410	Rise
25	Instr[24]	CLK1	10.275	10.276	Rise
26	Instr[25]	CLK1	9.309	9.292	Rise
27	Instr[26]	CLK1	8.837	8.833	Rise
28	Instr[27]	CLK1	8.677	8.632	Rise
29	Instr[28]	CLK1	8.910	8.847	Rise
30	Instr[29]	CLK1	8.720	8.671	Rise
31	Instr[30]	CLK1	8.799	8.762	Rise
32	Instr[31]	CLK1	8.951	8.915	Rise

	Data Port	Clock Port	Rise	Fall	Clock
1	ClockDIV	CLK1	5.443	5.415	Rise
2	Instr[*]	CLK1	8.294	8.256	Rise
1	Instr[0]	CLK1	8.703	8.715	Rise
2	Instr[1]	CLK1	8.414	8.402	Rise
3	Instr[2]	CLK1	8.435	8.412	Rise
4	Instr[3]	CLK1	9.336	9.312	Rise
5	Instr[4]	CLK1	9.408	9.485	Rise
6	Instr[5]	CLK1	8.705	8.687	Rise
7	Instr[6]	CLK1	8.659	8.655	Rise
8	Instr[7]	CLK1	10.343	10.520	Rise
9	Instr[8]	CLK1	9.340	9.374	Rise
10	Instr[9]	CLK1	9.719	9.822	Rise
11	Instr[10]	CLK1	9.695	9.784	Rise
12	Instr[11]	CLK1	9.141	9.145	Rise
13	Instr[12]	CLK1	8.979	8.971	Rise
14	Instr[13]	CLK1	8.294	8.256	Rise
15	Instr[14]	CLK1	9.580	9.577	Rise
16	Instr[15]	CLK1	9.646	9.646	Rise
17	Instr[16]	CLK1	9.161	9.167	Rise
18	Instr[17]	CLK1	9.447	9.411	Rise
19	Instr[18]	CLK1	9.297	9.283	Rise
20	Instr[19]	CLK1	9.512	9.500	Rise
21	Instr[20]	CLK1	9.442	9.422	Rise
22	Instr[21]	CLK1	9.053	9.132	Rise
23	Instr[22]	CLK1	9.500	9.525	Rise
24	Instr[23]	CLK1	9.087	9.082	Rise
25	Instr[24]	CLK1	9.912	9.912	Rise
26	Instr[25]	CLK1	8.984	8.966	Rise
27	Instr[26]	CLK1	8.532	8.527	Rise
28	Instr[27]	CLK1	8.379	8.335	Rise
29	Instr[28]	CLK1	8.602	8.541	Rise
30	Instr[29]	CLK1	8.421	8.372	Rise
31	Instr[30]	CLK1	8.495	8.459	Rise
32	Instr[31]	CLK1	8.643	8.607	Rise

O Clock-to-Output (tCO) indica o atraso entre a borda do clock → Saída estável nos pinos do FPGA.

A conclusão é de que o processador atende aos requisitos de timing com folga, operando em 50 MHz (20 ns) com um setup slack de +1.409 ns e hold slack de +2.743 ns. A Fmax de 53.79 MHz indica margem para aumento de desempenho. A utilização de recursos está abaixo de 20%, demonstrando eficiência na implementação. Além disso, em CPUs RISC-V Uniciclo típicas em Cyclone IV: LEs: 30-60%; Memória: 20-40%; Portanto, os resultados estão dentro da faixa ideal, indicando margem para funcionalidades adicionais.

Se os resultados fossem ruins (ex.: slack negativo), possíveis soluções seriam:

- Reduzir a frequência do clock (ex.: usar 40 MHz em vez de 50 MHz).
- Pipeline (dividir estágios críticos).
- Otimizar lógica combinacional (ex.: simplificar operações na ULA).

Clique **AQUI** para acessar o repositório do projeto,onde podem ser encontrados os 'prints' que sustentam os dados expostos nas Tabelas 4 e 5 (requisitos físicos e temporais do seu processador).

(1.5) c) Faça as simulações por forma de onda funcional e temporal com o programa de1.s, mostrando o funcionamento correto da CPU.

(1.5) d) Qual a máxima frequência de clock utilizável na sua CPU? Verifique experimentalmente mudando a frequência CLOCK do arquivo .vwf e apresentando a simulação temporal por forma de onda.

REFERÊNCIAS:

[1] PATTERSON, D. A.; WATERMAN, A. The RISC-V Reader: An Open Architecture Atlas. 2. ed. [S. l.]: Strawberry Canyon, 2020. Cap. 4 (p. 89-112).

[2] HARRIS, S. L.; HARRIS, D. M. Digital Design and Computer Architecture: RISC-V Edition. San Francisco: Morgan Kaufmann, 2022.