



Laboratório 4

- CPU RISC-V MULTICICLO –

Objetivos:

- Treinar o aluno com a linguagem de descrição de hardware Verilog;
- Familiarizar o aluno com a plataforma de desenvolvimento FPGA DE1SoC da Intel e o software QUARTUS Prime;
- Desenvolver a capacidade de análise e síntese de sistemas digitais usando uma Linguagem de Descrição de Hardware;
- Apresentar ao aluno a implementação de uma CPU RISC-V Multiciclo;

PARTE A: Apresentação do processador MIPS Multiciclo

- 1) (0.0) Abra o projeto do processador MIPS-v1.1:
 - 1.1) Defina o uso do processador UNICICLO, compile o projeto. Compare a execução do programa testeWaveform2.s no Mars e a simulação por forma de onda através do arquivo Waveform2.vwf. Manualmente, encontre a máxima frequência de clock no qual o programa ainda é corretamente executado e o tempo de execução.
 - 1.2) Defina o uso do processador MULTICICLO, compile o projeto. Compare a execução do programa testeWaveform2.s no Mars e a simulação por forma de onda através do arquivo Waveform2.vwf. Manualmente, encontre a máxima frequência de clock no qual o programa ainda é corretamente executado e o tempo de execução.
 - 1.3) Compare os requerimentos físicos i) Número de Elementos Lógicos (ALMs), ii) Número de Registradores e iii) Quantidade de bits de memória e iv) Número de blocos DSP; e temporais : i) caminho de maior atraso tpd, ii) caminhos com piores tempos th, tco, tsu e slacks, iii) máxima frequência de clock dos processadores MIPS UNICICLO e MULTICICLO.

PARTE B: Processador RISC-V Multiciclo

2. (8.0) Dado o ambiente de desenvolvimento fornecido no arquivo RISCV-v1.0.zip, implemente e teste o seu processador RISC-V Multiciclo.
 - 2.1) (4.0) Mantendo a interface com os barramentos (de dados e de instruções), MMIOs e sinais de clock, implemente um processador Multiciclo compatível com a ISA RV32I com as instruções listadas abaixo. Descreva o projeto realizado. Desenhe o Caminho de Dados (conforme diagrama visto em aula) e a máquina de estados do Bloco de Controle. Analise e comente as dificuldades técnicas enfrentadas e as soluções propostas.
- 1) add t1,t2,t3 Addition: set t1 to (t2 plus t3)
 - 2) sub t1,t2,t3 Subtraction: set t1 to (t2 minus t3)
 - 3) and t1,t2,t3 Bitwise AND : Set t1 to bitwise AND of t2 and t3
 - 4) or t1,t2,t3 Bitwise OR : Set t1 to bitwise OR of t2 and t3
 - 5) xor t1,t2,t3 Bitwise XOR : Set t1 to bitwise XOR of t2 and t3
 - 6) slt t1,t2,t3 Set less than : If t2 is less than t3, then set t1 to 1 else set t1 to 0
 - 7) sltu t1,t2,t3 Set less than : If t2 is less than t3 using unsigned comparison, then set t1 to 1 else set t1 to 0
 - 8) sll t1,t2,t3 Shift left logical: Set t1 to result of shifting t2 left by number of bits specified by low-order 5 bits of t3
 - 9) srl t1,t2,t3 Shift right logical: Set t1 to result of shifting t2 right by number of bits specified in low-order 5 bits of t3
 - 10) sra t1,t2,t3 Shift right arithmetic: Set t1 to result of sign-extended shifting t2 right by number of bits specified by value in low-order 5 bits of t3
 - 11) addi t1,t2,-100 Addition immediate: set t1 to (t2 plus signed 12-bit immediate)
 - 12) andi t1,t2,-100 Bitwise AND immediate : Set t1 to bitwise AND of t2 and sign-extended 12-bit immediate
 - 13) ori t1,t2,-100 Bitwise OR immediate : Set t1 to bitwise OR of t2 and sign-extended 12-bit immediate
 - 14) xori t1,t2,-100 Bitwise XOR immediate : Set t1 to bitwise XOR of t2 and sign-extended 12-bit immediate
 - 15) slti t1,t2,-100 Set less than imm. : If t2 is less than sign-extended 12-bit immediate, then set t1 to 1 else set t1 to 0
 - 16) sltiu t1,t2,-100 Set less than immediate unsigned : If t2 is less than sign-extended 16-bit immediate using unsigned comparison, then set t1 to 1 else set t1 to 0
 - 17) slli t1,t2,10 Shift left logical : Set t1 to result of shifting t2 left by number of bits specified by immediate
 - 18) srli t1,t2,10 Shift right logical : Set t1 to result of shifting t2 right by number of bits specified by immediate

- 19) srai t1,t2,10 Shift right arithmetic : Set t1 to result of sign-extended shifting t2 right by number of bits specified by immediate
- 20) auipc t1,-100000 Add upper immediate to pc: set t1 to (pc plus an upper signed 20-bit immediate)
- 21) lui t1,-100000 Load upper immediate: set t1 to 20-bit followed by 12 0s
- 22) beq t1,t2,label Branch if equal : Branch to statement at label's address if t1 and t2 are equal
- 23) bne t1,t2,label Branch if not equal : Branch to statement at label's address if t1 and t2 are not equal
- 24) bge t1,t2,label Branch if greater than or equal: Branch to label's address if t1 is greater than or equal to t2
- 25) bgeu t1,t2,label Branch if greater than or equal to (unsigned): Branch to statement at label's address if t1 is greater than or equal to t2 (with an unsigned interpretation)
- 26) blt t1,t2,label Branch if less than: Branch to statement at label's address if t1 is less than t2
- 27) bltu t1,t2,label Branch if less than (unsigned): Branch to statement at label's address if t1 is less than t2 (with an unsigned interpretation)
- 28) jal t1, target Jump and link : Set t1 to Program Counter (return address) then jump to statement at target address
- 29) jalr t1, t2, -100 Jump and link register: Set t1 to Program Counter (return address) then jump to statement at t2 + immediate
- 30) lb t1, -100(t2) Set t1 to sign-extended 8-bit value from effective memory byte address
- 31) lbu t1, -100(t2) Set t1 to zero-extended 8-bit value from effective memory byte address
- 32) lh t1, -100(t2) Set t1 to sign-extended 16-bit value from effective memory halfword address
- 33) lhu t1, -100(t2) Set t1 to zero-extended 16-bit value from effective memory halfword address
- 34) lw t1, -100(t2) Set t1 to contents of effective memory word address
- 35) sb t1, -100(t2) Store byte : Store the low-order 8 bits of t1 into the effective memory byte address
- 36) sh t1, -100(t2) Store halfword : Store the low-order 16 bits of t1 into the effective memory halfword address
- 37) sw t1, -100(t2) Store word : Store contents of t1 into effective memory word address

2.2) (0.5) Indique os requisitos físicos da implementação do seu processador Multiciclo RV32I: i) Número de Elementos Lógicos (ALMs), ii) Número de Registradores, iii) Quantidade de bits de memória e iv) Número de blocos DSP usados;

2.3) (0.5) (0.5) Usando o TimeQuest, defina um clock básico de 50MHz. Apresente os requerimentos temporais do seu processador Multiciclo RV32I: i) caminho de maior atraso tpd, ii) caminhos com piores tempos th, tco, tsu e slacks, iii) máxima frequência de clock utilizável, e iv) se há algum requerimento não atendido.

2.4) (2.0) Implemente um processador Multiciclo compatível com a ISA RV32IM acrescentando as instruções listadas abaixo. Descreva o projeto realizado e indique as modificações necessárias no caminho de dados e no bloco de controle. Comente as dificuldades técnicas enfrentadas e as soluções propostas.

- 38) mul t1,t2,t3 Multiplication: set t1 to the lower 32 bits of t2*t3
- 39) mulh t1,t2,t3 Multiplication: set t1 to the upper 32 bits of t2*t3 using signed multiplication
- 40) mulhu t1,t2,t3 Multiplication: set t1 to the upper 32 bits of t2*t3 using unsigned multiplication
- 41) mulhsu t1,t2,t3 Multiplication: set t1 to the upper 32 bits of t2*t3 using signed and unsigned multiplication
- 42) div t1,t2,t3 Division: set t1 to the result of t2/t3
- 43) divu t1,t2,t3 Division: set t1 to the result of t2/t3 using unsigned division
- 44) rem t1,t2,t3 Remainder: set t1 to the remainder of t2/t3
- 45) remu t1,t2,t3 Remainder: set t1 to the remainder of t2/t3 using unsigned division

2.5) (0.5) Indique os requisitos físicos da implementação do seu processador Multiciclo RV32IM: i) Número de Elementos Lógicos (ALMs), ii) Número de Registradores, iii) Quantidade de bits de memória e iv) Número de blocos DSP usados;

2.6) (0.5) Usando o TimeQuest, defina um clock básico de 50MHz. Apresente os requerimentos temporais do seu processador Multiciclo RV32IM: i) caminho de maior atraso tpd, ii) caminhos com piores tempos th, tco, tsu e slacks, iii) máxima frequência de clock utilizável, e iv) se há algum requerimento não atendido.

3. (1.0) Com o programa testbech.s criado no Laboratório 3, verifique a corretude de cada instrução implementada, indicando na tela (em hexadecimal), caso ocorra algum erro, qual foi o endereço e a instrução executada erroneamente. Filme a execução deste programa no Rars e na DE1SoC. Analise os resultados obtidos e comente as dificuldades enfrentadas.
4. (1.0) Usando o programa testeECALLv1.s (substituindo as instruções não implementadas ecall e uret por jal e jr) e o seu exception handler SYSTEMv1.s desenvolvido no Laboratório 1, filme o funcionamento do seu processador RISC-V Multiciclo RV32IM na DE1-SoC comprovando seu correto funcionamento. Analise os resultados obtidos e comente as dificuldades enfrentadas.