**UNIVERSIDADE DE SÃO PAULO**

ESCOLA DE ENGENHARIA DE SÃO CARLOS

**SEL0384 - Laboratório de Sistemas Digitais 1**

Prof. Maximiliam Luppe

Bárbara Fernandes Madera - nº: 11915032

Johnny Caselato Guimarães - nº: 11915481

**Prática 5 - Relatório de Prática de Laboratório: Somador Completo**

**SÃO CARLOS**

**2023**

**1. Objetivos**

O propósito deste projeto é permitir que os participantes se familiarizarem com a aplicação da ferramenta Quartus Lite da Intel e adquiram competência na realização de projetos e síntese de circuitos combinacionais em dispositivos reconfiguráveis (FPGA), empregando a linguagem de descrição de hardware VHDL e seguindo uma abordagem de projeto hierárquico baseada na metodologia Top-down no desenvolvimento de um somador de 2 entradas de 4 bits. Além disso, o circuito proposto também será implementado utilizando circuitos da família 74XX, disponíveis no software de simulação SimulIDE.

**2. Equipamentos Necessários para Prática:**

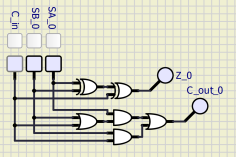
● Kit Mercurio® IV

● Software Quartus II Web Edition

**3. Procedimento Experimental**

No procedimento experimental, será demonstrada a construção de um somador de 4 bits utilizando somadores completos na configuração de RPC (Ripple-Carry). A implementação feita em código no software do Quartus, poderá também ser realizada com componentes TTL da família 74LSxx. Essa configuração do circuito somador é muito comum e fundamental para a soma de bits de múltiplas entradas em paralelo, onde cada bit é somado considerando a “sobra” (*carry*) do resultado anterior. Na figura 1 vemos um exemplo de implementação básica utilizando portas lógicas genéricas para a construção da lógica de soma de dois bits.

Figura 1 - Somador completo básico em portas lógicas genéricas.



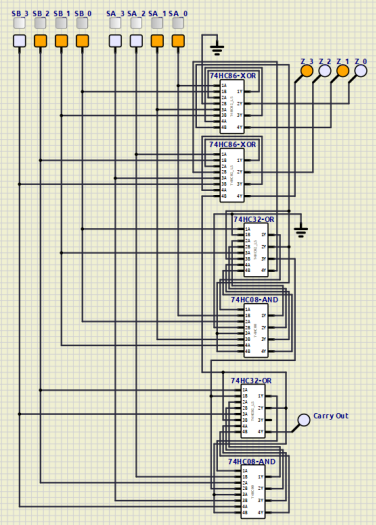
A partir desta unidade única, podemos combinar múltiplos circuitos da família 74XX para a fazer a soma de 2 entradas de 4 bits. Desta forma, o circuito completo presente na figura 2 é composto por:

● 2 circuitos 7486 (4 XOR - 2 Entradas);

● 2 circuitos 7432 (4 OR - 2 Entradas);

● 2 circuitos 7408 (4 AND - 2 Entradas);

Figura 2 - Circuito somador implementado com circuitos da família 74XX.



Para organizar e estruturar o projeto “MercurioIV\_adder” de forma adequada, como realizado nas práticas anteriores, é criada uma pasta principal de mesmo nome, que contém subpastas específicas, como "docs", "modelsim", "quartus" e "src". Dentro deste projeto, é implementada a entidade de projeto "MercurioIV\_adder.vhd", definida como a "Entidade de Nível Superior" (Top-Level Entity). Neste arquivo, são estabelecidas conexões entre as entradas A e B e os interruptores SA e SB, respectivamente. A saída RESULT é conectada aos LEDs LEDM\_R(3 downto 0). Vale ressaltar que, no contexto deste projeto, os LEDs se acenderão quando receberem um sinal de nível lógico '0', enquanto as colunas LEDM\_C estarão em nível lógico '1':

**--Projeto Somador top leve: MercurioIV\_adder.vhd**

**--SEL0384 - Atividade 5**

**--Autores:**

**--Johnny Caselato Guimaraes - N: 11915481**

**--Barbara Fernandes Madera - N: 11915032**

**--Prof.: Maxmillian Lupe**

**--data: 27/09/23**

**--Entidade e arquitetura q/ modula somador(adder) do kit MercurioIV**

**ENTITY MercurioIV\_adder is**

**PORT(**

**SA, SB : in bit\_vector(7 downto 0);--portas de entrada de 8 bits**

**LEDM\_R : out bit\_vector(7 downto 0);--porta de saída resultado do**

**--somador**

**LEDM\_C : out bit\_vector(4 downto 0) --porta saída,indica o tipo de carry );**

**END MercurioIV\_adder;**

**ARCHITECTURE hierarquia of MercurioIV\_adder is**

**BEGIN**

**adder\_0 : work.adder port map(**

**A => SA(3 downto 0),**

**B => SB(3 downto 0),**

**RESULT => LEDM\_R(3 downto 0)**

**);**

**LEDM\_C <= "00000";**

**LEDM\_R(7 downto 4) <= "0000";**

**END;**

**--A arquitetura hierarquia instancia um componente de somador p/ somar 4 bits menos --significativos das 2 entradas e define as saídas LEDM\_C e LEDM\_R como zero**

A implementação do componente somador de 4 bits é realizada com a utilização do componente "adder.vhd". Esse somador processa entradas A e B e gera uma saída denominada RESULT, que representa a soma dos bits de entrada:

**--Projeto Somador 4 bits: adder.vhd**

**--SEL0384 - Atividade 5**

**--Autores:**

**--Johnny Caselato Guimaraes - N: 11915481**

**--Barbara Fernandes Madera - N: 11915032**

**--Prof.: Maxmillian Lupe**

**--data: 27/09/23**

**--Entidade e arquitetura para um somador(adder) q/ opera em 4 bits**

**ENTITY adder is**

**PORT(**

**A, B : in bit\_vector(3 downto 0); --2 portas de entrada**

**RESULT : out bit\_vector(3 downto 0) --resultado da soma A + B**

**);**

**END adder;**

**ARCHITECTURE hierarquia of adder is --arquitetura realiza adição de 4 bits de A e B**

**signal C : bit\_vector(3 downto 1); --transporta o carry-out das adições --individuais de cada bit**

**BEGIN --cada fadd executa a adição de 1 bit considerando tanto os componentes --A e B e os carry-in e carry-out anteriores**

**fadd\_0 : work.fadd port map(**

**a => A(0),**

**b => B(0),**

**ci => '0',**

**s => RESULT(0),**

**co => C(1)**

**);**

**fadd\_1 : work.fadd port map(**

**a => A(1),**

**b => B(1),**

**ci => C(1),**

**s => RESULT(1),**

**co => C(2)**

**);**

**fadd\_2 : work.fadd port map(**

**a => A(2),**

**b => B(2),**

**ci => C(2),**

**s => RESULT(2),**

**co => C(3)**

**);**

**fadd\_3 : work.fadd port map(**

**a => A(3),**

**b => B(3),**

**ci => C(3),**

**s => RESULT(3)**

**);**

**END;**

Além disso, como último elemento na hierarquia, um somador completo adicional, chamado "fadd.vhd", é criado, no qual as lógicas de somatória serão efetivamente implementadas. Este componente possui entradas, ou parâmetros, a, b e ci (carry-in) e saídas s (soma) e co (carry-out). O comando concorrente "WITH-SELECT" é empregado para implementar a lógica da saída s, enquanto a estrutura "WHEN-ELSE" é utilizada para implementar a lógica da saída ‘co’:

**--Projeto Somador completo: fadd.vhd**

**--SEL0384 - Atividade 5**

**--Autores:**

**--Johnny Caselato Guimaraes - N: 11915481**

**--Barbara Fernandes Madera - N: 11915032**

**--Prof.: Maxmillian Lupe**

**--data: 27/09/23**

**--Entidade e Arquitetura p/ somador completo de único bit(fadd)**

**ENTITY fadd is**

**PORT(**

**a, b, ci : in bit; --portas de entrada a,b a serem somadas/ci tipo carry-in s, co : out bit --s eh porta de saida tipo bit/co eh carry-out**

**);**

**END fadd;**

**--Arquitetura faz operações feitas de acordo com a tabela verdade p/ somador completo de --1bit**

**ARCHITECTURE concorrente of fadd is**

**BEGIN**

**WITH a & b & ci SELECT**

**-- Saida s com WITH-SELECT**

**s <= '0' when "000" ,**

**'1' when "001" ,**

**'1' when "010" ,**

**'0' when "011" ,**

**'1' when "100" ,**

**'0' when "101" ,**

**'0' when "110" ,**

**'1' when "111" ;**

**-- Saida co com WHEN-ELSE**

**co <= '1' when a & b & ci = "011" else --a = '1' and b = '1'**

**'1' when a & b & ci = "101" else**

**'1' when a & b & ci = "110" else**

**'1' when a & b & ci = "111" else**

**'0';**

**END;**

**4. Resultados obtidos**

Em seguida encontra-se a visualização do respectivo Register-Transfer Levels (RTL) resultante da compilação do projeto, onde cada estrutura na hierarquia é explicitada nas figuras 3 a 5, partindo, respectivamente, de uma visão mais generalizada até a conexão interna dos blocos lógicos dentro de um somador completo individual.

Figura 3 - Visão geral somador.

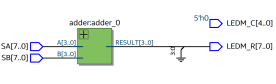


Figura 4 - Visão interna do elemento somador.

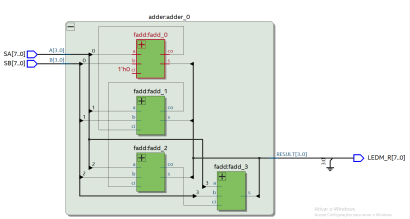
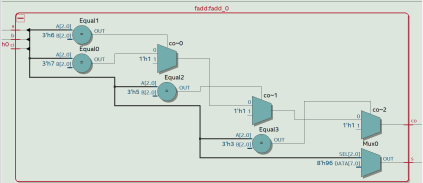


Figura 5 - Visão interna da implementação de uma unidade de somador completo. 

Portanto, conforme as imagens, enquanto no primeiro caso é envolvido a adição de dois números de 4 bits usando registradores temporários e saída armazena em um registrados distinto cujo carry-out é gerado internamente pelo componente do somador, o segundo, por sua vez, cria um somador de 4 bits usando quatro somadores de 1 bit em paralelo, o que resulta valores intermediários mantidos em registros internos e o carry- out é propagando entre os somadores. Por conseguinte, o terceiro caso possui um somador completo de 1 bit, onde as operações são realizadas diretamente em bits individuais. Dessa forma, o resultado da soma é armazenado na saída, e o carry-out é calculado com base em operações lógicas condicionais realizadas por meio da tabela verdade.

**5. Conclusão**

Nesta prática laboratorial, nosso foco foi a implementação de um somador de 4 bits em dispositivos de lógica programável FPGA, utilizando a ferramenta Quartus Lite da Intel e seguindo uma metodologia de projeto hierárquico com a abordagem Top-down. Essa experiência nos permitiu explorar a aplicação prática de mais um circuito combinacional e compreender a flexibilidade dos dispositivos reconfiguráveis através da criação de uma estrutura organizada de pastas e subpastas que foram gerenciadas no projeto, assim como componentes TTL da família 74LSxx e ambiente FPGA do kit Mercurio® IV foram utilizados nesta prática.

Assim, compreendeu-se circuitos combinacionais e manipulação da FPGA, e também demonstrou a importância de uma abordagem sistemática e hierárquica no desenvolvimento de sistemas digitais. Ademais, com a obtenção de resultados práticos, o número de células lógicas utilizadas e a visualização do circuito em funcionamento, contribuíram significativamente para a consolidação do aprendizado prático em Sistemas Digitais.