**UNIVERSIDADE DE SÃO PAULO**
ESCOLA DE ENGENHARIA DE SÃO CARLOS

**SEL0384 – Laboratório de Sistemas Digitais I**

Prof. Dr. Maximiliam Luppe

Bárbara Fernandes Madera - nº: 11915032

Johnny Caselato Guimarães - nº: 11915481

# PRÁTICA Nº9
# Dispositivos de Lógica Programável tipo FPGA
# Contador Binário Síncrono

**SÃO CARLOS**

**2023**

## 1. Objetivo

O propósito deste relatório é apresentar o processo de implementação de um contador binário síncrono com reset assíncrono utilizando a linguagem de descrição de hardware VHDL no kit Mercurio® IV (Cyclone® IV EP4CE30F23). O contador será projetado com um barramento de dados de tamanho parametrizável através de uma estrutura de arquitetura genérica, podendo ser sobrescrita para outras aplicações.

## 2. Introdução

Contadores são componentes fundamentais em aplicações digitais, utilizados para monitorar eventos ou sequências. Eles podem ser classificados como síncronos, que operam com um sinal de clock comum, e assíncronos, que não dependem de um clock compartilhado, permitindo transições independentes entre os flip-flops. Essas duas categorias de contadores desempenham papéis essenciais em circuitos digitais, proporcionando flexibilidade para atender a diferentes aplicações.

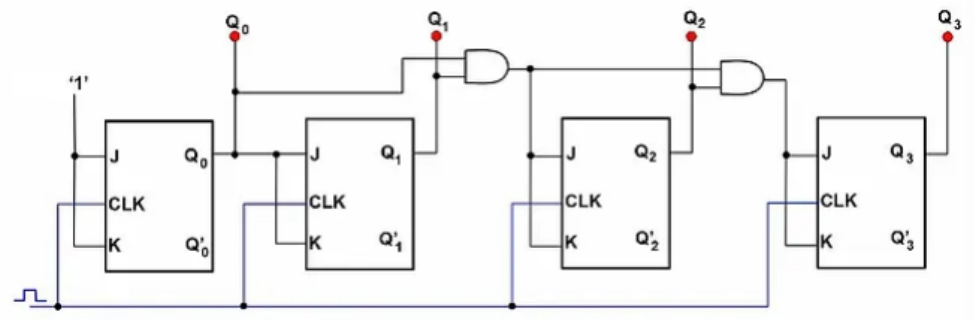


Figura 1 - Exemplo de contador binário síncrono implementado.

Os contadores síncronos garantem transições simultâneas e previsíveis entre flip-flops, nesta atividade, do tipo JK, sendo ideais para aplicações que exigem precisão de tempo. Já os contadores assíncronos, embora mais simples, podem apresentar problemas de temporização devido a variações nos atrasos de propagação.

### 3. Equipamentos Necessários para Prática:

- Kit Mercurio® IV
- Software Quartus II Web Edition

### 4. Implementação e Resultados

- Código VHDL:

Nesta atividade foram utilizados 4 arquivos VHDL: MercurioIV_counter, counter, jkff e MercurioIV_decod. Os códigos correspondentes ao decodificador e ao *flip-flop* JK não serão apresentados aqui pois já haviam sido disponibilizados ou implementados em outras atividades.

- MercurioIV_counter

No código de “MercurioIV_counter” é feita a interface entre os elementos do hardware no kit com os parâmetros e componentes do software para a operação do contador, através dos botões e do decodificador para o display de 7 segmentos.

```
-- Projeto Contador
-- Autores:
-- Bárbara Fernandes Madera   - n°: 11915032
-- Johnny Caselato Guimarães  - n°: 11915481
-- Professor: Maximiliam Luppe

entity MercurioIV_counter is
	port(
		KEY		: in bit_vector(11 downto 0);
		DISP0_D		: out bit_vector(7 downto 0)
	);
end MercurioIV_counter;

architecture top of MercurioIV_counter is
	--Declaração de sinais
	signal q_out: bit_vector(3 downto 0);

begin
	--Instanciação do componente do contador
	counter_0   : work.counter
		generic map(n => 4)
		port map(clk => KEY(0), clr => KEY(1), q => q_out);

	--Instanciação do componente do decodificador para o display
	disp_0: work.MercurioIV_decod
		port map(hexa => q_out, segments => DISP0_D(6 downto 0));

end top;
```

- counter

No código de “counter” é implementada lógica do contador propriamente dito, composto por um agregado de FF-JK 's, gerados a partir da parametrização. Além

disso, são usados dois sinais, A e B para realizar a ligação entre os elementos lógicos (portas AND), e outros dois sinais *and_out* e *result* para computar as saídas.

```
-- Projeto Contador
-- Autores:
-- Bárbara Fernandes Madera   - n°: 11915032
-- Johnny Caselato Guimarães  - n°: 11915481
-- Professor: Maximiliam Luppe

entity counter is
      generic(
            n     : integer := 2
      );

      port(
            clk, clr    : in bit;
            q                 : out bit_vector(n-1 downto 0)
      );
end counter;

architecture rtl of counter is
      --Declaração de sinais
            component jk_ff
                  port(
                        jk_clk, jk_clr, j, k    : in bit;
                        jk_q                                    : buffer bit
                  );
            end component;

            signal and_out    : bit_vector(n downto 0);
            signal and_A: bit_vector(n downto 0);
            signal and_B: bit_vector(n downto 0);
            signal result     : bit_vector(n downto 0);

begin
      and_A(0) <= '1';
      and_B(0) <= '1';

      gera_jk : for i in 0 to n-1 generate
            and_out(i) <= and_A(i) and and_B(i);

            jk_0 : jk_ff port map(
                  jk_clk => clk,
                  jk_clr => clr,
                  j => and_out(i),
                  k => and_out(i),
                  jk_q => and_B(i+1)
            );

            and_A(i+1) <= and_out(i);
            result(i) <= and_B(i+1);
      end generate gera_jk;

      q <= result(n-1 downto 0);

end rtl;
```

- Circuito RTL:

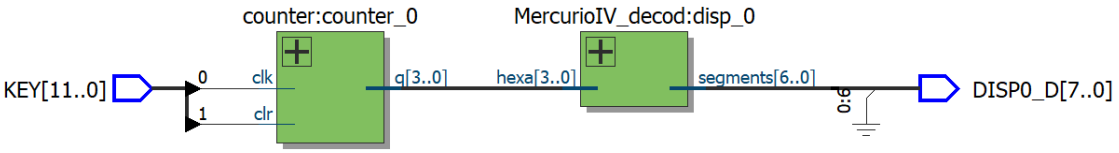


Figura 2 - Visualização RTL geral.

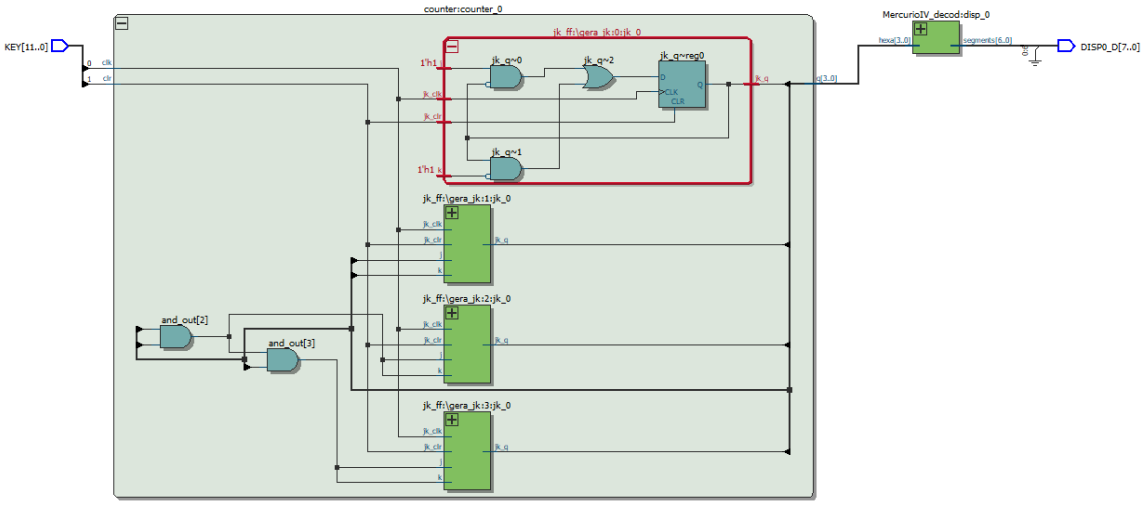


Figura 3 - Visualização RTL expandida.

- Número de células lógicas utilizadas:

| | |
|---|---|
| Quartus II 64-Bit Version | 13.1.0 Build 162 10/23/2013 SJ Web Edition |
| Revision Name | MercurioIV_counter |
| Top-level Entity Name | MercurioIV_counter |
| Family | Cyclone IV E |
| Device | EP4CE30F23C7 |
| Timing Models | Final |
| Total logic elements | 11 / 28,848 ( < 1 % ) |
| Total combinational functions | 11 / 28,848 ( < 1 % ) |
| Dedicated logic registers | 4 / 28,848 ( < 1 % ) |
| Total registers | 4 |
| Total pins | 20 / 329 ( 6 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 608,256 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 132 ( 0 % ) |
| Total PLLs | 0 / 4 ( 0 % ) |

Figura 4 - Resumo dos resultados da compilação.

## 5. Conclusão

Na implementação prática desta atividade, observou-se que os resultados obtidos no kit foram inconsistentes devido à necessidade de um *debouncer* nos botões. Ao tocar em um botão para gerar um pulso de clock, vários sinais eram computados simultaneamente, resultando em comportamento imprevisível no contador. Essa inconsistência ressalta a importância de considerar e mitigar fenômenos como o bouncing nos dispositivos de entrada, garantindo assim a estabilidade e confiabilidade das operações em circuitos digitais.