Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Laboratório de Sistemas Digitais

Bounce na Comutação de Contactos Mecânicos – Problema e Possíveis Soluções

1. Introdução

A comutação de contactos mecânicos presentes em interruptores, botões de pressão, relés e outros dispositivos similares apresenta *bounce*, isto é, a ocorrências de múltiplas transições (indesejadas) quando se realiza uma comutação OFF->ON ou ON->OFF. A figura 1 apresenta um exemplo de *bounce* capturado com um osciloscópio para uma comutação ON->OFF. No exemplo apresentado, em vez de ocorrer apenas uma transição, ocorrem sete transições. De notar que o número de transições e a largura dos impulsos (*glitches*) produzidos não é em geral fixo, nem previsível, mas normalmente da ordem dos micro-segundos. Em muitas aplicações estas múltiplas transições não são problemáticas. No entanto, há casos (e.g. quando se pretende usar esse sinal como entrada de relógio – *clock* – de um circuito, ou como *enable* de componentes com sinais de *clock* rápidos) em que estas transições possuem um efeito indesejado (e.g. provocando múltiplos incrementos incorretos de um contador).

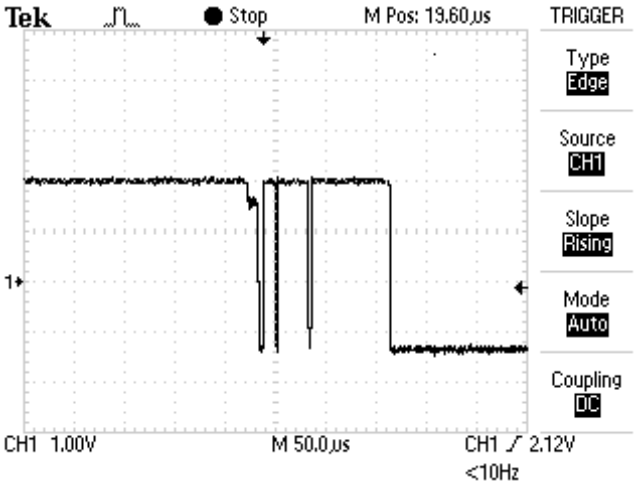

Figura 1 – Exemplo de *bounce* na transição de ON->OFF.

2. Circuito Elementar de *Debounce*

Existem várias formas de reduzir a probabilidade da ocorrência de *bounce*, a maior parte baseadas em circuitos que se adicionam à saída do contacto mecânico para produzir um impulso "limpo" (livre de *bounce*). A figura 2 ilustra uma possível arquitetura de um circuito de *debounce* baseado num *shift register*. Os sinais de *reset* dos *flip-flops* que constituem o *shift-register* são assíncronos. A entrada *dirtyIn_n* destina-se a ligar ao contacto de entrada (que apresenta *bounce*). A saída *cleanOut* disponibiliza um impulso filtrado (*debounced*). Além destes dois portos, existe ainda um sinal de entrada de *clock* responsável por efetuar o deslocamento da informação no *shift register*.

O circuito apresentado na figura 2 pode ser usado juntamente com os botões de pressão (*KEYs*) do *kit* DE2-115, pelo que se assume que a entrada *dirtyIn_n* está em lógica negativa (ativa baixa). O impulso filtrado ("limpo") produzido na saída *cleanOut* está em lógica positiva.

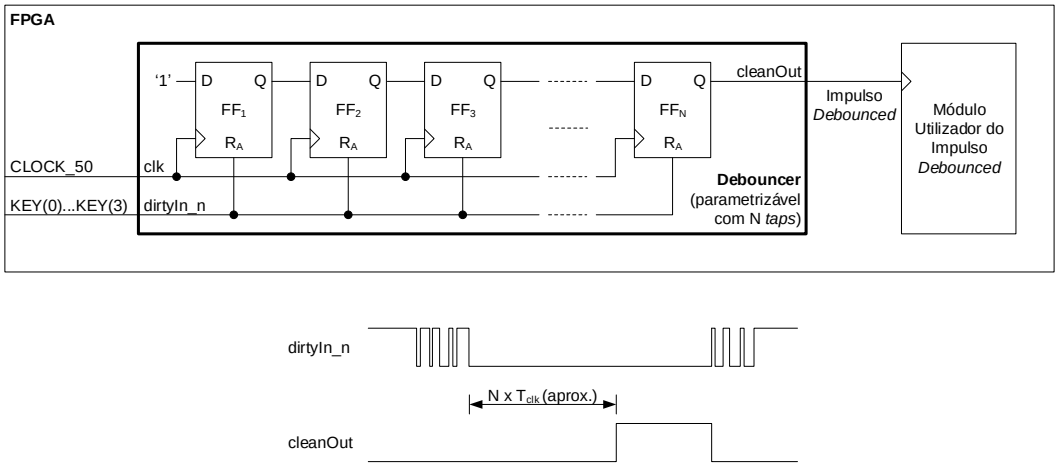

Figura 2 – Arquitetura de um circuito de *debounce* baseado num *shift register* e utilização no contexto de um projeto para o *kit* DE2-115.

Tarefas a realizar

- Analise o módulo "*Debouncer*" apresentado na figura 2 e modele-o em VHDL.
- Utilize o módulo "*Debouncer*" no contexto de um projeto do guião 4 de LSDig onde observou a ocorrência de *bounce*.
- Teste o sistema resultante para vários valores de N (e.g. 2, 20 e 200) e avalie o seu comportamento.

3. Implementação Alternativa

No seguinte endereço é disponibilizada uma arquitetura de um circuito de *debounce* potencialmente mais "poupado" em termos de recursos de implementação:
https://www.eewiki.net/display/LOGIC/Debounce+Logic+Circuit+%28with+VHDL+example%29

Por conveniência é replicada neste documento a arquitetura proposta (figura 3). Nesta arquitetura o número de bits do contador pode ser parametrizável. Interprete o circuito apresentado, modele-o em VHDL e use-o no contexto de um projeto.

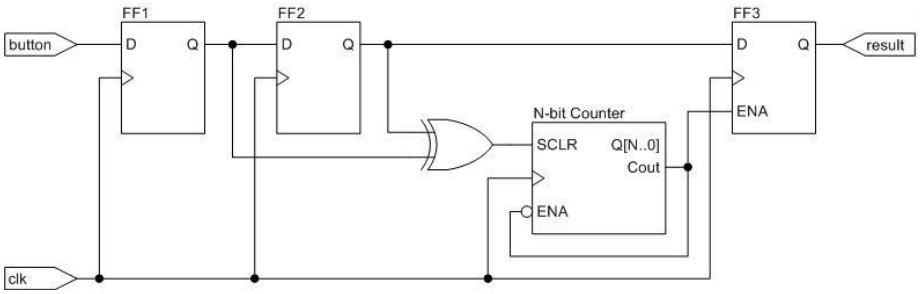

Figura 3 - Arquitetura alternativa para o circuito de *debounce* (a saída C_{out} transita para '1' quando o contador atinge o final de contagem; a porta lógica XOR é usada para detetar transições na entrada).

4. Implementação Parametrizável

Os circuitos anteriores filtram o *bounce* que pode ocorrer na entrada, produzindo na saída um impulso "limpo", atrasado relativamente à transição de entrada e com uma duração que depende do tempo que a entrada permanece ativa. No entanto, por vezes é necessário produzir um impulso livre de *bounce* mas com uma duração fixa, bem definida (tipicamente um período do sinal de relógio de referência) e que portanto não dependa da duração da ativação da entrada. Neste caso a saída do circuito de *debounce* pode também ser usada como *enable* (em vez de *clock*), permitindo utilizar o mesmo sinal de relógio em todos os componentes, mas com sinais de *enable* individuais.
O código VHDL da figura 5 descreve um d*ebouncer* parametrizável, cujo comportamento temporal é ilustrado na figura 4. A operação deste *debouncer* pode ser descrita sucintamente da seguinte forma:

- Quando ocorre uma transição ativa na entrada para o estado ativo, a saída transita também logo para o estado ativo durante um ciclo de relógio (T_{refClk}) do sinal de *clock* de referência.
- Decorrido este tempo (T_{refClk}), a saída do *debouncer* regressa ao estado inativo e o circuito ignora a entrada até decorrer o tempo *blindmSec* (figura 4), o qual deve ser superior à janela temporal em que pode correr *bounce*.
- Uma vez decorrido o tempo *blindmSec*, uma nova transição para o estado ativo da entrada volta a produzir na saída um impulso ativo com a duração T_{refClk}.

O tempo *blindmSec* destina-se precisamente a filtrar as transições espúrias da entrada.

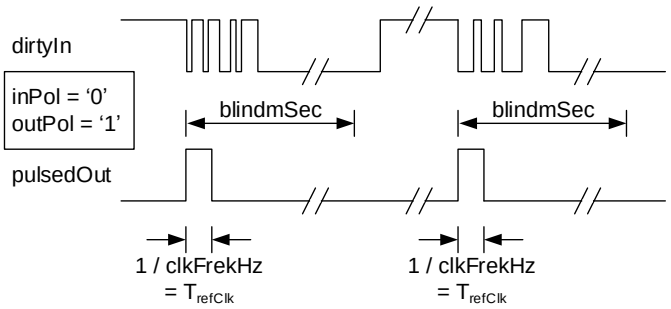

Figura 4 – Comportamento temporal do circuito de *debounce* parametrizável (assumindo que a entrada é ativa baixa e a saída é ativa alta).

O código apresentado na figura 5 pode ser encapsulado num módulo VHDL e usado de forma análoga aos *debouncers* anteriores, possuindo apenas uma entrada de inicialização adicional. Possui também os seguintes parâmetros genéricos que permite a sua adaptação e utilização em diferentes sistemas:

- **`clkFrekHz`** – frequência do sinal de *clock* de referência (em kHz).
- **`blindmSec`** – tempo que o circuito ignora a entrada após a ocorrência de uma transição para o estado ativo.
- **`inPol`** – polaridade da entrada ('0' para entrada ativa baixa; '1' para entrada ativa alta).
- **`outpol`** – polaridade da saída ('0' para saída ativa baixa; '1' para saída ativa alta).

No exemplo mostrado na figura 4, **`inPol`** = '0' e **`outPol`** = '1' (estando a entrada adaptada aos botões de pressão do *kit* DE2-115). Para uma entrada controlada por um botão ou interruptor, **`blindmSec`** pode estar compreendido entre 1 e 100 (milissegundos). Caso o *clock* de referência seja proveniente do oscilador de 50 MHz do kit DE2-115, **`clkFrekHz`** deve ser 50000 (KHz).

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity Debouncer is
    generic(clkFrekHz : positive;
            blindmSec : positive;
            inPol     : std_logic;
            outPol    : std_logic);
    port(reset        : in  std_logic;
         refClk       : in  std_logic;
         dirtyIn      : in  std_logic;
         pulsedOut    : out std_logic);
end Debouncer;

architecture Behavioral of Debouncer is
    signal s_dirtyIn, s_resetPulse, s_pulsedOut : std_logic;
    signal s_debounceCnt : natural;
begin
    sync_proc : process(refClk)
    begin
        if (rising_edge(refClk)) then
            s_dirtyIn <= dirtyIn;
        end if;
    end process;

    out_proc : process(reset, s_resetPulse, s_dirtyIn)
    begin
        if ((reset = '1') or (s_resetPulse = '1')) then
            s_pulsedOut <= not outPol;
        elsif ((s_dirtyIn'event) and s_dirtyIn = inPol) then
            s_pulsedOut <= outPol;
        end if;
    end process;

    pulsedOut <= s_pulsedOut;

    timer_proc : process(reset, refClk)
    begin
        if (reset = '1') then
            s_debounceCnt <= 0;
            s_resetPulse  <= '0';
        elsif (rising_edge(refClk)) then
            if (s_debounceCnt /= 0) then
                s_debounceCnt <= s_debounceCnt - 1;
                s_resetPulse  <= '1';
            elsif (s_pulsedOut = outPol) then
                s_debounceCnt <= blindmSec * clkFrekHz;
                s_resetPulse  <= '1';
            else
                s_resetPulse  <= '0';
            end if;
        end if;
    end process;
end Behavioral;
```

Figura 5 – Código fonte completo do circuito de *debouncer* parametrizável.