Trabalho 2

Encriptação em *stream*

Este trabalho pretende desenvolver uma arquitetura em VHDL capaz de encriptar e desencriptar dados em streaming.

Este processo de encriptação utiliza um pseudo-gerador de números aleatórios, nomeadamente o LFSR, (Linear Feedback Shift-Register), para chave de encriptação.

É nomeado de pseudo-gerador pois o mesmo deixa de ser aleatório e volta a repetir uma sequência de valores aleatórios a um determinado número de vezes.

Este projeto utilizará um LFSR de 8 bits, com isso, utilizando o polinómio perfeito, irá repetir-se novamente 256ª vez.

Para encriptação em si, será feita através de uma operação XOR com o bit de dados e o bit aleatório gerado pelo LFSR. Já a sua decriptação é feita seguindo mesmo processo, porém com o bit encriptado, voltando ao dado original.

Com isto as entidades projetadas para este trabalho são as seguintes:

- LFSR:
    - Portas de entrada:
        - Clock
        - Enable
        - Reset
        - Seed
    - Porta de saída:
        - Bit aleatório
- Chiper
    - Portas de entrada:
        - Clock
        - Enable
        - Reset
        - Bit de Dados
    - Porta de saída:
        - Bit encriptado/desencriptado

Na listagem anterior é observável duas entidades, no entanto para o desenvolvimento do processo de encriptação e desencriptação será necessário a duplicação da mesma uma para o parte de encriptação e a outra para a desencriptação.

O Chiper terá na sua arquitetura uma instância da entidade LFSR, para o mesmo gerar valores aleatório.

A entidade LFSR é representado em VHDL conforme a imagem abaixo.

```
entity lfsr is
    Port (
        clk: in std_logic;
        rst: in std_logic;
        en: in std_logic;
        seed : in std_logic_vector(7 downto 0);
        out_bit : out std_logic
    );
end lfsr;
```

Entidade LFSR em VHDL

É observável que esta entidade tem 4 entradas como já mencionadas anteriormente.

“clk” - entrada de relógio para controlar a sua sincronização.

“rst” - entrada que permite o *reset* deste modulo voltando o estado para o inicial.

“en” - entrada que permite continua transmissão de valores aleatórios, caso desabilitado a mesma retorna sempre o último valor transmitido.

“seed” - este valor é um vetor de 8 bits, que pretende ser o ponto de início e alteração após cada interação do LFSR, sendo o bit aleatório de saída uma constituição desta *seed*, assim diferentes *seed’s* gerarão sequências de valores diferentes.

LFSR:

O processo LFSR segue o respetivo seguimento, começando com a primeira interação:

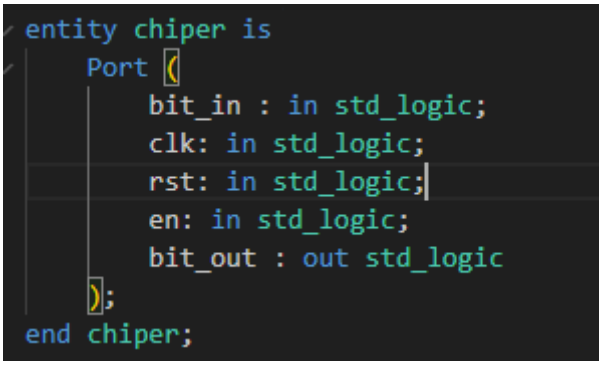
1- Colocação da seed na variável “state”
2- XOR aos bits nas casas, 7, 5, 4 e 3 e armazena o resultado em uma “variável” (fb).
    a. As casas 7, 5, 4 e 3 são escolhidas de modo formarem o polinômio perfeito
3- Shift left do state e colocação do bit calculado anteriormente na última casa do vetor.

É sempre colocado no bit de saída o último valor do vetor state.

Chiper:

A entidade *Chiper* irá receber um bit de informação/dados, operará a função XOR com o bit de saída da instância LFSR.

Abaixo podemos verificar a entidade Chiper em VHDL.



Entidade chiper em VHDL

A arquitetura do *Chiper,* implementará o componente LFSR, sendo que a *seed* necessária para este componente será definida numa constante na própria arquitetura do chiper.

Tendo já uma instância de LFSR, o Chiper consegue retornar o valor de saída através o bit de informação e o bit “aleatório” gerado pelo LFSR, operando com a função XOR.

Isto é feito a cada levantamento do relógio e caso esteja *enable*, como demonstra a imagem abaixo.

```
    constant seed : std_logic_vector(7 downto 0) := "10101010";
begin
    lsfr_inst: lfsr port map (
        rst => rst,
        en => en,
        clk => clk,
        seed => seed,
        out_bit => key_bit
    );

    process(clk, rst, en, key_bit)
    begin
        if rst = '1' then
            bit_out <= '0';
        elsif rising_edge(clk) then
            if en = '1' then
                bit_out <= bit_in xor key_bit;
            end if;
        end if;
    end process;

end Behavioral;
```

Arquitetura Chiper em VHDL

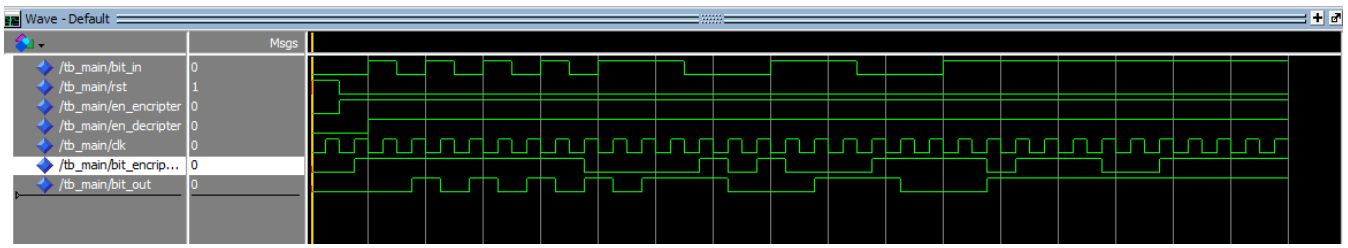
Com a implementação desenvolvimento destes componentes é possível simular um processo de *streaming* com encriptação e consequentemente desencriptação.

Para isso precisaremos, num *test-bench* instanciar duas vezes o nosso *Chiper*, pois os mesmos serão o encriptador e o desencriptador, respetivamente, onde o bit de saída já encriptado irá entrar no bit no desencriptador e devolvendo o bit original providenciado ao encriptador.

Devido a sincronização entre encriptador e desencriptador é necessário atrasar o desencriptador, para o mesmo não tentar desencriptador um valor ou inválido.

Tendo um tempo de relógio de diferença permite o desencriptador ir buscar o nomeadamente o valor anterior e não o que ainda esta a ser “processado”.

Abaixo apresenta-se uma imagem da simulação.



Simulação no software ModelSim

Podemos ver que a primeira a linha coincide antecipadamente com a última, e a penúltima linha com a informação encriptada.