PCS3335 - Laboratório Digital A - Experiência 4

por Bruno de Carvalho Albertini

18/03/2024

Na experiência 4, usaremos um registrador para transformar a função em um circuito multiciclo.

Introdução

Os circuitos sequenciais são fundamentais pois possuem uma memória do estado. Nesta experiência, você exercitará dois circuitos sequenciais fundamentais: o registrador e o contador. Leia todo o enunciado pois a forma como implementará estes dispositivos depende de decisões de projeto que devem ser tomadas conscientemente.

Experiência 4

Você deve ter seu módulo `stepfun` da Experiência 3. Este módulo, como o nome sugere, é um passo de uma função multiciclo. Este tipo de função é comum na matemática, em especial em criptografia, pois são uma maneira simples de embaralhar dados.

Nesta experiência, transformaremos a função monociclo desenvolvida na Experiência 3 em uma função multiciclo.

Preparando a função

```
entity multisteps is
    port (
        clk, rst: in bit;
        msgi :    in bit_vector(511 downto 0);
        haso :   out bit_vector(255 downto 0);
        done:    out bit
);
```

Figura 1: Diagrama para a Experiência 3

Na Figura 1 podemos ver a listagem da entidade do módulo a ser desenvolvido nesta experiência. Pare e identifique as entradas e saídas de dados e controle.

Para começarmos, o algoritmo que estamos implementando neste oferecimento é uma função de *hash* bastante conhecida, o SHA256. Esta função define duas constantes: H e K, ambas vetores de palavras de 32b. A constante H é a raiz quadrada dos oito primeiros números

primos e a constante K a raiz cúbica dos 64 primeiros números primos. Em ambos os casos, cada palavra é os primeiros 32b da parte fracionária do número primo em questão. Para facilitar, no final deste enunciado há os valores em hexadecimal para ambas as constantes.

A função multiciclo `multisteps` equivale a 64 execuções da função `stepfun`. Cada saída de `ao` à `ho` do módulo é reinjetada na própria função nas entradas de `ai` a `hi`. No entanto, o módulo `stepfun` tem uma entrada `kpw` e ainda precisamos de um valor inicial para as entradas de `ai` a `hi`.

Cálculo do W e `kpw`

O valor `kpw` a ser fornecido para o módulo em cada uma das 64 execuções é calculado da seguinte maneira:

- $W_t = M_t$ para os primeiros 16 valores de 32b, o que equivale à entrada `msgi` de 512 bits. Note que a entrada é ordenada `downto`, ou seja, o W_0 é o equivalente a `msgi(31 downto 0)` e o W_{15} equivale a `msgi(511 downto 480)`.
- $W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$ para os demais W, de 16 a 63.

O vetor W possui então 64 posições de 32b, sendo as 16 primeiras equivalentes à mensagem de entrada e as demais equivalentes ao cálculo acima. Para o cálculo, use as funções `sigma0` e `sigma1` da Experiência 2 e o somador de 32b da Experiência 3.

Para cada uma das 64 execuções do módulo `stepfun` necessárias para o módulo `multisteps`, a entrada `kpw` é a soma de W_t e K_t. Exemplo: para a primeira execução, `kpw`=$W_0 + K_0$, ou seja, `msgi(31 downto 0)+x"428a2f98"`.

Valores iniciais e finais

O segundo problema é que os valores iniciais de `ai` a `hi` são sempre iguais aos valores de H. Exemplo: `ai=6a09e667` para a primeira execução.

O valor final do algoritmo, após as 64 execuções, é dado pelas saídas `ao` a `ho` do módulo `stepfun`, adicionadas do H equivalente, usado como valor inicial. Exemplo: saída final para `a` é `haso(31 downto 0)=ao+x"6a09e667"`. Repare que o `haso` também é `downto` e a saída tem 256b, o que equivale a 8 palavras de 32b (de `ao` até `ho`).

Montagem

Monte o seu módulo `multisteps` com as constantes, com o seu `stepfun` e com um contador com as seguintes premissas:

- No *reset* as entradas do `stepfun` são *H* e `done` é baixo.
- Ao desabilitar o *reset* o contador conta as 64 execuções do algoritmo, reinjetando as saídas do `stepfun` na sua própria entrada, mas atualizando o `kpw` para o calculado para aquele passo.
- Ao final dos 64 passos, `done` é alto e o valor final é dado pelas saídas somadas dos *H* correspondentes. A saída deve permanecer fixa até o próximo *reset*, assim cono o `done`.

Para esta montagem, você deve considerar a mensagem como a repetição dos valores nas chaves de 7-0. O valor de 8b deve ser repetido 64 vezes para formar a palavra de entrada de 512b. A saída deve ser somente nos *displays* e corresponde aos 6 dígitos hexadecimais menos significativos do valor final (da palavra `a`). Use um dos LEDs para mostrar o `done` e o *reset* deve ser um botão. Seu *clock* deve ser proveniente do *clock* da placa de 50MHz.

Você pode dividir o *clock* da placa, mas **jamais** use um botão para *fazer* um *clock*.

A submissão para o juiz é do módulo `multisteps` conforme entidade mostrada na Figura 1. O juiz contém uma implementação válida do `stepfun` e de todas as funções desenvolvidas até o momento, portanto não inclua-as, mas obviamente use-as.

Um caso de teste padrão é as chaves todas em zero, o que deve retornar após 64 passos o valor hexadecimal 5698be nos *displays*.

Observações

Não menospreze esta experiência, a inserção de registradores complica o circuito. Faça um *textbench* temporizado para testar o circuito.

Exemplo de *testbench* aqui.

Citamos dois elementos sequenciais que você deve usar. O contador tem uso óbvio para contar o número de execuções. Você pode fazer um módulo contador separado ou usar um `proccess` e `variable` para embutir um no seu projeto. Caso opte por fazer separado, você deve incluí-lo na sua solução. Contadores também podem ser usados para diversas outras finalidades, inclusive neste projeto.

Os registradores são menos óbvios. Você usará registradores para o vetor *W* e nas saídas de `ao` a `ho` do módulo `stepfun`, para ser capaz de reinjetar as saídas nas entradas. Assim como o contador, você pode usar `proccess` e atribuição incompleta para embutir um registrador no seu projeto ou fazer um módulo separado, que também deverá incluir.

Desafio

Seu projeto deve usar menos de 10% da capacidade de FPGA em termos de células lógicas. Caso opte por fazer este desafio, ele deve ser apresentado no dia da experiência para o seu professor.

Constante K, onde K_0=`428a2f98`, K_1=`71374491`, K_7=`ab1c5ed5`, K_8=`d807aa98` e K_{63}=`c67178f2`.

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

Constante H, onde H_0=`6a09e667`, H_1=`bb67ae85` e H_7=`5be0cd19`.

```
6a09e667 bb67ae85 3c6ef372 a54ff53a 510e527f 9b05688c 1f83d9ab 5be0cd19
```