Implementação em VHDL da geração de subchaves para o algoritmo DES

Diogo de Campos - digo.campos@terra.com.br
João Paulo Pizani Flor - joaopizani@inf.ufsc.br
Jucemar Luis Monteiro - jucemar20@yahoo.com

1 Introdução

O algoritmo DES foi muito utilizado como o padrão de criptografia até 1998, ou seja, até o surgimento do AES. Hoje uma chave DES pode ser quebrada em pouco mais de 20 horas usando outros algoritmos, ou alguns dias através de força bruta. Inclusive a busca de chave mais rápida por força bruta atualmente é implementada utilizando um sistema massivamente paralelo com FPGA's.

Ele se estrutura basicamente em dezesseis passos, ou rodadas, de encriptação que trabalham sucessivamente sobre o bloco de texto a ser encriptado. Cada uma dessas rodadas utiliza uma *subchave* como parâmetro para uma função de encriptação.

Uma boa parte do algoritmo consiste então em, a partir de uma chave dada, gerar as dezesseis subchaves que serão usadas para encriptar o bloco de texto. Para gerar as subchaves, o algoritmo parte de uma chave e a permuta a partir de uma matriz predefinida, depois o resultado obtido é dividido em dois, e em cada metade é aplicada uma rotação para a esquerda, ou seja, o elemento mais da esquerda se torna o último. Após este passo, a subchave passa por uma última permutação, e assim é formada a primeira subchave. As próximas subchaves são geradas a partir da anterior, logo após a rotação para a esquerda. Para dar maior segurança, algumas chaves são rotacionadas duas vezes e não somente uma. Este processo de geração de chaves foi o foco deste trabalho, que utilizou da linguagem de descrição de hardware VHDL e visou a implementação em um dispositivo de lógica programável (FPGA).

2 Solução adotada

Para atingir uma solução eficiente para o problema, desenvolvemos um projeto em nível RT que consiste de um bloco de controle e um bloco operativo.

A nossa solução tem somente uma saída para as subchaves, e consequentemente, as subchaves são geradas uma de cada vez, esta idéia foi adotada porque diminuiu muito o custo do bloco operacional e o período do relógio, já que cada subchave depende de uma anterior para ser gerada, o que criaria uma estrutura em série (chaining) ineficiente. Além disso, a outra parte do algoritmo que utiliza estas subchaves usa somente uma por período de relógio, e então não faria sentido gerar todas as chaves de uma só vez.

O bloco de controle do sistema digital é uma máquina de Moore com 7 estados, sendo um deles o estado de espera, e um o estado final. Ele controla algumas funções do bloco operativo, incluindo um Mux (que vai decidir se a rotação vai ser aplicada sobre a chave inicial ou uma das chaves anteriores) e um controle para o rotacionador (rotate left), que vai decidir se a chave deve ser girada uma ou duas vezes. O número de rotações de cada subchave está descrito na seguinte tabela:

subchave	rotações	subchave	rotações
1	1	9	2
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	1	16	1

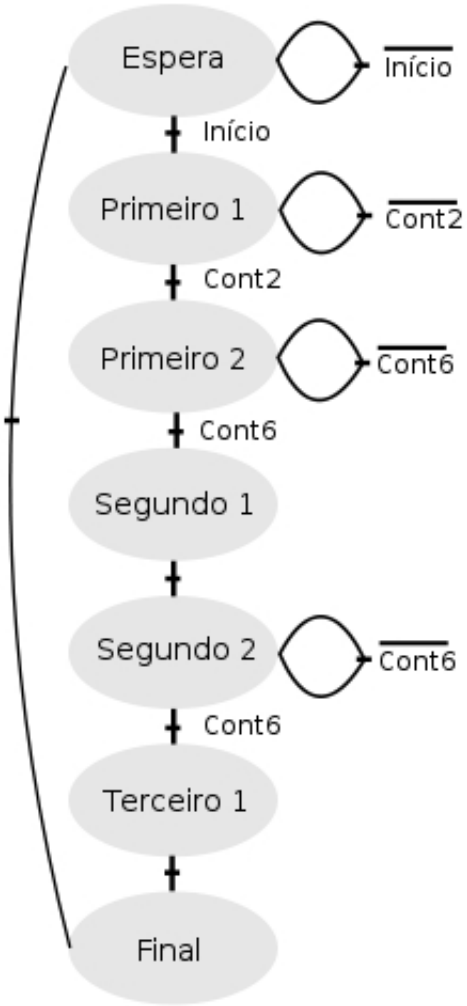
A partir da tabela de rotações podemos ver que são necessários mais 5 estados (além de espera e fim), gerando a seguinte tabela de saídas para os estados do bloco de controle:

Estado	n	selMux	Fim
Espera	0	0	0
Primeiro 1	0	1	0
Primeiro 2	1	1	0
Segundo 1	0	1	0
Segundo 2	1	1	0
Terceiro 1	0	1	0
Final	0	1	1

(n = 0 implica em uma rotação, n = 1 implica em duas rotações)

Para isso, utilizamos um contador de 2 ciclos e outro de 6 ciclos para manter o bloco de controle em um certo estado pelo número desejado de rodadas. Aqui está um diagrama de estados do bloco de controle implemen-

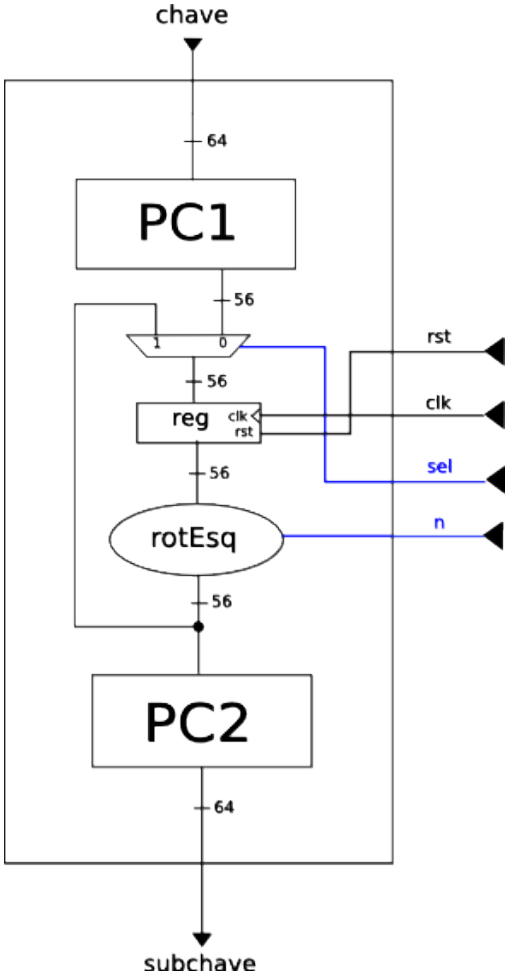
tando os contadores para efetuar a troca de estados:



Atual	Início	cnt2	cnt6	Próx. Estado
Espera	0	X	X	Espera
	1	X	X	P1
P1	X	0	X	P1
	X	1	X	P2
P2	X	X	0	P2
	X	X	1	S1
S1	X	X	X	S2
S2	X	X	0	S2
	X	X	1	T1
T1	X	X	X	Final
Final	X	X	X	Espera

O sistema digital que implementa esta parte do DES tem como entradas uma chave inicial de 64 bits, um sinal de relógio, um reset assíncrono e um sinal "início". Suas portas de saída são subchave, com 48 bits e o sinal fim, o qual fica em '1' no ciclo de relógio que sucede a geração da última subchave.

A parte operativa do sistema consiste de um registrador, um MUX, um rotacionador de bits e dois blocos que realizam permutações pré-definidas sobre vetores de bits. Este bloco tem como sinais externos o sinal de clock e reset, além dos sinais provindos do bloco de controle, que são o selecionador do MUX e um sinal "n" que é o que controla o número de rotações para a esquerda do vetor de bits atual (subchave). Depois de passar pelas permutações e rotações, a chave gerada vai para a saída do sistema, para ser eventualmente utilizada por outro bloco, que continuará a encriptar o texto.



3 Resultados da Síntese

A síntese dos blocos foi feita utilizando um dispositivo Stratix II da Altera, utilizando o Quartus II como ferramenta de síntese. Os dados obtidos foram os seguintes:

Resultado	BO	BC
ALUTS	56	14
Pinos	116	6
Registradores	1	11
tco	7.369ns ()	5.780ns
tsu	3.611ns	2.849ns
tho	0.036ns	-2.610ns
tpd	9.707 ns	

A partir dos dados obtidos podemos perceber que a frequência máxima do sistema, que é o inverso do atraso crítico de propagação, fica próxima de 100MHz, ou seja, podemos gerar 100 milhões de subchaves por segundo (uma a cada ciclo de relógio).

4 Resultados da Validação

Em anexo estão as simulações de onda do bloco de controle e do sistema completo. Segue uma descrição sucinta da estratégia de simulação adotada:

A simulação do bloco de controle tem um único caso de funcionamento sendo testado, porém com cobertura maior que 90%. Isso porque o bloco de controle só tem um modo de funcionamento. A única entrada que influencia é início, e após início, a FSM segue sempre a mesma seqüência de estados. Os sinais cont2 e cont6 são internos do bloco de controle, e apenas servem para determinar o número de vezes em que a FSM permanece em um mesmo estado.

Já a simulação do bloco operativo é bem mais desafiadora: Como temos uma chave de 64 bits como entrada, teríamos um *espaço de entrada* de $2^{65} - 1$; ou seja, para termos completa certeza do funcionamento do BO, teríamos que realizar $2^{65} - 1$ casos de teste, um para cada entrada. Realizamos a simulação com uma entrada randômica provinda de exemplos ilustrativos do algoritmo presentes em [1]. Além disso realizamos a simulação com as entradas em que todos os bits nas posições múltiplas de 8 eram '1'. Isso mostrou que o algoritmo de *key-scheduling* ignora estes bits no processo: de fato, eles são eliminados já na primeira permutação (PC1).

5 Conclusão

Com este trabalho, pudemos analisar como funciona um dos mais importantes métodos de encriptação, que apesar de ser relativamente inseguro, é usado (em variações como o TripleDES) até hoje. Procuramos implementar o bloco de geração de subchaves se maneira que ele pudesse se integrar mais facilmente a outros blocos implementando o DES como um todo.

Tivemos com esse trabalho a oportunidade de colocar em prática todo o ciclo de projeto de um sistema digital completo. Desde a fase de especificação, passando pela descrição usando uma HDL, até a síntese, mapeamento para um dispositivo de lógica programável e verificação funcional e de atrasos.

6 Referências

1) BROWN, Stephen; VRANESIC, Zvonko. *Fundamentals of digital logic with VHDL design, 2nd edition*. New York: McGraw-Hill, 2004.
2) ASHENDEN, Peter J. *The student's guide to VHDL*. San Francisco: M. Kaufmann, 1998.
3) http://www.aci.net/kalliste/des.htm
4) http://en.wikipedia.org/w/index.php?title=DES_supplementary_material&oldid=146382906
5) http://en.wikipedia.org/w/index.php?title=Data_Encryption_Standard&oldid=172344919