Aluno: Jonatan Felipe Hartmann
Disciplina: EEL7123-08235 - Tópicos Avançados em Sistemas Digitais

Otimização da operação de exponenciação em vetores de 4 bits

Neste trabalho, buscou-se otimizar a operação de exponenciação em um vetor de 4 bits, sendo x^2, x^3 e x^4 as operações abordadas. Além disso, também pensou-se em analisar a progressão do número de vetores finais necessários para realizar estas operações.

Dessa forma, foram utilizadas duas maneiras para tal otimização, sendo:

a) **Distribuição de bits**: consiste em juntar dois bits iguais e colocar uma representação sua na próxima posição, como ilustrado na tabela abaixo:

2^{n+1}	2^n
	x
	x
x	

b) **Combinação de bits utilizando tabela verdade**: permite que bits semelhantes em uma posição sejam analisados combinacionalmente a fim de otimizar sua soma, várias combinações foram usadas, as tabelas abaixo ilustram as principais:

2^{n+1}	2^n
	xyz
	yz
xyz	x'yz

2^{n+1}	2^n
	xyz
	yz
xyz	x'yz

2^n
xy'z
yz
yz + xz

Além disso, foi também analisada a possibilidade de usar a **Recodificação de Booth** para otimização, entretanto, como a recodificação faz uso de bits negativos, deveria ser utilizado complemento de 2 para sua representação, aumentando consideravelmente o atraso para calcular a operação. Logo, foi necessário descartar essa alternativa.

Dessa maneira, ao combinar essas otimizações, foi possível chegar nas seguintes otimizações:

a) **x^2:** $x_3x_2x_1x_0$

2^6	2^5	2^4	2^3	2^2	2^1	2^0
34	24	14	13	12	0	1
4	23	2'3	0	2	0	0

b) x^3:

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
34	234	124	124	123	13	12	13	13	12	1
0	4	134	14	3 + 14	124	14	14	0	0	0
0	124	0	23+43	24	0	23 + 13	2	0	0	0

c) x^4:

2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
234	1234	2'34	124	123	1'34	1'23	12'3+1'234	124'	14	123+13	0	0	0	1
0	34	24	34	12'34	1'234+12'34	13'4	13+14	23+13	12'3	2	0	0	0	0
0	0	12'34	14	23'4	23+34	23'4	23+124	23	234+134	0	0	0	0	0
0	0	4	0	34	23'4	3	123+134	234+134	234+134	0	0	0	0	0

OBS: Nas tabelas, cada índice corresponde à posição do dígito binário, sendo $x_4x_3x_2x_1$. Além disso, uma aspa simples (') após um índice corresponde à sua negação, sendo por exemplo: 1' = not(x_1).

Após otimizar as tabelas, notou-se que para um binário de 4 bits, a quantidade n de vetores a serem somados para elevar esse número à uma certa potência, é exatamente o mesmo número que o expoente determinado, ou seja:

$$n = b$$

sendo X um número de **4 bits** com:

$$X^{b}$$

Apesar desse resultado, ao comparar com o resultado da operação ao cubo de um número de 5 bits (dado durante a disciplina), nota-se que para um número **Y de 5 bits** a progressão não é a mesma. Isso foi considerado visto que para Y^2 é preciso 3 vetores, e para Y^3 é necessário 5.

O repositório contendo a tabela otimizada, os VHDLs descritos, e os scripts em Python para o teste da otimização estão disponíveis em: https://github.com/hartmannjonatan/optimized-digital-exponentiation.