FEDERAL DE ENGENHARIA DE ITAJUBÁ

MULTIPLICADOR BINARIO SERIAL "PIPELINE" DE 8 BITS PARA APLICAÇÕES EM VLSI

Edwin Antonio Cuadros Sánchez

ITAJUBÁ - MG - 1998

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

DISSERTAÇÃO DE MESTRADO

MULTIPLICADOR BINÁRIO SERIAL "PIPELINE" DE 8 BITS PARA APLICAÇÕES EM VLSI

Dissertação apresentada à Escola Federal de Engenharia de Itajubá, como parte dos requisitos à obtenção do título de Mestre em Engenharia Elétrica.

Autor : *Edwin Antonio Cuadros Sánchez*

Orientador : *Prof. Dr. Laércio Caldeira*

Aos meus pais, David e Elizabeth,
e à Cidinha, dedico.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Laércio Caldeira pela orientação com que me brindou durante todo este trabalho e a todos os engenheiros do Grupo de Microeletrônica da EFEI pela ajuda oferecida.

RESUMO

Os "chips" VLSI (Very Large Scale Integration) vêem sendo incorporados a todo tipo de produto eletrônico, tornando o projeto de sistemas nesta tecnologia de grande interesse. Este trabalho propõe o projeto de um multiplicador serial de 8 bits do tipo "pipeline", para aplicações em VLSI. É baseado na Célula de Multiplicação de Cheng e no Algoritmo de Booth Modificado. A Célula de Cheng foi modificada neste trabalho visando maior performance, área reduzida e grande confiabilidade. A utilização do Algoritmo de Booth Modificado assegura a modularidade das células de multiplicação, permitindo a operação com palavras maiores que 8 bits. Utilizando-se o processo CMOS de 1 μm da ES2, conseguiu-se integrar o multiplicador em uma área de apenas 3,8 x 2,3 mm^2. O projeto proposto indica a correta operação do multiplicador em frequências de "clock" de até 125 MHz, o que conduz a taxas superiores a 2,7 milhões de multiplicações por segundo.

ABSTRACT

The VLSI chips have been incorporated to every kind of electronics product which makes the project of systems in this technology of great interest. This work is a proposal of an 8-bit serial pipelined multiplier for VLSI applications. It is based on Cheng's cell and on the Modified Booth's Algorithm. The Cheng's cell was modified in this work in order to increase its performance, confiability and to reduce the area. The Modified Booth's Algorithm assures the modularity of the cells, allowing operations with words bigger than 8-bits. One of the goals of this work is the little consumption of area, therefore attained that the multiplier layout takes up an area of 3,8 x 2,3 mm^2. This project allows that the multiplier works in frequencies of clock up to 125 MHz, which leads it to rates higher than 2,7 millions of multiplications per second, using CMOS 1μm technology.

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO AOS MULTIPLICADORES DIGITAIS

1.1 A Importância Dos Multiplicadores 1-1
1.2 A Multiplicação Binária 1-1
1.3 Considerações Sobre os Multiplicadores Binários 1-3
1.4 O Multiplicador Paralelo 1-3
1.5 O Multiplicador Serial 1-6
1.6 O Multiplicador Serial / Paralelo 1-8

CAPÍTULO 2

ALGORITMOS DE MULTIPLICAÇÃO BINÁRIA

2.1 Introdução 2-1
2.2 O Método de Multiplicação Direta 2-1
2.3 Método Burks - Goldstine - von Neumann 2-4
2.4 Primeiro Método de Robertson 2-8
2.5 Segundo Método de Robertson 2-10
2.6 O Método de Booth 2-11
2.7 Outros Algoritmos de Multiplicação 2-15
2.7.1 Multiplicador com Salvador de "Carry" 2-15
2.7.2 A Árvore de Wallace 2-17
2.8 O Algoritmo de Booth Modificado 2-19

CAPÍTULO 3

A CÉLULA BÁSICA DE MULTIPLICAÇÃO

3.1 Implementação do Multiplicador Serial 3-1
3.2 Comparação entre a Célula de Cheng e a Célula Modificada 3-4
3.3 Análise da Célula Básica de Multiplicação 3-5

CAPÍTULO 4

IMPLEMENTAÇÃO DO MULTIPLICADOR, "LAYOUT" E TESTES

4.1 Introdução 4-1
4.2 Temporização dos Flip - Flops 4-1

4.3 O Circuito Completo 4-7
4.4 O Bloco de Controle 4-8
4.4.1 O Gerador de Fases 4-9
4.4.2 O Bloco Reset 4-10
4.4.3 O Bloco Set_PS 4-12
4.4.4 Contador Síncrono de 5 Bits 4-12
4.4.5 O Bloco CTRL 4-13
4.4.6 O Bloco HLD 4-14
4.5 Registrador Paralelo-Série de 8 Bits 4-15
4.6 Registrador Série-Paralelo de 16 Bits 4-15
4.7 O "Layout" 4-18
4.8 Simulações com Capacitâncias Parasitas 4-19
4.9 Testes 4-21
4.9.1 Testes em Baixa Frequência 4-21
4.9.2 Testes em Alta Frequência 4-25

CAPÍTULO 5

CONSIDERAÇÕES E CONCLUSÕES FINAIS

5.1 Considerações 5-1
5.2 Conclusões 5-3

ANEXO

COMPARAÇÃO ENTRE AS TÉCNICAS "FULL CUSTOM" E "FPGA"

1. Considerações Sobre FPGA's A-1
2. Resultados Obtidos com a Técnica "FPGA" A-1
3. Observações A-2

BIBLIOGRAFIA REFERENCIADA E RECOMENDADA

CAPÍTULO 1

INTRODUÇÃO AOS MULTIPLICADORES DIGITAIS

1.1 A Importância Dos Multiplicadores.

O multiplicador é uma parte essencial de qualquer circuito de processamento digital de sinal, já que muitos cálculos numéricos podem ser reduzidos pela multiplicação. Por esta razão, um dos blocos básicos mais utilizados no mundo do processamento digital de sinais é o multiplicador.

Principalmente pelas razões citadas acima o multiplicador digital foi o tema escolhido para este trabalho. Embora o multiplicador paralelo seja sempre o mais procurado pela rapidez do processamento, o multiplicador serial tem sua aplicação garantida quando se busca um compromisso entre performance e baixo consumo de área.

1.2 A Multiplicação Binária.

A forma mais elementar de multiplicação é a multiplicação de dois números binários positivos. A solução desta multiplicação pode ser encontrada através da tradicional técnica de adições e deslocamentos sucessivos.

Para exemplificar, a multiplicação de dois números binários positivos tais como 1011 e 0111 (equivalentes aos decimais 11 e 7, respectivamente) pode ser realizada como mostrado na Tab. 1.1.

				1	0	1	1	Multiplicando (11_{10})
				0	1	1	1	Multiplicador (7_{10})
				1	0	1	1	Produtos parciais
			1	0	1	1		devidamente
		1	0	1	1			deslocados
	0	0	0	0				
0	1	0	0	1	1	0	1	Produto final (77_{10})

Tab. 1.1 Multiplicação de Dois Número Binários Positivos.

Pode-se perceber que o processo de multiplicação consiste em três etapas :

1) Cálculo dos produtos parciais;
2) Acumulação dos produtos parciais deslocados;
3) Soma dos produtos parciais para se achar o produto final.

O cálculo dos produtos parciais consiste em se efetuar a operação lógica "E" entre o multiplicando e o equivalente bit do multiplicador. Logo, são acumulados os produtos parciais devidamente deslocados e, finalmente, estes são somados para se obter o resultado final da multiplicação **[4].**

Há diversas técnicas para se realizar uma multiplicação sendo que estas técnicas estão relacionadas com velocidade, precisão numérica, área de integração e outros fatores.

1.3 Considerações Sobre Os Multiplicadores Binários.

De uma forma geral os multiplicadores podem ser classificados, segundo **Weste e Kamram [4]**, da seguinte maneira:

a) Multiplicador Paralelo;
b) Multiplicador Serial;
c) Multiplicador Serial / Paralelo.

Em termos de velocidade o multiplicador paralelo é o mais veloz dos multiplicadores . Como a velocidade do multiplicador serial é dada por **mxn** ciclos de "clock" (**m** é o número de bits do multiplicador e **n** é o número de bits do multiplicando) e a velocidade do multiplicador serial/paralelo é dada por **m+n** ciclos de "clock", pode-se observar que o multiplicador serial é o mais lento dos multiplicadores.

Em termos de *área de integração*, quem utiliza a maior área é o multiplicador paralelo enquanto que o multiplicador serial é o que ocupa a menor. Este é um dos motivos que torna o multiplicador serial tão atrativo em algumas aplicações.

Pode-se concluir, embora não de uma forma criteriosa, que a velocidade de um multiplicador está intimamente relacionada com a sua área de integração, ou seja, quanto mais veloz o multiplicador, mais área tende a ocupar.

1.4 O Multiplicador Paralelo.

A maneira mais veloz de se efetuar o produto de dois números envolve a geração dos produtos parciais e a soma destes, da melhor forma possível, para se conseguir o resultado final. Logo, o principal objetivo da multiplicação paralela é o de se encontrar :

1) A melhor forma para se somar os produtos parciais e, assim, chegar ao resultado.
2) Métodos que reduzam o número de produtos parciais.

Os produtos parciais podem ser computados em paralelo de uma forma independente, fato que torna este tipo de multiplicador, o mais veloz.

A representação em complemento de dois do multiplicando X e do multiplicador Y **[1]** pode ser dada como:

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \qquad (1\text{-}1)$$

$$Y = -y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i \qquad (1\text{-}2)$$

e do produto P por:

$$P = -p_{m+n-1}2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = YX \qquad (1\text{-}3)$$

onde:

- 2^{n-1} , 2^{m-1} , 2^{m+n-1} , 2^i representam a posição do bit.
  Por exemplo, 2^0 representa o bit menos significativo (LSB).
- p_i são as denominadas somas parciais.

Pode-se perceber que há **mxn** somandos parciais, como mostra a Tab. 1.2 (**m** é o número de bits do multiplicador e **n** é o número de bits do multiplicando), que são produzidos em paralelo por um conjunto de **mxn** portas "E", como mostrado nas Figs. 1.1 e 1.2.

Como exemplo, a multiplicação de dois números binários de quatro bits pode ser escrita da forma mostrada na Tab. 1.2 .

				X_3	X_2	X_1	X_0	Multiplicando
				Y_3	Y_2	Y_1	Y_0	Multiplicador
				X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0	
			X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1		Produtos
		X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2			Parciais
	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3				
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	Produto Final

Tab. 1.2 Matriz de Produtos Parciais Deslocados.

A Fig. 1.1 mostra uma célula que pode ser utilizada para se construir um multiplicador paralelo **[4]**. O termo X_i se propaga verticalmente enquanto que o termo Y_i o faz horizontalmente. Os produtos parciais que chegam à célula entram pelo centro de seu topo e o "carry" de entrada, pela direita do topo. A lógica "E" é realizada dentro da célula e a soma é levada à próxima célula pelo centro da parte inferior. Finalmente o "carry" de saída é enviado à próxima célula pela extremidade inferior esquerda.

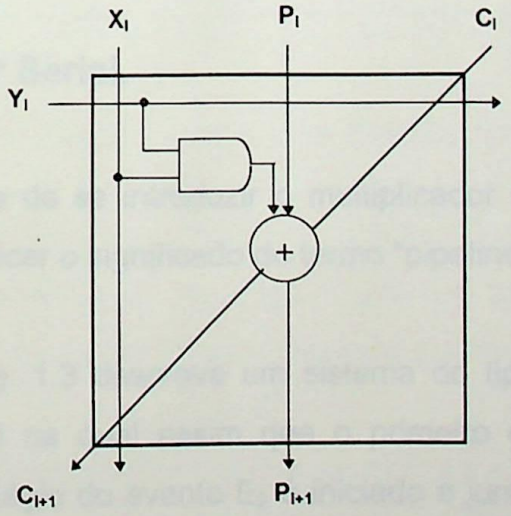


Fig. 1.1 Célula de um Multiplicador Paralelo.

O projeto da célula deste multiplicador é direto levando-se em consideração que o somador deve possuir tempos iguais de propagação para a soma e para o "carry".

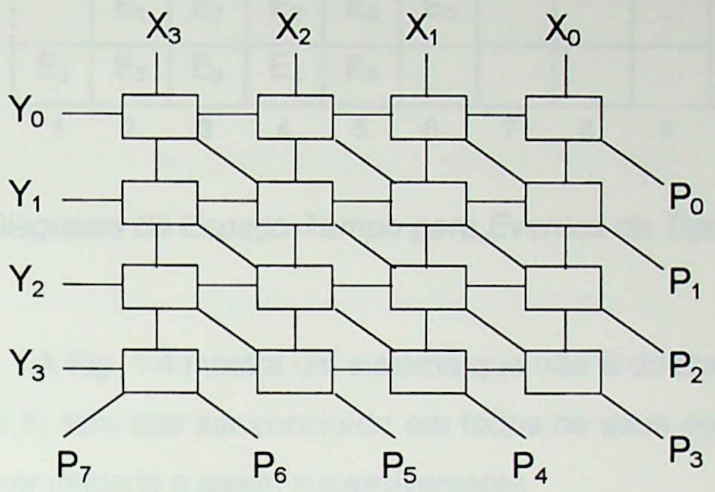


Fig. 1.2 Arranjo de um Multiplicador Paralelo.

A Fig. 1.2 mostra como ficaria o arranjo do exemplo anterior de uma multiplicação de dois números binários de quatro bits, já apresentado em forma quadrada, o que facilita grandemente a sua implementação.

1.5 O Multiplicador Serial.

Antes de se introduzir o multiplicador serial, para efeito deste trabalho, deve-se esclarecer o significado do termo "pipeline".

A Fig. 1.3 descreve um sistema do tipo "pipeline" que possui uma operação periódica na qual assim que o primeiro estágio do evento E_1 é concluído, o primeiro estágio do evento E_2 é iniciado e junto com ele é realizado o segundo estágio do evento E_1 e assim, sucessivamente, até que os eventos sejam concluídos em sua totalidade **[2]**.

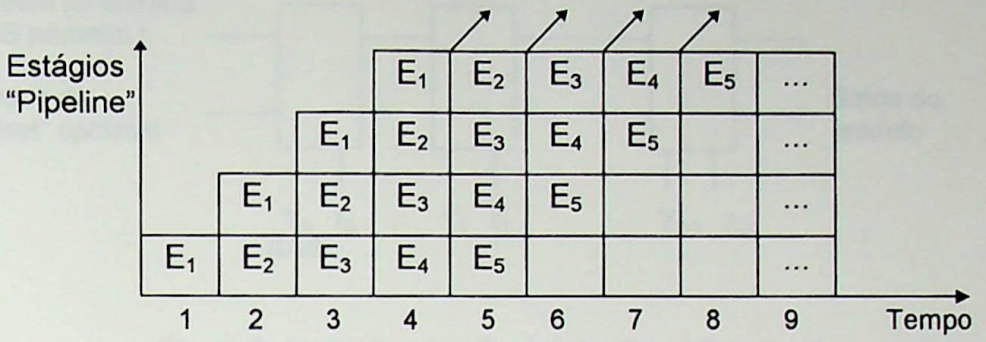


Fig. 1. 3 Diagrama de Espaço-Tempo para Eventos do Tipo "Pipeline".

A Fig. 1.4 mostra um sistema que não é do tipo "pipeline". Neste sistema, o evento E_1 tem que ser concluído em todos os seus estágios para que o evento E_2 possa ser iniciado e assim sucessivamente.

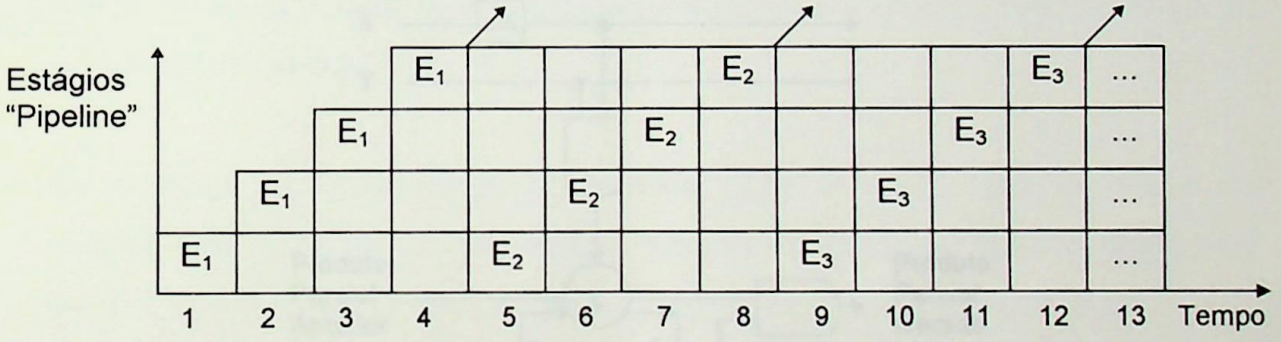


Fig. 1.4 Diagrama de Espaço-Tempo para Eventos do Tipo Não "Pipeline".

Observando-se as Figs. 1.3 e 1.4 pode-se concluir que um sistema do tipo "pipeline" é muito mais veloz que um sistema do tipo não "pipeline".

A implementação de um multiplicador serial do tipo "pipeline" está ilustrada na Fig. 1.5. Ele consiste de K módulos, com interconexões seriais para a palavra de dado de entrada e para os produtos parciais **[3]**. Os sinais utilizados para "reset" são aplicados em paralelo (r_i).

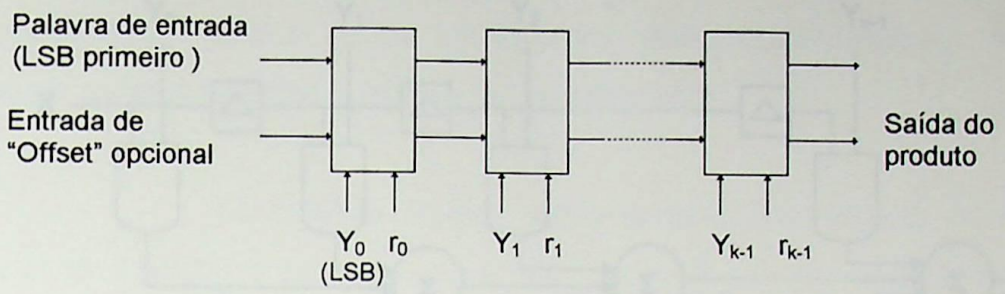


Fig. 1.5. Multiplicador Serial do Tipo "Pipeline".

O interior de cada módulo está ilustrado na Fig. 1.6 que indica como é feita a operação da multiplicação. A palavra do Multiplicando X passa por uma operação "E" com cada bit particular do Multiplicador Y e logo é realizada uma adição, gerando-se um produto parcial.

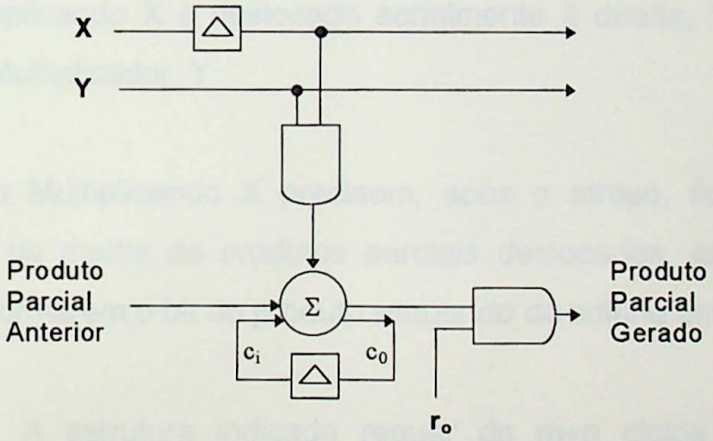


Fig. 1.6 Interior de um Módulo do Multiplicador Serial.

1.6 O Multiplicador Serial / Paralelo.

A Fig. 1.7 ilustra um multiplicador serial/paralelo que pode ser implementado com base no multiplicador serial **[3]**.

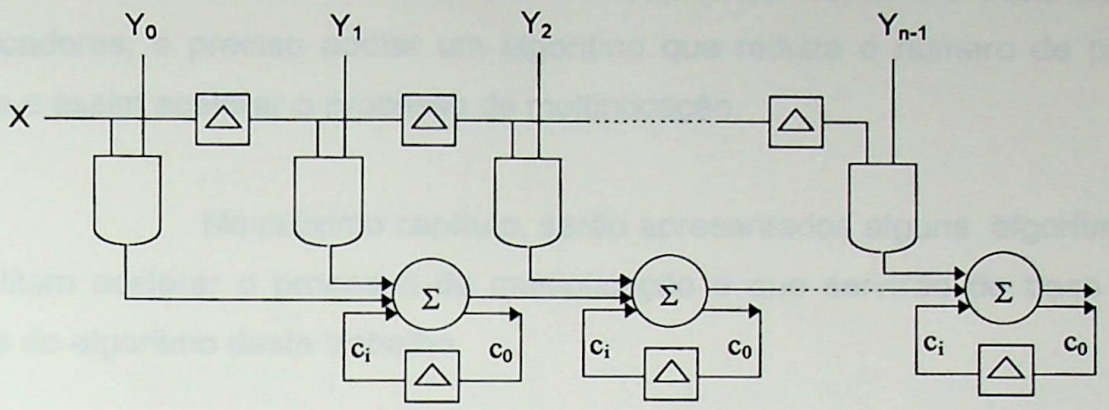


Fig. 1.7 Estrutura de um Multiplicador Serial / Paralelo.

Com a estrutura da Fig. 1.7 a multiplicação é realizada da seguinte maneira:

- O deslocamento dos números é conseguido com um elemento de atraso. Logo, um bit do Multiplicando X é deslocado serialmente à direita, "gatilhando" o bit apropriado do Multiplicador Y.

- Os estágios do Multiplicando X precisam, após o atraso, ficar alinhados na mesma coluna da matriz de produtos parciais deslocados, quando então são somados para formarem o bit do produto requerido da coluna em questão.

A estrutura indicada requer de **m+n** ciclos de "clock" para produzir o produto final. A principal limitação deste multiplicador é que a frequência máxima de operação é proporcional ao tempo de propagação do sinal através do arranjo de somadores.

Neste trabalho, optou-se pelo desenvolvimento do multiplicador binário serial pelo fato dele ocupar menos área que os demais e pela sua vantajosa modularidade, que permite ajustá-lo para trabalhar com palavras de vários comprimentos.

Tendo em vista que o multiplicador serial é o mais lento dos multiplicadores, é preciso adotar um algoritmo que reduza o número de produtos parciais e assim acelerar o processo de multiplicação.

No próximo capítulo, serão apresentados alguns algoritmos que possibilitam acelerar o processo da multiplicação e que servirão de base para a escolha do algoritmo deste trabalho.

CAPÍTULO 2

ALGORITMOS DE MULTIPLICAÇÃO BINÁRIA

2.1 Introdução.

Há diversos métodos de multiplicação binária que usam a técnica da adição repetitiva. Segundo **Y. CHU [5]**, os métodos mais conhecidos são os seguintes:

1. Método de Multiplicação Direta.
2. Método Burks - Goldstine - von Neumann.
3. Primeiro Método de Robertson.
4. Segundo Método de Robertson.
5. Método de Booth.

Tendo em vista que a multiplicação binária de dois números com sinais diferentes é influenciada pelo método usado para se representar os números negativos, a escolha da representação dos números negativos é muito importante.

2.2 O Método da Multiplicação Direta.

Este método é usado para números positivos, ou seja não trabalha com números em complemento de dois **[6]**. Seja o multiplicando X e o multiplicador Y representados por:

$$X = \sum_{i=0}^{n-1} x_i 2^i \qquad (2\text{-}1)$$

$$Y = \sum_{i=0}^{n-1} y_i 2^i \qquad (2\text{-}2)$$

onde:
- x_i e y_i podem ser, cada um, 0 ou 1;
- 2^i indica a posição do bit dentro do número.

O produto é representando pela letra P:

$$P = \sum_{i=0}^{n-1} X(y_i 2^i) = \sum_{i=0}^{n-1} (x_i 2^i) Y \qquad (2\text{-}3)$$

Já que os bits "y" do multiplicador podem assumir apenas os valores 0 e 1, a multiplicação binária torna-se em adições repetitivas com o multiplicando X adequadamente deslocado, como está mostrado na Fig. 2.1.

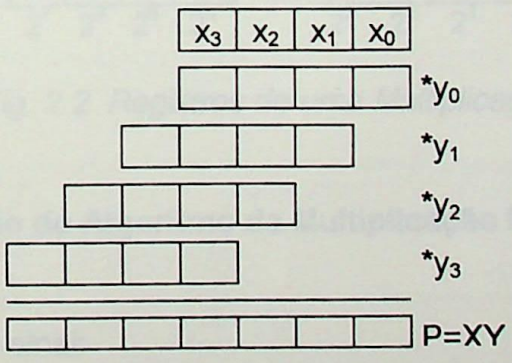


Fig. 2.1 Modelo de uma Multiplicação Direta de 4 bits.

A multiplicação no computador geralmente utiliza três registros **[5]**. Já que o produto tem duas vezes o número de bits do multiplicador (ou

multiplicando), o acumulador pode ser suplementado por um registro adicional para acumular o produto completo.

Como os registros são caros, o registro adicional pode ser substituído por um arranjo, ou seja, após a multiplicação por um bit do multiplicador Y, este bit não é mais útil ; ao mesmo tempo, o produto parcial é aumentado em um bit. Assim, o acumulador pode ser colocado junto ao registro do multiplicador para formar um só registro combinado, como mostra a Fig. 2.2 onde as potências de dois debaixo de cada quadrado indicam a posição de cada bit dentro do registro.

Depois da multiplicação de cada bit do multiplicador Y, o conteúdo de ambos registros, o acumulador e o registro do multiplicador, é deslocado de um bit para a direita. Dessa forma, o bit do multiplicador que já não é mais útil é deslocado para fora do registro e perdido. O Exemplo 2.1 ilustra este algoritmo.

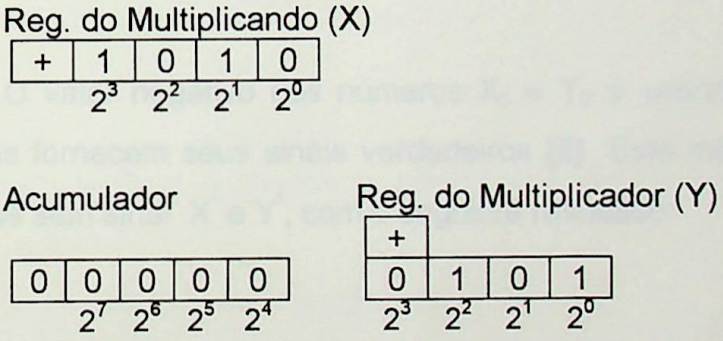


Fig. 2.2 Registros de uma Multiplicação.

Exemplo 2.1 Ilustração do Algoritmo da Multiplicação Direta

Sejam os números sem sinal:

X = multiplicando = 1010 (10_{DEC})

Y = multiplicador = 0101 (5_{DEC})

```
      1010  = X
x     0101  = Y
      1010    copia X
    0 000     soma 0
   10 10      copia X
  000 0       soma 0
 011 0010  = XY ( 50DEC )
```

2.3 Método Burks - Goldstine - von Neumann.

Este método multiplica números binários com sinal, representados em complemento de dois. Sendo o multiplicando X e o multiplicador Y, tem-se que:

$$X = -X_0 + \sum_{i=1}^{n-1} x_i 2^i = -X_0 + X^* \qquad (2\text{-}4)$$

$$Y = -Y_0 + \sum_{i=1}^{n-1} y_i 2^i = -Y_0 + Y^* \qquad (2\text{-}5)$$

O valor negativo dos números X_0 e Y_0 é usado porque desta maneira os números fornecem seus sinais verdadeiros **[5]**. Este método calcula o produto dos números sem sinal X^* e Y^*, com o seguinte resultado:

$$X^*Y^* = (X + X_0)(Y + Y_0) = XY + X_0Y + Y_0X + X_0Y_0 \qquad (2\text{-}6)$$

Caso	X_0	Y_0	*Produto*	*Correção*
a	0	0	$X^*Y^*=XY$	Nenhuma
b	0	1	$X^*Y^*=XY+Y_0X$	$-Y_0X$
c	1	0	$X^*Y^*=XY+X_0Y$	$-X_0Y$
d	1	1	$X^*Y^*=XY+X_0Y+Y_0X+X_0Y_0$	$-(X_0Y+Y_0X+X_0Y_0)$

Tab. 2.1 Algoritmo do Método Burks-Goldstine-von Neuman

O produto desejado é XY; portanto, uma correção tem que ser aplicada. O algoritmo deste método está mostrado na Tab. 2.1. O Exemplo 2.2 ilustra este algoritmo.

Exemplo 2.2. Ilustração do Algoritmo do Método Burks-Goldstine-von Neuman.

(a) ***Quando ambos números são positivos***, o caso é similar ao descrito no exemplo 2.1.

X = multiplicando = 0.1001 (9_{DEC}) $X_0 = 0$

Y = multiplicador = 0.1011 (11_{DEC}) $Y_0 = 0$

```
        .1001  = X*
  x     .1011  = Y*
        ------
        1001
       1 001
      00 00
     100 1
-------------------
0.0110 0011  = X* Y* = XY        (99 DEC)
```

(b) ***Quando o multiplicando X é positivo e o multiplicador Y é negativo***, tem-se:

X = multiplicando = 0.1001 (9_{DEC}) $X_0 = 0$

Y = multiplicador = 1.0101 (-11_{DEC}) $Y_0 = 1$

```
        .1001  = X*
  x     .0101  = Y*
        ------
        1001
       0 000
      10 01
     000 0
-------------------
 0. 0010 1101  = X* Y*
+ 1. 0111      = - Y0X Correção
-------------------
 1. 1001 1101  = XY        (-99 DEC)
```

O resultado da multiplicação é $X^{\cdot}Y^{\cdot}$, que logo sofre uma correção de $-Y_0X$ = -(1x0.1001) = -(0.1001) que é equivalente a se somar o complemento de dois de Y_0X (1.0111). Pode-se notar que o resultado XY é negativo, pois o bit de sinal é 1, e o número 99_{DEC} é calculado pelo complemento de dois de 1001 1101.

(c) ***Quando X é negativo e Y positivo***, tem-se:

O resultado da multiplicação é $X^{\cdot}Y^{\cdot}$ e requer uma correção que é subtrair X_0Y (0.1011). A subtração equivale a se somar o complemento de dois de X_0Y (1.0101).

Há um complicador neste caso pois o multiplicador Y é perdido como foi mencionado previamente e mostrado na Fig. 2.2 e, portanto, não está disponível para correção. A solução deste problema está em se considerar o número de correção 1.0101 como a soma dos números 1.0000, 0100 e 0001 que são obtidos da seguinte maneira:

- o primeiro número existe quando X_0 é 1 (entretanto, esta correção não é necessária);
- o segundo número .0100 é o complemento de um de $Y^{\cdot}$;
- o terceiro número .0001 é o valor somado ao complemento de um de $Y^{\cdot}$ para que se forme o complemento de dois.

O primeiro e terceiro números são somados e colocados no último passo como 1.0001. O segundo número é somado junto com os produtos parciais onde os dígitos de $Y^{\cdot}$ são zero.

X = multiplicando = 1.0111 (-9_{DEC}) $X_0 = 1$

Y = multiplicador = 0.1011 (11_{DEC}) $Y_0 = 0$

```
                    .0111  = X*
           x        .1011  = Y*
                    0111
                   0 111
Dígito de
correção  ------> 100 00
                011 1
             0. 1000 1101  = X* Y* com X0Y parcialmente corrigido
           + 1. 0001       = Correção que falta de - X0Y
             1. 1001 1101  = XY ( - 99 DEC )
```

(d) ***Quando X e Y são negativos***, tem-se:

X = multiplicando = 1.0111 (- 9$_{DEC}$) $X_0 = 1$

Y = multiplicador = 1.0101 (- 11$_{DEC}$) $Y_0 = 1$

```
                    .0111  = X*
           x        .0101  = Y*
                    0111
Dígitos de
correção  ------> 10 000
          \       01 11
           `---> 1000 0
             0.1100 0011   = X* Y* com X0Y parcialmente corrigido
           + 1.0001        = Correção que falta de - X0Y
             1.1101 0011   = XY + Y0X
           + 0.1001        = -Y0X  Correção devido ao sinal negativo de Y
          (1) 0.0110 0011  = XY  (99 DEC)
           ↑
      descartado
```

No caso "d", há dois zeros (y_4 e y_2) no multiplicador Y. Durante cada multiplicação pelos dígitos y_4 e y_2 , o digito 1 é adicionado ao produto parcial (dígitos de correção).

Na multiplicação do caso "d" é preciso se fazer algumas correções, ou seja, subtrair as quantidades X_0Y, Y_0X e X_0Y_0 . A subtração dos dois primeiros valores é feita de maneira similar aos casos 'b" e "c". Subtrair o valor X_0Y_0 (que possui o valor 1) é equivalente a adicionar 1, porque qualquer "carry" além do bit de sinal é descartado. Entretanto, a adição do número 1, como descrito

anteriormente, não é necessária. A correção de: *subtrair X_0Y* é o mesmo que: *adicionar o número 0.1011.*

Examinando-se o exemplo acima, vê-se que foi somado o valor 1.1011. O dígito mais significativo 1 é incorreto e poderia não ter sido somado, mas isto serve ao propósito de correção da quantidade X_0Y_0. *Consequentemente, nenhuma correção para X_0Y_0 é realmente necessária se o multiplicando e o multiplicador forem negativos.*

2.4 Primeiro Método de Robertson.

Este método precisa de uma correção simples quando o multiplicador Y é negativo mas quando o multiplicando X é negativo a correção é complexa. Este método é usado também para números representados em complemento de dois.

No método Burks-Goldstine-von Neumann, são multiplicados os números sem sinal X^* e Y^*. Já o primeiro método de Robertson multiplica o número com sinal X com o número sem sinal Y^* . O algoritmo deste método está mostrado na Tab. 2.2.

Caso	X_0	Y_0	Produto	Correção
a	0	0	$XY^*=XY$	Nenhuma
b	0	1	$XY^*=XY+Y_0X$	$-Y_0X$
c	1	0	$XY^*=XY$	Nenhuma
d	1	1	$XY^*=XY+Y_0X$	$-Y_0X$

Tab. 2.2 Algoritmo do Primeiro Método de Robertson

É interessante observar que Y^* é igual a Y quando Y_0=0. Quando o multiplicador Y é negativo (Y_0=1), a correção da subtração Y_0X é necessária. Nenhuma correção é necessária quando o multiplicando X é negativo. O Exemplo 2.3 ilustra este algoritmo.

Exemplo 2.3 ilustração do Algoritmo do Primeiro Método de Robertson

(a) ***Quando o multiplicador Y é positivo*** (sem correção),

X = multiplicando = 1.0111 (- 9_{DEC}) $X_0 = 1$

Y = multiplicador = 0.1011 (11_{DEC}) $Y_0 = 0$

```
          1.0111  = X
   x      .1011  = Y*
  ---------------
  1.1111 0111
  1.1110 111
  0.0000 00
  1.1011 1
 ---------------
(0) 1.1001 1101  = X Y* = XY ( -99 DEC )
 ^
 |
descartado
```

No exemplo acima os dígitos que estão sublinhados, são a extensão do sinal. Esta extensão de sinal é conseguida com o uso de um algoritmo de deslocamento. Como o multiplicador é positivo, nenhuma correção é necessária, como indica a tabela deste algoritmo.

(b) ***Quando o multiplicador Y é negativo***,

X = multiplicando = 1.0111 (- 9_{DEC}) $X_0 = 1$

Y = multiplicador = 1.0101 (-11_{DEC}) $Y_0 = 1$

```
          1.0111  = X
    x     .0101  = Y*
  ---------------
  1.1111 0111
  0.0000 000
  1.1101 11
  0.0000 0
  ---------------
(1) 1. 1101 0011   = X Y*
    0. 1001        = -Y0X  correção
  ---------------
(1) 0. 0110 0011   = XY ( 99 DEC )
  ↑
descartado
```

2.5 Segundo Método de Robertson.

No primeiro método de Robertson, a correção é feita quando o multiplicador Y é negativo, ou seja, quando Y_0=1. Para eliminar esta correção, Robertson propôs formar o produto de -X e -Y, onde Y ainda é negativo. Desta maneira, o número de sinal Y_0 de -Y torna-se 0.

Este método falha quando o multiplicador Y é igual ao número -1 (- 0001). O algoritmo deste método está dado na Tab.2.3. O Exemplo 2.4 ilustra este método.

Caso	X_0	Y_0	Produto	Correção
a	0	0	$XY= XY^*$	Nenhuma
b	0	1	(-X)(-Y)=XY	Nenhuma
c	1	0	$XY= XY^*$	Nenhuma
d	1	1	(-X)(-Y)=XY	Nenhuma

Tab. 2.3 Algoritmo do Segundo Método de Robertson

Exemplo 2.4 Ilustração do Algoritmo do Segundo Método de Robertson.

(a) ***Quando o multiplicando X é positivo***,

X = multiplicando = 0.1001 (+ 9_{DEC}) ⇒ -X=1.0111

Y = multiplicador = 1.0101 (-11_{DEC}) ⇒ -Y=0.1011

```
        1.0111  = -X
 x      0.1011  = -Y
-------------------
1.1111 0111
1.1110 111
0.0000 00
1.1011 1
-------------------
1.1001 1101   = XY ( - 99 DEC )
```

(b) ***Quando o multiplicando X é negativo***,

X = multiplicando = 1.0111 (-9_{DEC}) -X=0.1001

Y = multiplicador = 1.0101 (-11_{DEC}) -Y=0.1011

```
        0.1001  = -X
 x      0.1011  = -Y
-------------------
0.0000 1001
0.0001 001
0.0000 00
0.0100 1
-------------------
0.0110 0011   = XY ( 99 DEC )
```

Em ambos métodos de Robertson, precisa-se de circuitos adicionais para se implementar o algoritmo de extensão de sinal.

2.6 O Método de Booth.

Este método utiliza números em representação de complemento de dois e não precisa de nenhum fator de correção. Sejam o multiplicando X e o multiplicador Y, cujas representações são dadas pelas Eqs. (2-7) e (2-8) e onde os números X_0 e Y_0 são os números negativos ou nulos:

$$X = X_0 + \sum_{i=1}^{n-1} x_i 2^i \qquad (2\text{-}7)$$

$$Y = Y_0 + \sum_{i=1}^{n-1} y_i 2^i \qquad (2\text{-}8)$$

O produto XY é:

$$XY = Y_0 X + y_1 2^1 X + \ldots + y_{n-1} 2^{(n-1)} X + y_n 2^n X \qquad (2\text{-}9)$$

Substituindo-se cada bit y_i na Eq. (2-9) pela diferença $y_{i+1} - y_i$, que é uma subtração de dois dígitos adjacentes do multiplicador, tem-se:

y_n é substituído por y_{n+1}-y_n onde: y_{n+1}=0
y_{n-1} é substituído por y_n-y_{n-1}
.
.
. (2-10)
y_2 é substituído por y_3-y_2
y_1 é substituído por y_2-y_1
y_0 é substituído por y_1-y_0

Pode-se observar que quando a diferença de (y_{i+1} - y_i) é +1 ou -1, y_i e y_{i+1} são dois bits adjacentes do multiplicador Y que são, respetivamente, 01 e 10. Quando a diferença é 0, os bits y_i e y_{i+1}, são na verdade 00 ou 11.

Pode-se examinar dois bits adjacentes y_i y_{i+1} do multiplicador Y, e selecionar uma das três possíveis operações seguintes: adição, subtração ou nenhuma operação. Este processo é resumido como um algoritmo na Tab. 2.4.

Bits do Multiplicador $y_i\ y_{i+1}$	Operação
01	Soma o multiplicando X e desloca uma posição à direita o prod. parcial.
10	Subtrai o multiplicando X e desloca uma posição à direita o prod. parcial.
00	Desloca o produto parcial uma posição à direita.
11	Desloca o produto parcial uma posição à direita.

Tab. 2.4 Algoritmo de Booth para Multiplicação Binária

Após a substituição de cada y_i pela diferença $y_{i+1} - y_i$ na Eq. (2-9) tem-se:

$$XY = \left(-Y_0 + \sum_{i=1}^{n} y_i 2^i \right) X \qquad (2\text{-}11)$$

onde o valor entre parênteses acima representa o valor de Y. O Exemplo 2.5 ilustra este algoritmo.

Exemplo 2.5 Ilustração do Algoritmo de Booth.

(a) ***Quando o multiplicador é positivo***,

X = multiplicando = 1.0111 (-9_{DEC}) $\Rightarrow$(Complemento de 2= 01001)

Y = multiplicador = 0.1011 (11_{DEC})

1.0111 = X
0.1011(0) = Y
0.0000 0000 condição inicial no registro.

O bit **(0)** é o bit de referência com o qual se inicia o algoritmo. A operação de deslocamento à direita, devido ao último par de bits do multiplicador, não é necessária. Neste exemplo os dígitos "1" e "0" que estão sublinhados, são a extensão do sinal. A operação de subtrair X, significa o mesmo que somar o complemento de dois de X.

Então, tomando-se os bits do multiplicador Y (010110), de dois em dois e iniciando-se pelo bit menos significativo, tem-se:

		0.0000 0000
10	Subtrai X (soma o c. de 2 de X)	0.1001
		0.1001 0000
	Deslocamento à direita	0.0100 1000
11	Deslocamento à direita	0.0010 0100
01	Soma X	1.0111
		1.1001 0100
	Deslocamento à direita	1.1100 1010
10	Subtrai X	0.1001
		0.0101 1010
	Deslocamento à direita	0.0010 1101
01	Soma X	1.0111
		1.1001 1101=XY(-99_{DEC})

(b) ***Quando o multiplicador é negativo***,

X = multiplicando = 1.0111 (-9_{DEC})
Y = multiplicador = 1.0101 (-11_{DEC})

1.0111 = X
1.0101(0) = Y
0.0000 0000 condição inicial no registro.

		0.0000 0000
10	Subtrai X	0.1001
		0.1001 0000
	Deslocamento à direita	0.0100 1000
01	Soma X	1.0111
		1.1011 1000
	Deslocamento à direita	1.1101 1100
10	Subtrai X	0.1001
		0.0110 1100
	Deslocamento à direita	0.0011 0110
01	Soma X	1.0111
		1.1010 0110
	Deslocamento à direita	1.1101 0011
10	Subtrai X	0.1001
		0.0110 0011= XY(99_{DEC})

Então pode ser observado que o Algoritmo de Booth possui uma grande vantagem, não requer nenhum tipo de correção se comparado com o método de Burks-Goldstine-von Neumann e os métodos de Robertson.

2.7 Outros Algoritmos de Multiplicação.

Existem outros algoritmos que possibilitam acelerar o processo da multiplicação binária. Segundo **Fred Taylor [6]** , pode-se incluir ainda como importantes, o método do multiplicador com salvador de "carry" e o método da Árvore de Wallace.

2.7.1 Multiplicador com Salvador de "Carry".

O multiplicador com salvador de "carry" basicamente reduz o atraso gerado pelo "carry" que aparece durante uma multiplicação. Uma condição para melhorar a performance é configurar o último nível do multiplicador como um somador "look-ahead". O somador utilizado no multiplicador com salvador de "carry" usa um flip-flop para salvar a informação do "carry". Conforme mostrado na Fig. 2.3, este algoritmo realmente acelera a multiplicação. O caminho mais lento para o

"carry" é o caminho "***a***" e o mais rápido é o caminho ***"b"*** . A Fig. 2.4 ilustra este algoritmo.

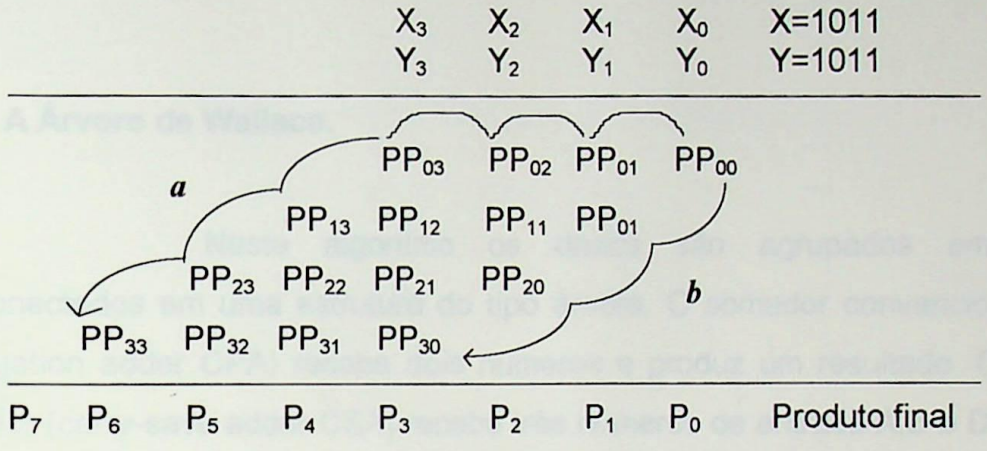


Fig. 2.3 Caminhos Possíveis para o "Carry" .

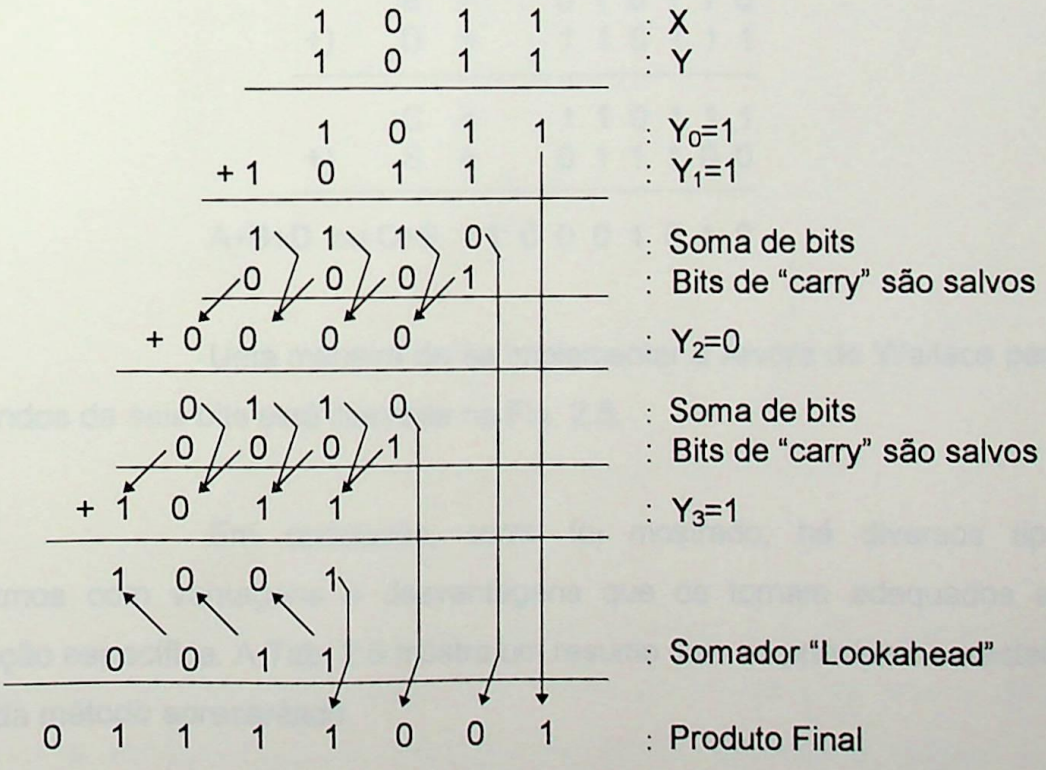


Fig. 2.4 Sequência de Operações de um Multiplicador Salvador de "carry".

Esta estrutura usa um dispositivo chamado pseudo-somador. Esta unidade soma três números juntos e disponibiliza como saída uma soma e um "carry". A vantagem desta estrutura é minimizar o caminho de propagação do "carry".

2.7.2 A Árvore de Wallace.

Neste algoritmo os dados são agrupados em três e interconectados em uma estrutura do tipo árvore. O somador convencional (carry propagation adder CPA) recebe dois números e produz um resultado. O pseudo-somador (carry-save adder CSA) recebe três números de entrada A,B e D, e possui duas saídas, o vetor soma S e o vetor "carry" C **[13]**, como está ilustrado no exemplo abaixo.

```
        A  =    1 1 1 1 0 1
        B  =    0 1 0 1 1 0
+)      D  =    1 1 0 1 1 1
-------------------------------
        C  =    1 1 0 1 1 1
+)      S  =    0 1 1 1 0 0
-------------------------------
A+B+D ou C+S = 1 0 0 0 1 0 1 0
```

Uma maneira de se implementar a Árvore de Wallace para dois operandos de seis bits está ilustrada na Fig. 2.5.

Em conclusão, como foi mostrado, há diversos tipos de algoritmos com vantagens e desvantagens que os tornam adequados a cada aplicação específica. A Tab. 2.5 mostra um resumo com as principais características de cada método apresentado.

Uma observação importante a ser feita é que: *o "Tempo de Latência" de um circuito, é o tempo contabilizado, desde que ingressa o primeiro bit em*

alguma porta de entrada do circuito, até que sai o último bit de alguma porta de saída do circuito **[14]**.

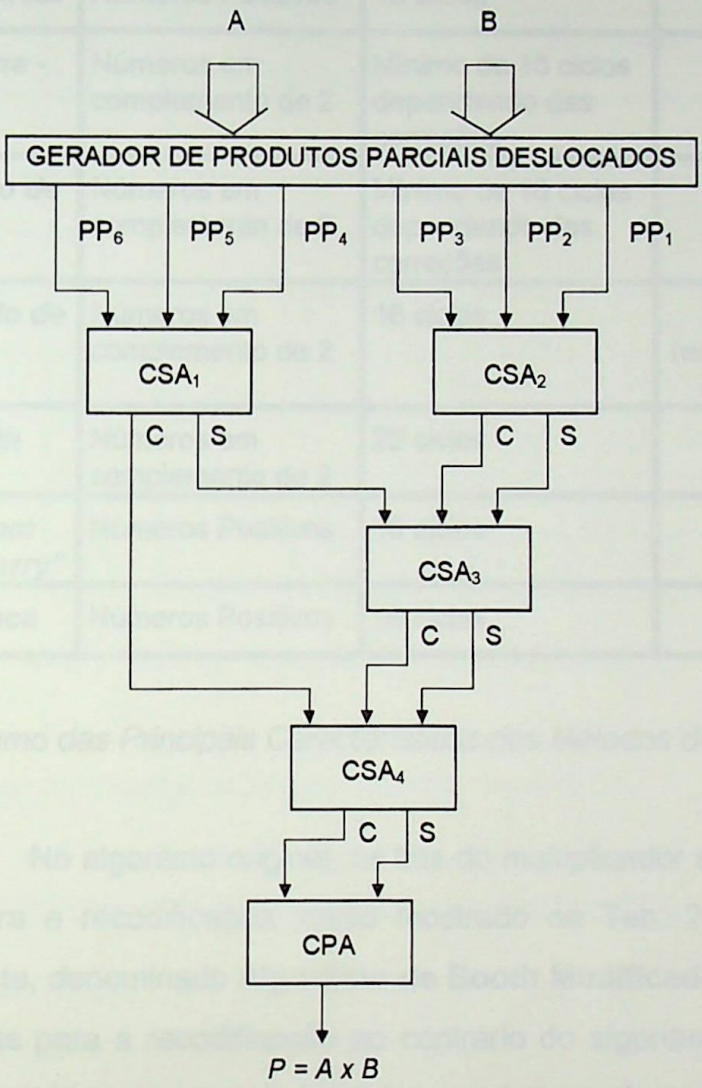


Fig.2.5 Multiplicador em Árvore de Wallace.

Como a proposta deste trabalho é a de desenvolver um multiplicador de alta performance, que ocupe pouca área de integração e que preserve a modularidade, o Algoritmo de Booth se adapta melhor a essas exigências pois é um algoritmo que não necessita nenhum tipo de correção durante a multiplicação, pode ser implementado em pequena área, desde que se faça uma boa escolha da célula de multiplicação, e também preserva a modularidade.

Método	Tipo de números com os quais trabalha	Tempo de Latência	Precisa de Correção
Multiplicação Direta	Números Positivos	16 ciclos	Não
Burks - Goldstine - von Neumann	Números em complemento de 2	Mínimo de 16 ciclos dependendo das correções	Sim
Primeiro Método de Robertson	Números em complemento de 2	Mínimo de 16 ciclos dependendo das correções	Sim
Segundo Método de Robertson	Números em complemento de 2	16 ciclos	Não (este método falha quando Y= -1)
Método de Booth	Números em complemento de 2	22 ciclos	Não
Multiplicador com Salvador de "Carry"	Números Positivos	16 ciclos	Não
Árvore de Wallace	Números Positivos	16 ciclos	Não

Tab. 2.5 Resumo das Principais Características dos Métodos de Multiplicação.

No algoritmo original, os bits do multiplicador são agrupados de dois em dois para a recodificação, como mostrado na Tab. 2.4. Um algoritmo levemente diferente, denominado **Algoritmo de Booth Modificado [8,9]**, considera grupos de três bits para a recodificação ao contrário do algoritmo original. E com isto, consegue um número menor de produtos parciais, acelerando o processo da multiplicação e reduzindo o hardware do projeto.

2.8 O Algoritmo de Booth Modificado.

Como ilustração do método, serão considerados dois operandos de oito bits, onde:

$X = X_7, X_6, \ldots, X_0$ = Multiplicando

$Y = Y_7, Y_6, \ldots, Y_0$ = Multiplicador

O esquema de recodificação para cada três bits do algoritmo de Booth pode ser encontrado em **[8]** da forma mostrada na Tab. 2.6.

Bit i+1	Bit i	Bit i-1	Dígito Recodificado	Operação em X
0	0	0	0	Some 0 ao produto parcial.
0	0	1	+1	Some X ao produto parcial.
0	1	0	+1	Some X ao produto parcial.
0	1	1	+2	Some 2∗X ao produto parcial.
1	0	0	-2	Subtraia 2∗X do produto parcial.
1	0	1	-1	Subtraia X do produto parcial.
1	1	0	-1	Subtraia X do produto parcial.
1	1	1	0	Some 0 ao produto parcial.

Tab. 2.6 Tabela de Recodificação do Algoritmo de Booth Modificado

O exemplo a seguir, mostrará como este algoritmo trabalha.

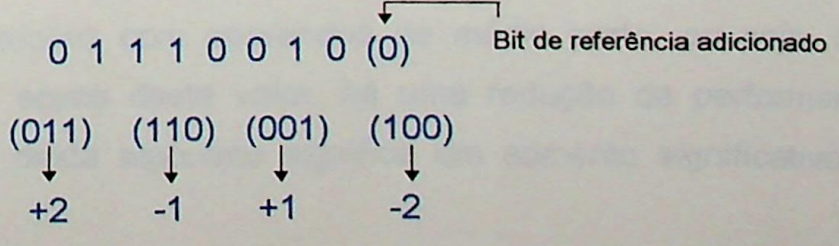
Exemplo 2.6 Ilustração do Algoritmo de Booth Modificado.

Sejam os números binários com sinal:

$$X = 1\,0\,1\,1\,0\,1\,0\,1\ (-75_{DEC})$$

$$Y = 0\,1\,1\,1\,0\,0\,1\,0\ (114_{DEC})$$

Recodificando-se o multiplicador Y, com base na Tab. 2.6, tem-se:



Então, o resultado final é de dezesseis bits, como pode-se observar na Tab. 2.7, e como é um número negativo (observe-se o bit de sinal igual a 1) deve-se tomar o complemento de dois para achar o verdadeiro valor dele, neste caso é equivalente a -8550_{DEC}.

Uma observação importante a ser feita é que a sequência do algoritmo, começa com o dígito recodificado dos bits menos significativos, e termina com o dígito recodificado dos bits mais significativos, então, a sequência deste exemplo será : -2, +1 , -1 e +2. Outra observação importante é que deve ser feita a extensão do sinal de cada produto parcial.

X = 1011 0101 x Y = 0111 0010		**Operação**
0000 0000 1001 0110	(-2)	*Soma o compl. de dois de X deslocado uma posição à esquerda.*
1111 1110 1101 01	(+1)	*Soma X.*
0000 0100 1011	(-1)	*Soma o compl. de dois de X.*
1101 1010 10	(+2)	*Soma X deslocado uma posição à esquerda.*
1101 1110 1001 1010		*Resultado Final* (-8550_{DEC})

1 ou 0 ⇒ Extensão de sinal

Tab. 2.7 Exemplo do Algoritmo de Booth Modificado

Sem dúvida, o algoritmo que recodifica grupos de três bits (Algoritmo de Booth Modificado) é muito mais vantajoso do que o que recodifica grupos de dois bits (Algoritmo de Booth Original). O Algoritmo de Booth Modificado é indicado para implementações com operandos de médio porte, ou seja, com operandos de até 32 bits, acima deste valor, há uma redução de performance. Também a implementação deste algoritmo significa um aumento significativo na

velocidade e uma redução no hardware e, consequentemente, na área de integração.

A seguir será feito um estudo da Célula Básica de Multiplicação utilizada neste projeto e também será descrita a implementação do Multiplicador Serial.

CAPÍTULO 3

A CÉLULA BÁSICA DE MULTIPLICAÇÃO

3.1 Implementação do Multiplicador Serial.

A implementação da Célula de Multiplicação Serial baseada no Algoritmo de Booth Modificado foi apresentada por **LYON [3]**. A estrutura "pipeline" modular apresentada foi proposta inicialmente por **E. K. CHENG** em julho de 1974 no Instituto Tecnológico da Califórnia em um trabalho não publicado. Esta célula ou módulo de multiplicação está mostrada na Fig. 3.1.

Os sinais A, B e C são sinais de gatilhamento recodificados, ou seja, eles fazem parte da recodificação do multiplicador Y em: 0, +1, +2, -1 e -2.

A implementação de um multiplicador serial do tipo "pipeline" de K bits, consiste de K/2 células se K for par, ou (K+1)/2 células se K for ímpar. Logo, se o multiplicador Y possui oito bits, tem-se quatro células de multiplicação. A Fig. 3.2 mostra a célula que foi escolhida para a implementação do multiplicador serial deste projeto **[10]** , com algumas modificações feitas pelo autor deste trabalho após uma criteriosa análise, e que será descrita mais adiante.

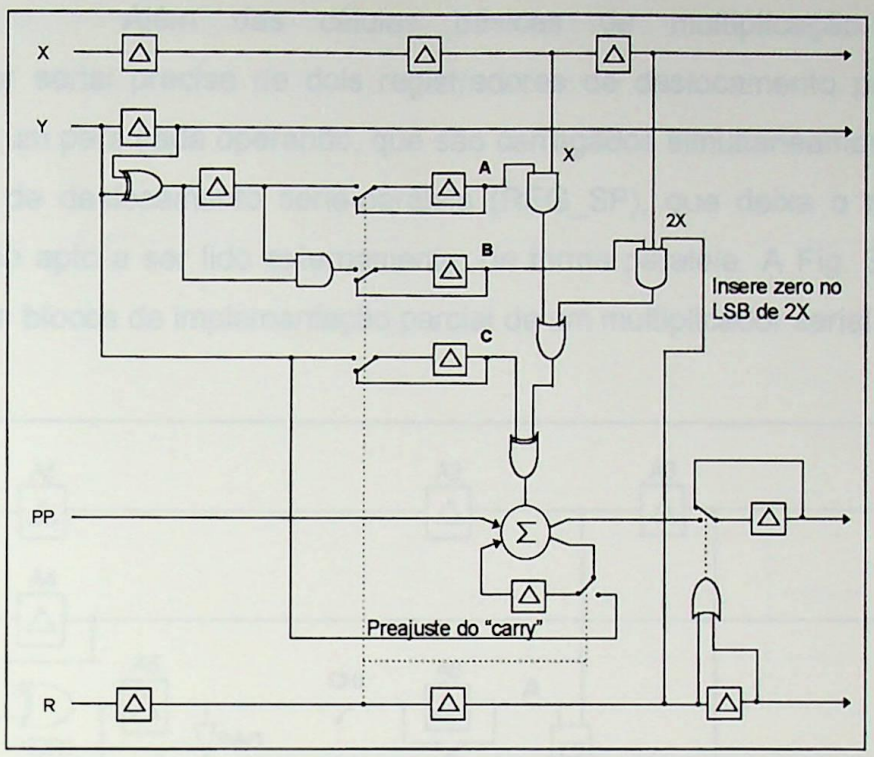


Fig. 3.1 Célula de Multiplicação de CHENG.

Os operandos X e Y da Fig. 3.2, são deslocados serialmente de forma sincronizada, para as entradas das células básicas de multiplicação (CBM), começando pelos bits menos significativos.

Inicialmente é realizada a recodificação do multiplicador Y. Os blocos lógicos XOR1, INV1, E1 e os atrasos A4 e A5, são os responsáveis pela recodificação de Y.

A recodificação do multiplicador $Y = Y_7\ Y_6\ Y_5\ Y_4\ Y_3\ Y_2\ Y_1\ Y_0\ (0)$ é feita da seguinte maneira:

1°. A primeira CBM recodifica os bits $Y_1\ Y_0\ (0)$.

2°. A segunda CBM recodifica os bits $Y_3\ Y_2\ Y_1$.

3°. A terceira CBM recodifica os bits $Y_5\ Y_4\ Y_3$.

4°. A quarta CBM recodifica os bits $Y_7\ Y_6\ Y_5$.

Além das células básicas de multiplicação (CBM), o multiplicador serial precisa de dois registradores de deslocamento paralelo-série (REG_PS), um para cada operando, que são carregados simultaneamente, e de um registrador de deslocamento série-paralelo (REG_SP), que deixa o resultado da multiplicação apto a ser lido externamente, de forma paralela. A Fig. 3.3 mostra o diagrama de blocos da implementação parcial de um multiplicador serial.

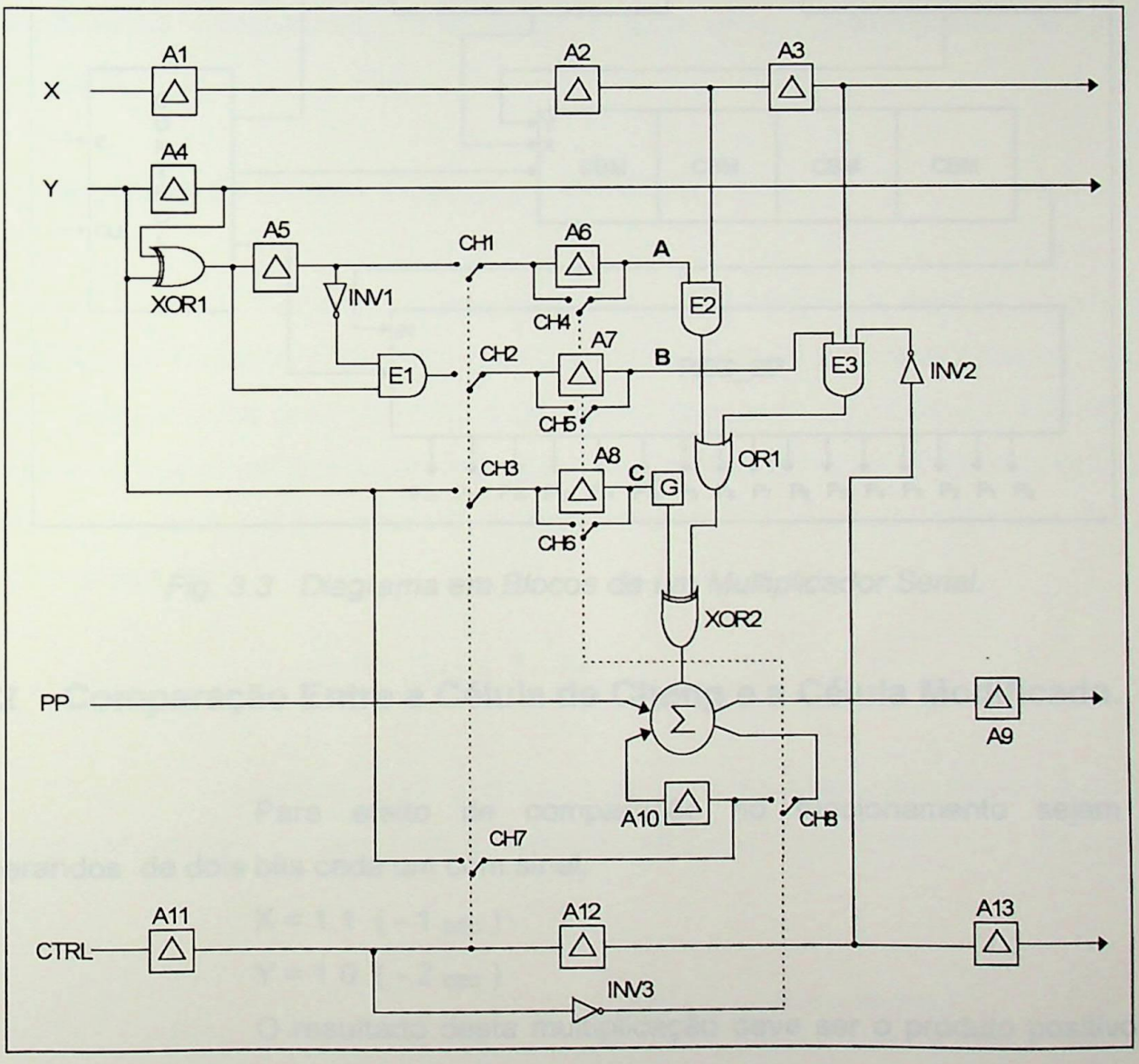


Fig. 3.2 Célula Básica de Multiplicação (CBM).

O multiplicador também precisa de uma Unidade de Controle, que recebe sinais externos de "clock" e um sinal de habilitação. Esta Unidade de

Controle gera sinais que gerenciam os registradores de deslocamento e as células básicas de multiplicação (CBM). Todos estes sinais serão analisados posteriormente.

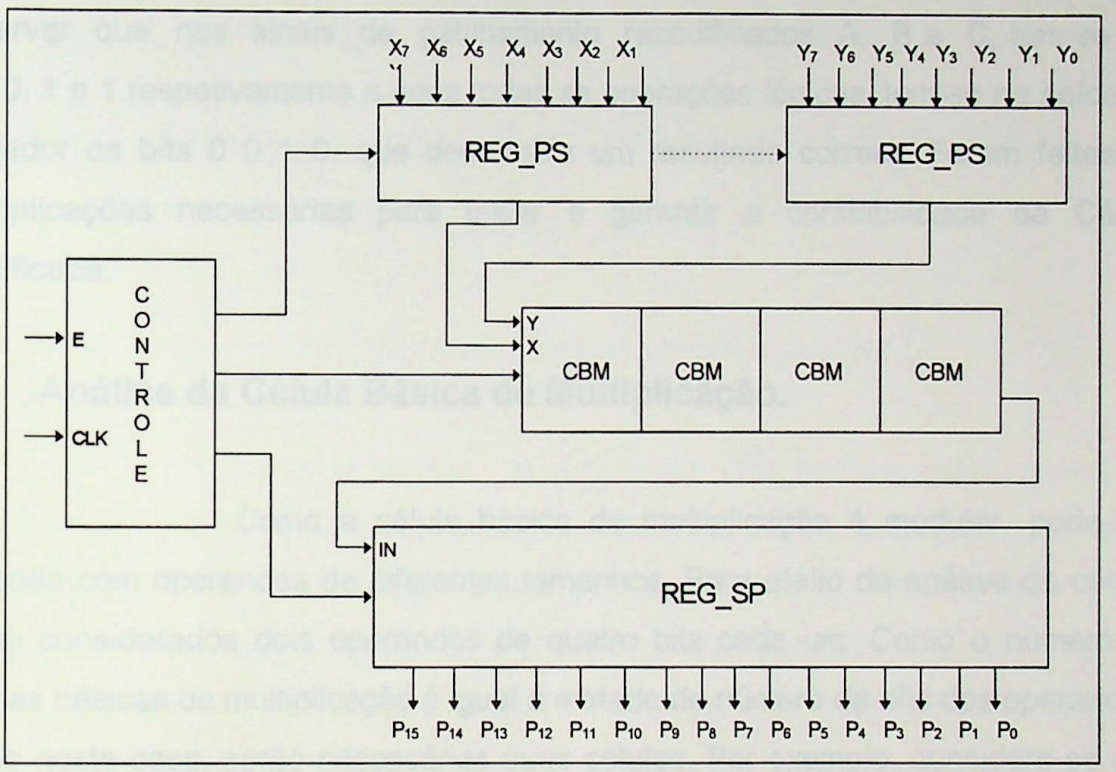


Fig. 3.3 Diagrama em Blocos de um Multiplicador Serial.

3.2 Comparação Entre a Célula de Cheng e a Célula Modificada.

Para efeito de comparação no funcionamento sejam os operandos de dois bits cada um com sinal:

$X = 1\ 1\ (-1_{DEC})$

$Y = 1\ 0\ (-2_{DEC})$

O resultado desta multiplicação deve ser o produto positivo de quatro bits: $P = 0\ 0\ 1\ 0\ (+2_{DEC})$.

Analisando primeiro a Célula de Cheng da Fig. 3.1, pode-se observar que nos sinais de gatilhamento recodificados A, B e C, tem-se os

bits 0, 0 e 1 respetivamente e após todas as operações lógicas, tem-se na saída do somador os bits 0 0 0 0, que demonstra um resultado errado.

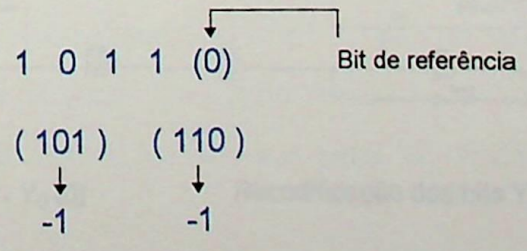
Analisando agora a Célula Modificada da Fig. 3.2, pode-se observar que nos sinais de gatilhamento recodificados A, B e C, tem-se os bits 0, 1 e 1 respetivamente e após todas as operações lógicas, tem-se na saída do somador os bits 0 0 1 0, que demonstra um resultado correto. Foram feitas as multiplicações necessárias para testar e garantir a confiabilidade da Célula Modificada.

3.3 Análise da Célula Básica de Multiplicação.

Como a célula básica de multiplicação é modular, pode ser utilizada com operandos de diferentes tamanhos. Para efeito de análise da célula, serão considerados dois operandos de quatro bits cada um. Como o número de células básicas de multiplicação é igual à metade do número de bits dos operandos, então neste caso, serão necessárias duas células. Por exemplo, considere-se que os operandos são:

$$X = X_3\ X_2\ X_1\ X_0$$
$$Y = Y_3\ Y_2\ Y_1\ Y_0 = 1\ 0\ 1\ 1$$

Inicialmente, a célula básica de multiplicação receberá os bits menos significativos do operando Y. Recodificando-se Y segundo a Tab. 3.1 tem-se que:



Como o dígito recodificado é -1, então deve-se somar o complemento de dois do operando X. O resultado esperado desta multiplicação

está mostrado na Fig. 3.4, onde a letra "c" representa o "carry" que deve ser adicionado.

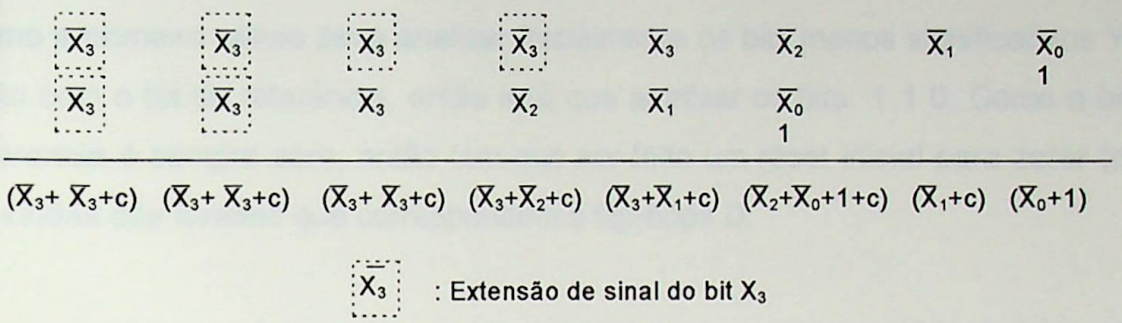
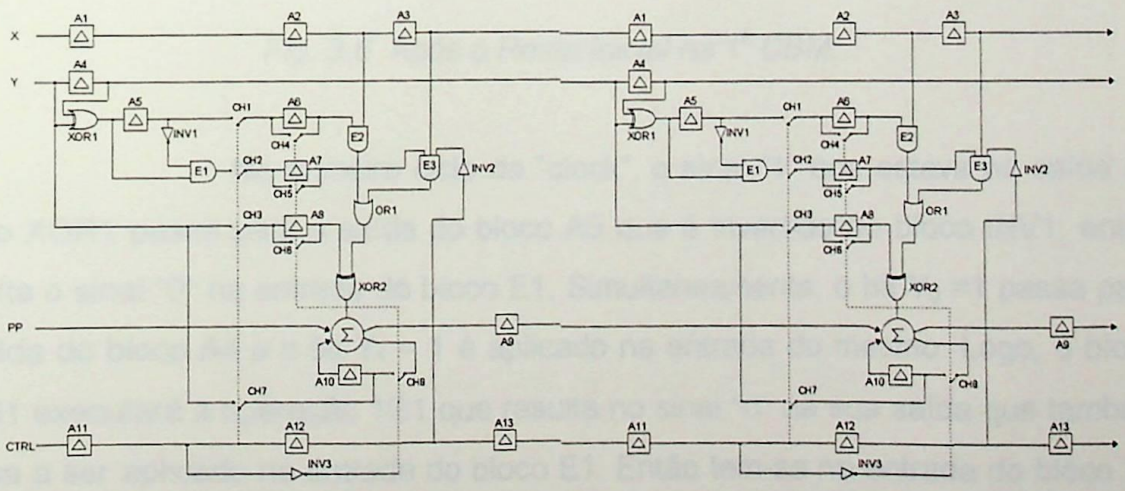


Fig. 3.4 Resultado Esperado da Multiplicação

Como já foi dito, o grupo de bits menos significativos do multiplicador Y (Y_1 Y_0 0) é recodificado na primeira CBM e o grupo de bits mais significativos (Y_3 Y_2 Y_1) é recodificado na segunda CBM. O arranjo das células básicas de multiplicação para este exemplo está ilustrado na Fig. 3.5. Na sequência, será feita uma análise do que ocorre nos blocos lógicos dentro da célula básica de multiplicação.



Recodificação dos bits Y_1 Y_0 (0) Recodificação dos bits Y_3 Y_2 Y_1

Fig. 3.5 Arranjo de duas CBM.

Blocos A4, XOR1, A5, INV1 e E1

Como já comentado, estes blocos recodificam o multiplicador Y. Como a primeira célula deve analisar inicialmente os bits menos significativos Y_1 Y_0 junto com o bit de referência, então terá que analisar os bits 1 1 0. Como o bit de referencia é sempre zero, então tem que ser feito um reset inicial para zerar todas as saídas dos atrasos que correspondem a flip-flops D.

Logo após um reset inicial, tem-se zero na saída do bloco A4 e o bit Y_0 = 1 na entrada do mesmo. Estes dois sinais passam pelo bloco XOR1 obtendo-se na sua saída o sinal "1" (0⊕1 = 1), como ilustra a Fig. 3.6.

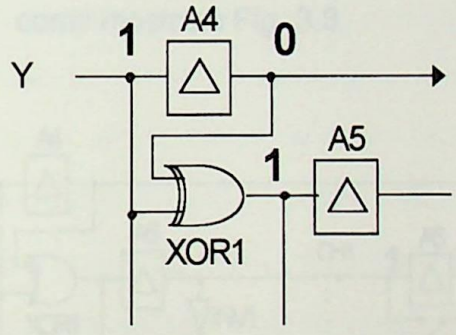


Fig. 3.6 Após o Reset Inicial na 1ª CBM.

No primeiro ciclo de "clock", o sinal "1" que estava na saída do bloco XOR1 passa para a saída do bloco A5 que é invertido no bloco INV1, então resulta o sinal "0" na entrada do bloco E1. Simultaneamente, o bit Y_0 =1 passa para a saída do bloco A4 e o bit Y_1 = 1 é aplicado na entrada do mesmo. Logo, o bloco XOR1 executará a operação 1⊕1 que resulta no sinal "0" na sua saída que também passa a ser aplicado na entrada do bloco E1. Então tem-se na entrada do bloco E1 os sinais "0" e "0", na saída do mesmo ter-se -á o sinal "0". Estes passos estão ilustrados na Fig. 3.7.

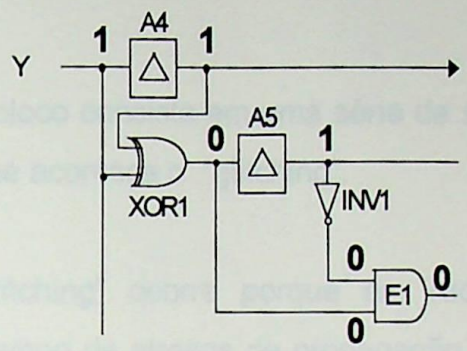


Fig. 3.7 Primeiro Ciclo de "clock" na 1ª CBM.

Como as chaves CH1, CH2 e CH3 estão fechadas no primeiro ciclo de "clock", e as chaves CH4, CH5 e CH6 estão abertas, devido ao sinal de controle CTRL, então os sinais "1", "0", e "1" passam respetivamente às entradas dos atrasos A6, A7 e A8, como mostra a Fig. 3.8.

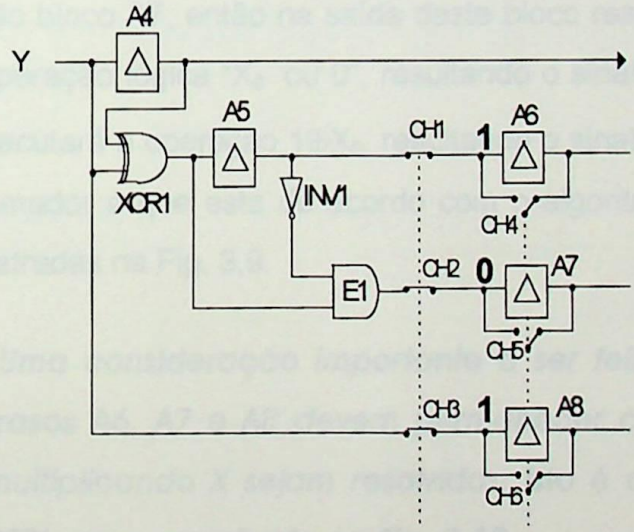


Fig. 3.8 Primeiro Ciclo de "clock" na 1ª CBM.

Após o segundo ciclo de "clock", os sinais que estavam nas entradas dos atrasos A6, A7 e A8, passam para suas saídas (ver Fig. 3.9). Observar que neste ciclo de "clock", tem-se nas saídas dos blocos A1, A2 e A3 os sinais X_1, X_0 e 0 respetivamente ("0" devido ao reset inicial).

Bloco G

Este bloco consiste em uma série de seis inversores colocados em cascata para evitar que aconteça o "glitching".

O "glitching" ocorre porque em redes de circuitos lógicos multiestágios existe um tempo de atrasos de propagação diferentes para os vários caminhos existentes do circuito. Para minimizar o problema de "glitching", os atrasos de propagação devem ser iguais **[15]**.

Blocos E2, E3, OR1 e XOR2

Como na entrada do bloco E2 tem-se os sinais "1" e X_0, então na sua saída resultará o sinal X_0. Como na entrada do bloco E3 tem-se o sinal "0", que vem da saída do bloco A7, então na saída deste bloco resultará o sinal "0". O bloco OR1 fará a operação lógica "X_0 ou 0", resultando o sinal X_0 na entrada do bloco XOR2 que executará a operação $1 \oplus X_0$, resultando o sinal $\overline{X}_0$. Este é o valor que ingressa no somador e que esta de acordo com o algoritmo de Booth. Estas operações estão ilustradas na Fig. 3.9.

Uma consideração importante a ser feita é que os sinais nas saídas dos atrasos A6, A7 e A8 devem permanecer constantes até que todos os bits do multiplicando X sejam resolvidos. Isto é conseguido com o sinal de controle CTRL, como mostrado na Fig. 3.10.

Blocos A10, CH7 e CH8

Os blocos A10 e CH7, tem a função de adicionar ou não um sinal "1" ao bit menos significativo (LSB), para assim obter o complemento de dois de X, o LSB neste caso é o bit X_0. Os blocos A10 e CH8, tem a função de adicionar um "carry" se for necessário.

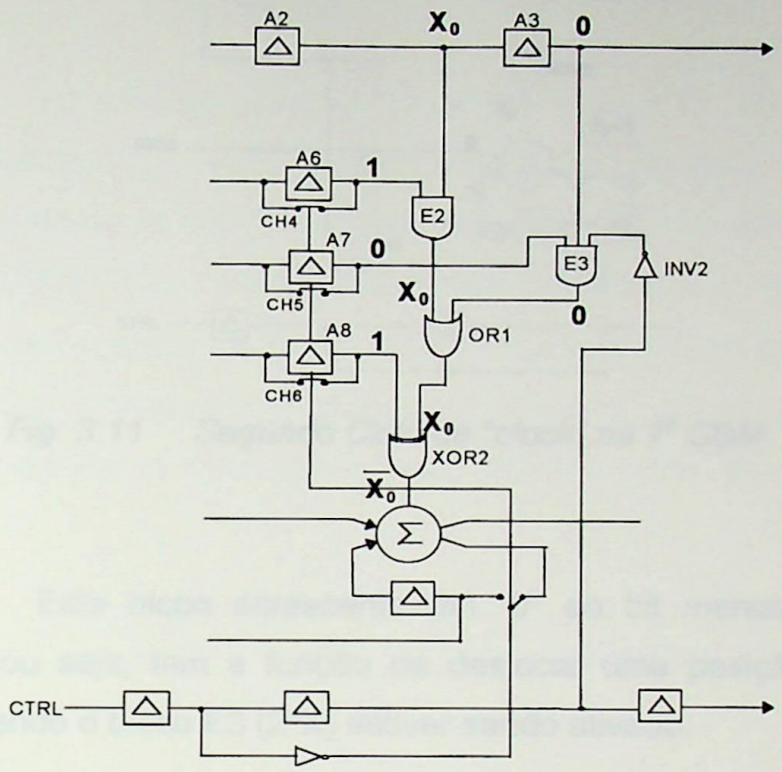


Fig. 3.9 Segundo Ciclo de "clock" na 1ª CBM.

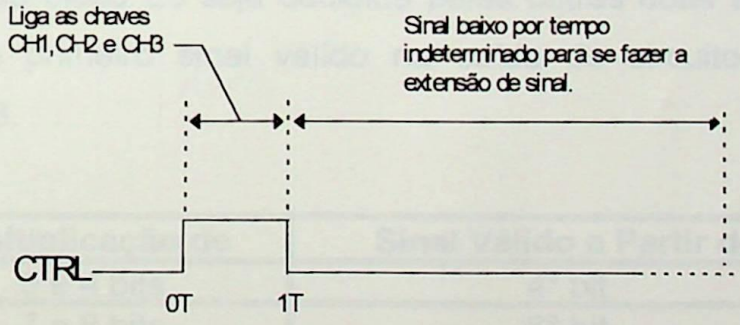


Fig. 3.10 Temporização do Sinal de Controle CTRL .

Analisando este exemplo, tem-se que o bit Y_1=1 é levado para a saída do bloco A10 através da chave CH7 no segundo ciclo de "clock". Assim o somador executará $0+\overline{X}_0+1 = \overline{X}_0+1$, o que está de acordo com o esperado. Isto esta ilustrado na Fig. 3.11.

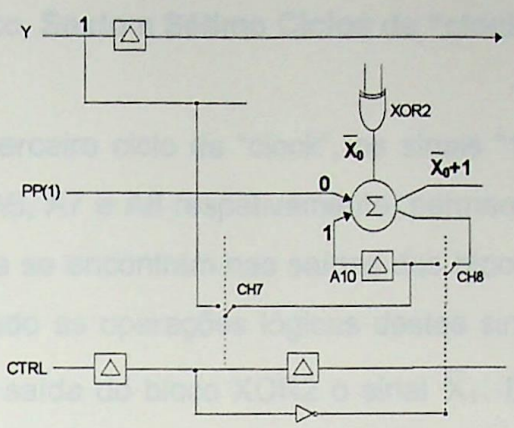


Fig. 3.11 Segundo Ciclo de "clock" na 1ª CBM.

Bloco INV2

Este bloco acrescenta um "0" ao bit menos significativo do multiplicando X, ou seja, tem a função de deslocar uma posição à esquerda o multiplicando, quando o bloco E3 (2*X) estiver sendo ativado.

Na sequência, o bloco XOR2 decidirá se este bit continua sendo "0" (que corresponde a +2), ou se será "1" (que corresponde a -2). A partir do terceiro ciclo de "clock", o sinal na saída do bloco INV2 será sempre "1", o que faz com que a saída do bloco E3 seja decidida pelas outras duas entradas. Deve ser observado que, o primeiro sinal válido na saída do circuito do multiplicador, obedece à Tab. 3.3.

Multiplicação de	Sinal Válido a Partir de
3 e 4 bits	4° bit
7 e 8 bits	6° bit
11 e 12 bits	8° bit
15 e 16 bits	10° bit

Tab. 3.3 Validade do Sinal de Saída.

Blocos A11, A12 e A13

A função destes blocos é a de acrescentar atrasos no sinal de controle CTRL aplicado na entrada do bloco A11 e gerado pelo Módulo de Controle.

Terceiro, Quarto, Quinto, Sexto e Sétimo Ciclos de "clock"

No terceiro ciclo de "clock", os sinais "1" , "0" e "1" que estão nas saídas dos blocos A6, A7 e A8 respetivamente, permanecem constantes e, por outro lado, os sinais que se encontram nas saídas dos blocos A2 e A3 são X_1 e X_0 respetivamente. Efetuando as operações lógicas destes sinais nos blocos E2, E3, INV2 e OR1 tem-se na saída do bloco XOR2 o sinal $\overline{X}_1$. Este sinal é somado ao valor ("carry") que estiver nesse momento na saída do bloco A10, o valor resultante passa para a saída do somador da primeira CBM. Isto está ilustrado na Fig. 3.12.

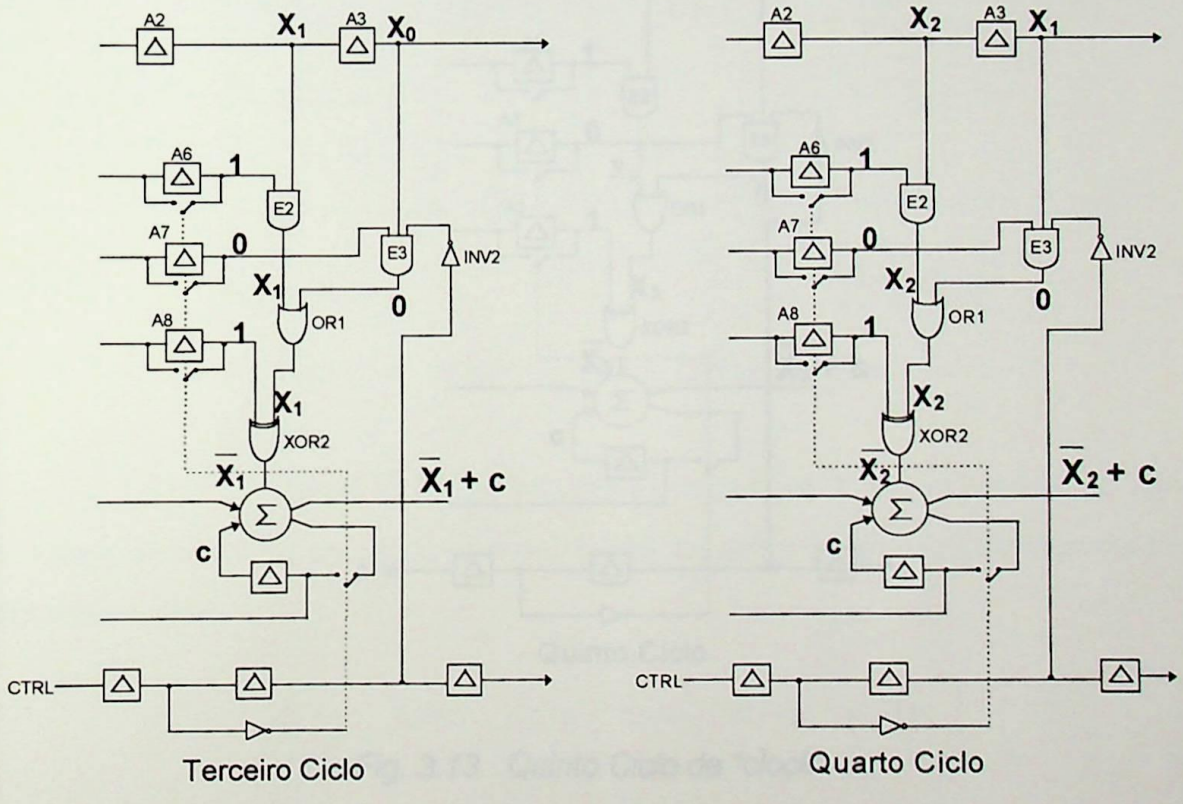


Fig. 3.12 Terceiro e Quarto Ciclos de "clock".

No quarto ciclo de "clock", os sinais nas saídas dos blocos A6, A7 e A8 ainda permanecem constantes e nas saídas dos blocos A2 e A3 tem-se os sinais X_2 e X_1 respetivamente. Efetuando as operações lógicas correspondentes, tem-se na saída do bloco XOR2 o sinal $\overline{X}_2$, que é aquele que se terá na saída do

bloco somador mais o sinal ("carry") da saída do bloco A10. Isto está ilustrado na Fig. 3.12.

No quinto ciclo de "clock", tem-se nas saídas dos blocos A2 e A3 os sinais X_3 e X_2 respetivamente e as saídas dos blocos A6, A7 e A8 permanecem constantes. Logo, depois de efetuadas as operações lógicas correspondentes, tem-se na saída do bloco somador o sinal $\overline{X}_3$ mais o sinal ("carry") da saída do bloco A10. Isto está ilustrado na Fig. 3.13.

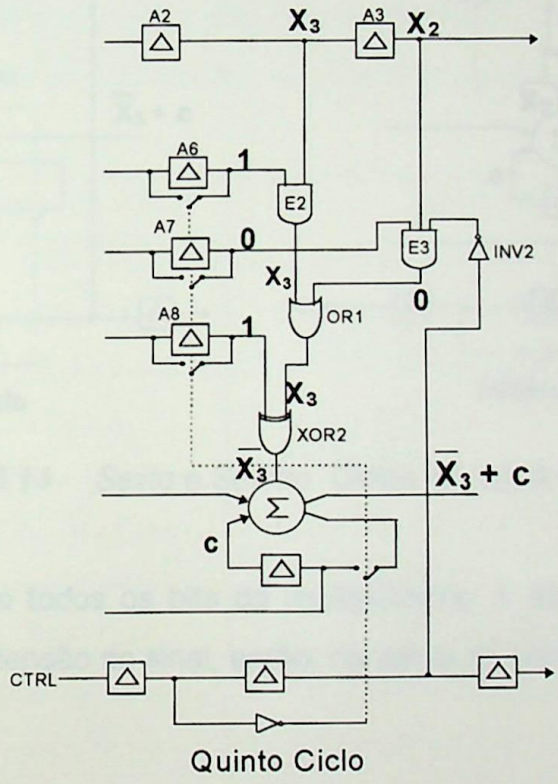


Fig. 3.13 Quinto Ciclo de "clock".

No sexto e sétimo ciclos de "clock" é efetuada a extensão de sinal do produto parcial. Tem-se nas saídas dos blocos A2 e A3 os sinais X_3 e X_3 (este bit X_3 é extendido com ajuda do registrador paralelo-série como será apresentado mais adiante). Na saída do bloco XOR2 tem-se então $\overline{X}_3$, para os dois

ciclos, e na saída do bloco somador será agregado o "carry" se houver. Isto está ilustrado na Fig. 3.14.

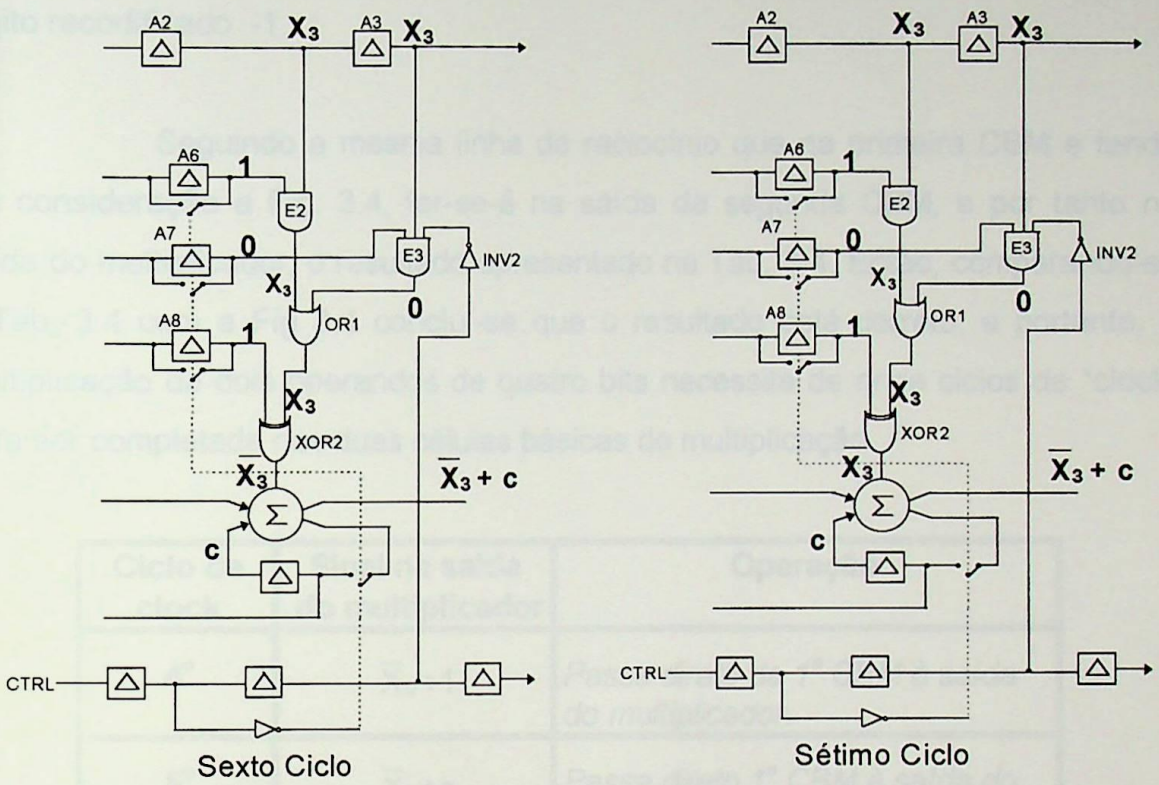


Fig. 3.14 Sexto e Sétimo Ciclos de "clock".

Como todos os bits do multiplicando X foram resolvidos junto com mais dois bits de extensão de sinal, então, na saída da primeira CBM tem-se os seguintes sinais:

$$\overline{X}_3+c \quad \overline{X}_3+c \quad \overline{X}_3+c \quad \overline{X}_2+c \quad \overline{X}_1+c \quad \overline{X}_0+1$$
$$(7^o) \quad (6^o) \quad (5^o) \quad (4^o) \quad (3^o) \quad (2^o)$$

que estão corretos como era esperado já que, a recodificação do grupo dos três bits menos significativos do operando Y (1 1 0) resultou no dígito recodificado -1 que equivale a dizer "somar o complemento de dois de X".

Continuando com o exemplo em questão, na segunda CBM é recodificado o grupo de três bits mais significativos, e no quinto ciclo de "clock" tem-

se gatilhado as saídas dos blocos A6, A7 e A8 com os sinais "1" , "0" e "1" respetivamente, que são os mesmos sinais que gatilhavam a primeira CBM, o que era de se esperar porque a recodificação deste grupo de bits também resultou no dígito recodificado -1.

Seguindo a mesma linha de raciocínio que na primeira CBM e tendo em consideração a Fig. 3.4, ter-se-á na saída da segunda CBM, e por tanto na saída do multiplicador, o resultado apresentado na Tab. 3.4. Então, comparando-se a Tab. 3.4 com a Fig 3.4 conclui-se que o resultado está correto, e portanto, a multiplicação de dois operandos de quatro bits necessita de onze ciclos de "clock" para ser completada nas duas células básicas de multiplicação.

Ciclo de clock	Sinal na saída do multiplicador	Operação
4°	$\overline{X}_0+1$	*Passa direto da 1ª CBM à saída do multiplicador.*
5°	$\overline{X}_1+c$	*Passa direto 1ª CBM à saída do multiplicador.*
6°	$\overline{X}_2+\overline{X}_0+1+c$	*O somador da 2ª CBM realiza esta operação.*
7°	$\overline{X}_3+\overline{X}_1+c$	*O somador da 2ª CBM realiza esta operação.*
8°	$\overline{X}_3+\overline{X}_2+c$	$\overline{X}_3$ *é bit de extensão e é somado a* $\overline{X}_2$.
9°	$\overline{X}_3+\overline{X}_3+c$	$\overline{X}_3$ *é bit de extensão e é somado a* $\overline{X}_3$.
10°	$\overline{X}_3+\overline{X}_3+c$	*CTRL mantém o sinal do ciclo anterior na 2ª CBM (ext. de sinal).*
11°	$\overline{X}_3+\overline{X}_3+c$	*CTRL mantém o sinal do ciclo anterior na 2ª CBM (ext. de sinal).*

Tab. 3.4 Resultado Final da Multiplicação.

Uma vez apresentada a célula básica de multiplicação que será utilizada no multiplicador serial, o capítulo seguinte tratará sobre a implementação desta célula e dos demais blocos necessários no multiplicador serial.

CAPÍTULO 4

IMPLEMENTAÇÃO DO MULTIPLICADOR, "LAYOUT" E TESTES

4.1 Introdução.

Este capítulo trata da implementação dos circuitos usados nos diferentes blocos do multiplicador serial. Todos estes blocos estão descritos em detalhes, também é feito um estudo da temporização para saber com qual frequência máxima este multiplicador poderá trabalhar corretamente. Finalmente é apresentado o layout do multiplicador e os testes feitos com o protótipo.

4.2 Temporização dos Flip - Flops.

Para se conseguir a frequência máxima de "clock" com que funcionará o multiplicador, deve ser feita uma análise dos tempos de propagação do mestre e do escravo dos flip-flops utilizados neste projeto e também do caminho crítico.

O flip-flop JK é utilizado para se implementar os blocos registradores de deslocamento paralelo-série e série-paralelo, como mostram as Figs. 4.16 e 4.17. O flip-flop D é utilizado como um elemento de atraso na célula básica de multiplicação (CBM). O caminho crítico encontrado na CBM, após uma análise rigorosa, está mostrado na Fig. 4.1.

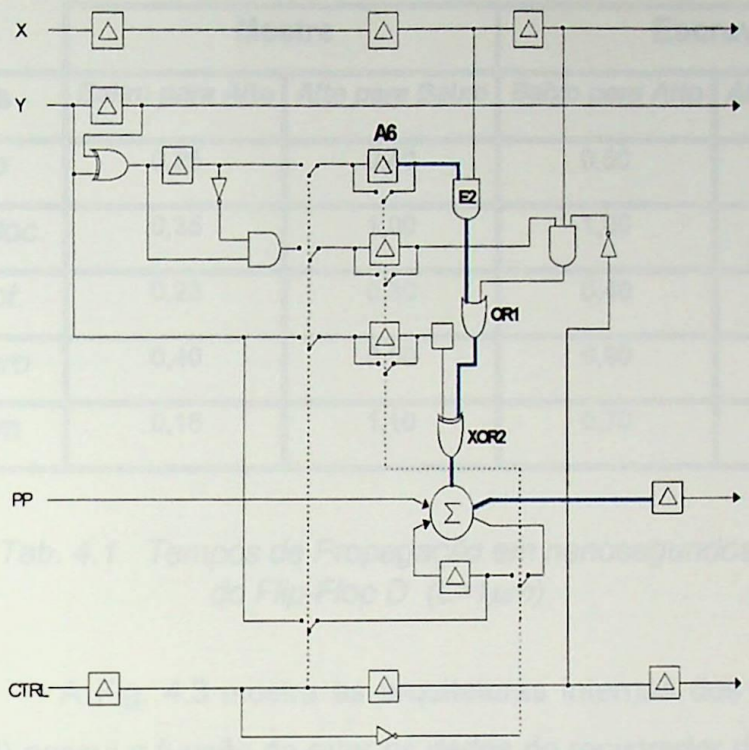
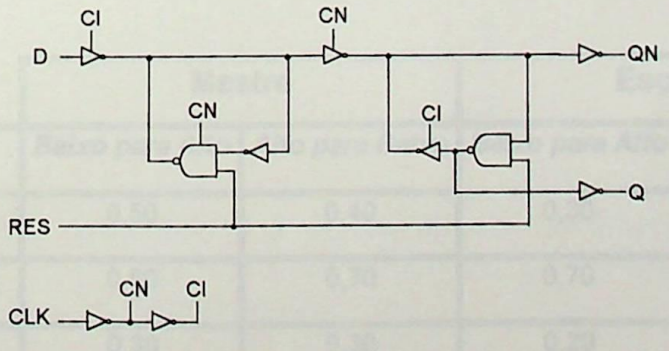


Fig. 4.1 Caminho Crítico.

A Fig. 4.2 mostra o circuito do flip-flop D e a Tab. 4.1 mostra sua temporização obtida com o simulador elétrico Tspice versão 3.0 da Tanner Research.



RES	D	CLK	Q
0	X	X	0
1	1	↓	1
1	0	↓	0
1	X	↑	Não Muda

Fig. 4.2 Flip-Flop D com Reset e sua Tabela de Verdade.

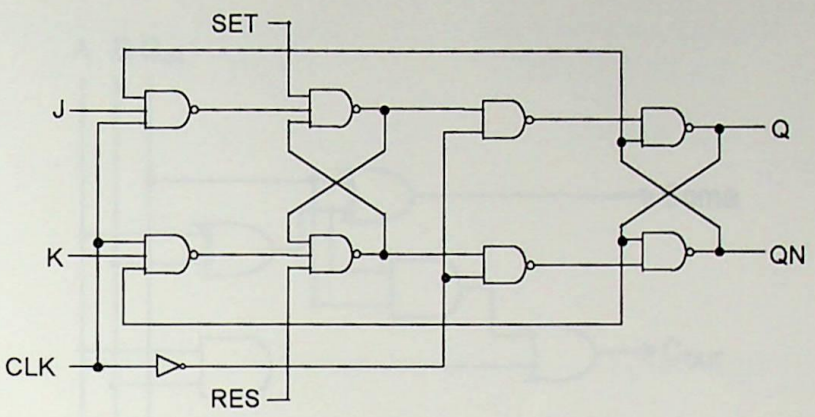
	Mestre		Escravo	
Casos	***Baixo para Alto***	***Alto para Baixo***	***Baixo para Alto***	***Alto para Baixo***
Típico	0,25	0,40	0,50	0,70
Pior Veloc.	0,35	1,00	1,00	1,40
Pior Pot.	0,23	0,30	0,40	0,40
Pior Zero	0,40	0,50	0,90	0,70
Pior Um	0,18	1,10	0,70	1,00

Tab. 4.1 Tempos de Propagação em nanosegundos do Flip-Flop D (L=1μm)

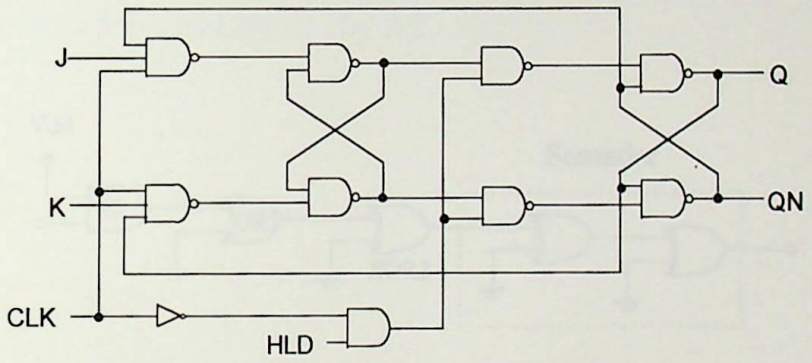
A Fig. 4.3 mostra as arquiteturas internas dos flip-flops JK. O sinal HLD (Hold) possui a função de reter os dados do registrador de deslocamento série-paralelo com um sinal "0" quando a multiplicação estiver completa, assim, os dados armazenados neste registrador estão disponíveis a qualquer instante e não serão perdidos. A Tab. 4.2 mostra a temporização do flip-flop JK. Pode ser observado que o maior tempo de propagação dos flip-flops JK e D se encontra no flip-flop D, portanto, utilizando-se este tempo para o cálculo da frequência máxima, se garante que o sinal viajará pelos flip-flops sem nenhum problema.

	Mestre		Escravo	
Casos	***Baixo para Alto***	***Alto para Baixo***	***Baixo para Alto***	***Alto para Baixo***
Típico	0,50	0,40	0,30	0,30
Pior Veloc.	0,80	0,70	0,70	0,50
Pior Pot.	0,30	0,30	0,20	0,20
Pior Zero	0,70	0,50	0,50	0,40
Pior Um	0,60	0,40	0,30	0,30

Tab. 4.2 Tempos de Propagação em nanosegundos do Flip-Flop JK (L=1μm).



(a)



(b)

J	K	CLK	SET	RES	Q_{n+1}
X	X	0	0	0	Indet.
X	X	0	0	1	1
X	X	0	1	0	0
X	X	0	1	1	Q_n
0	0	↓	1	1	Q_n
0	1	↓	1	1	0
1	0	↓	1	1	1
1	1	↓	1	1	QN_n

(c)

Fig. 4.3 (a) Flip-Flop JK do Registrador Paralelo-Série.
(b) Flip-Flop JK do Registrador Série-Paralelo.
(c) Tabela de Verdade.

A arquitetura do Somador Completo utilizado na célula básica de multiplicação e que faz parte do caminho crítico, está mostrada na Fig. 4.4.

A Fig. 4.5 mostra o caminho crítico da CBM, utilizado para a obtenção da Tab. 4.3 que mostra sua temporização.

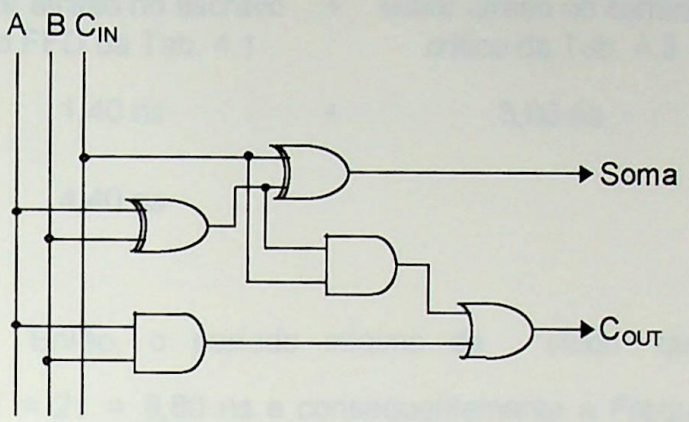


Fig. 4.4 Somador Completo

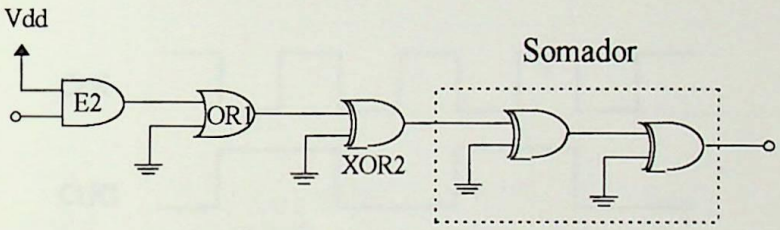


Fig. 4.5 Caminho Crítico.

Casos	*Baixo para Alto*	*Alto para Baixo*
Típico	1,60	1,80
Pior Veloc.	1,75	3,00
Pior Pot.	1,00	1,00
Pior Zero	2,00	2,30
Pior Um	2,00	2,30

Tab. 4.3 Tempos de Propagação em nanosegundos do Caminho Crítico (L=1μm).

Logo, para que um sinal viaje pelo escravo do atraso A6 mais o caminho crítico (ver Fig. 4.1), o "clock" deve permanecer em "0" (sinal baixo) durante o tempo τ , onde:

τ = Maior atraso no escravo do FFD da Tab. 4.1 + Maior atraso no caminho crítico da Tab. 4.3

τ = 1,40 ns + 3,00 ns

τ = 4,40 ns

Então, o período mínimo de "clock" que ingressará no multiplicador é $T = 2\tau = 8,80$ ns e consequentemente a Frequência Máxima de trabalho será de 125 MHz. Portanto as formas de onda necessárias de "clock" para este projeto estão apresentadas na Fig. 4.6.

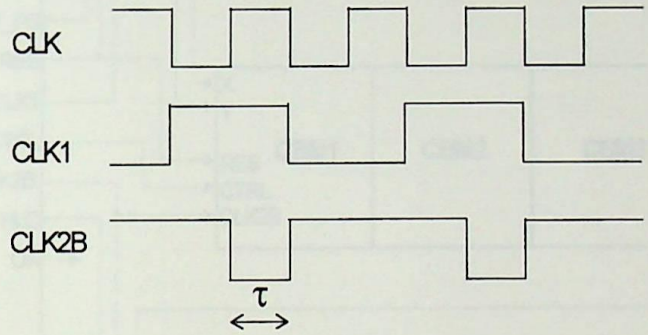


Fig. 4.6 Formas de Onda dos "Clocks".

Observe-se que o sinal CLK na Fig. 4.6 é o sinal de "clock" de entrada, que possui um período de 8,80ns (2τ). O sinal CLK1 é o "clock" que gerenciará os flip-flops JK dos dois registradores paralelo-série; e o sinal CLK2B é o "clock" que gerenciará os flip-flops D da CBM e os flip-flop JK do registrador série-paralelo. O sinal CLK2B deve ser o oposto do sinal CLK1 e não possui um ciclo de trabalho de 50%. Este tipo de distribuição dos "clocks" foi feita para garantir que não aconteçam erros durante a leitura dos dados seriais que saem dos registradores paralelo-série e ingressam na CBM.

4.3 O Circuito Completo.

O circuito completo em blocos do multiplicador serial de 8 bits está mostrado na Fig. 4.7.

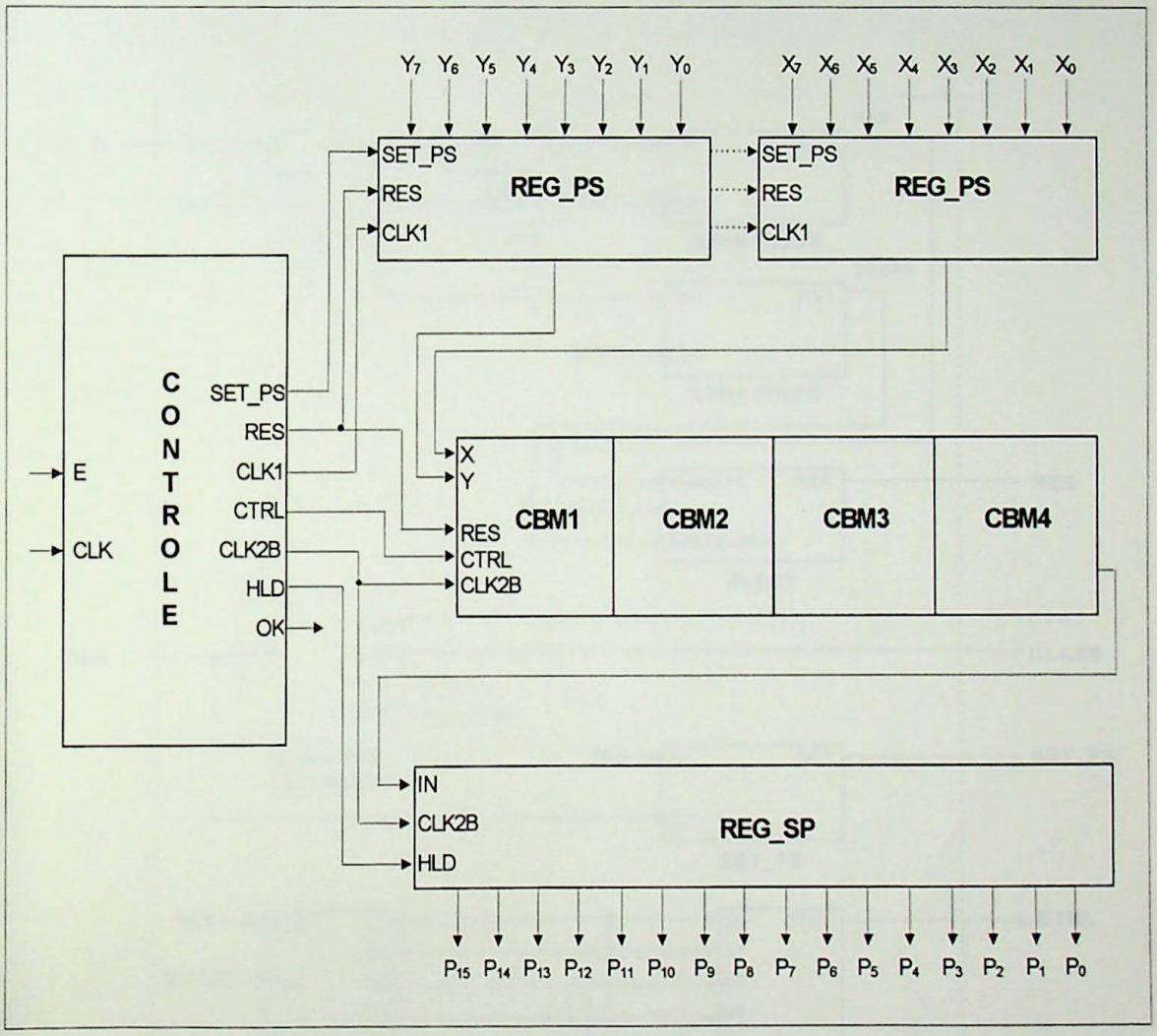


Fig. 4.7 Circuito Completo em Blocos.

No Cap. 3 já foi feita a descrição da Célula Básica de Multiplicação (CBM). A seguir será feita uma descrição dos blocos que compõem o restante do circuito.

4.4 O Bloco de Controle.

O Bloco de Controle é composto por vários módulos, cada um com uma função específica como mostra a Fig. 4.8. São eles: Gerador de Fases, Reset, Set_PS, Contador Síncrono de cinco bits, CTRL e HLD.

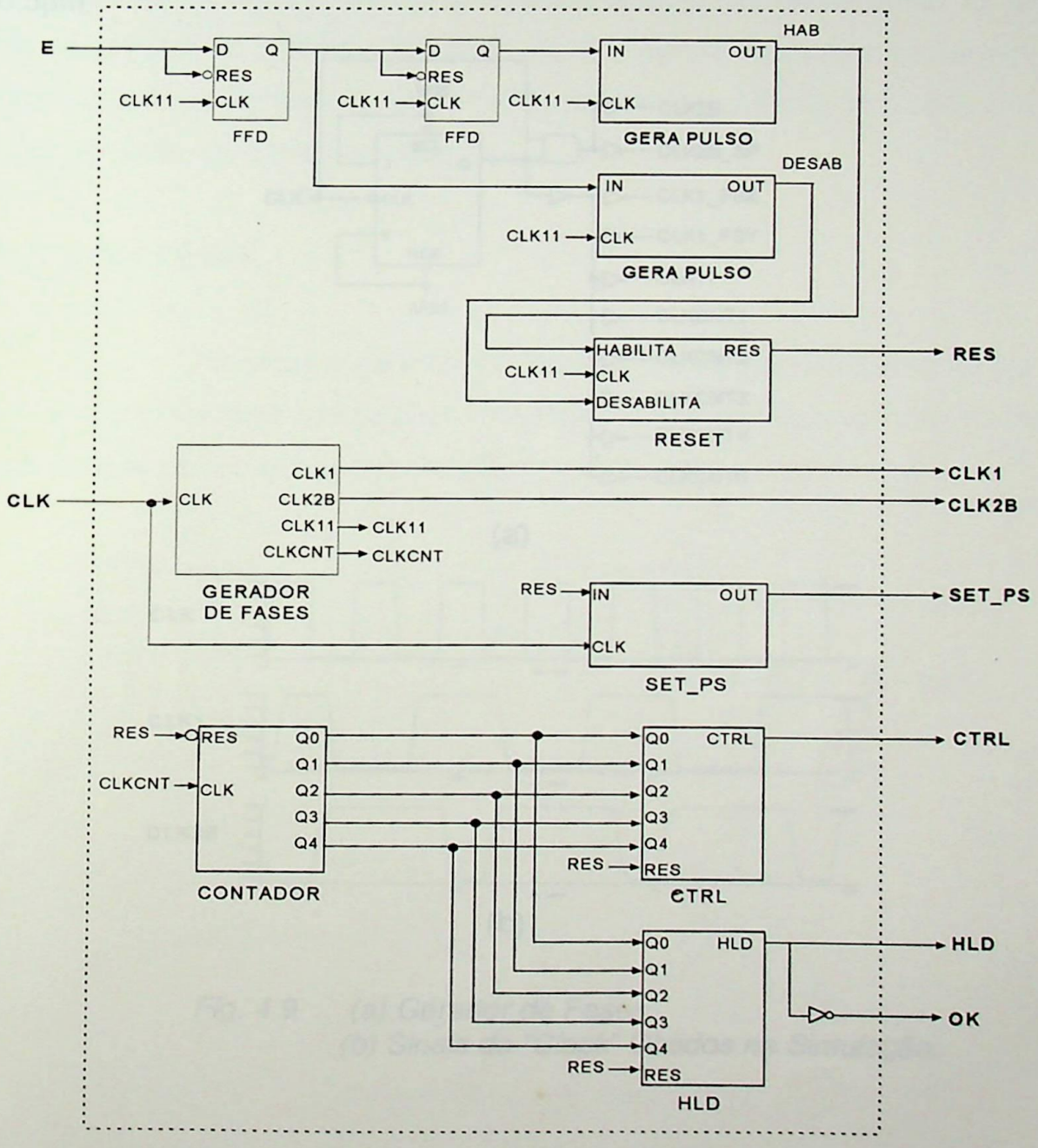
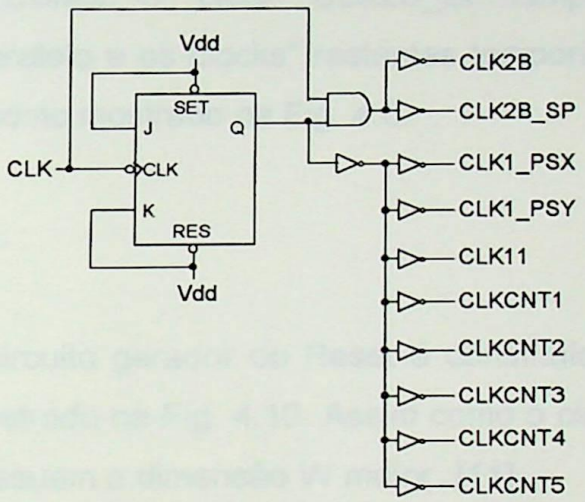


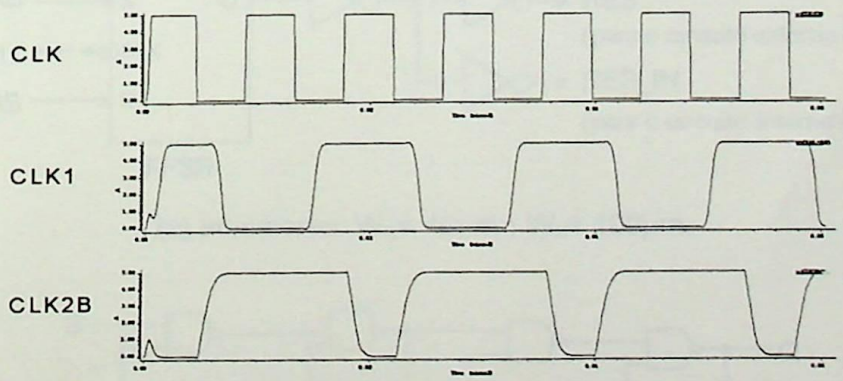
Fig. 4.8 Bloco de Controle.

4.4.1 O Gerador de Fases.

O gerador de fases é implementado com um flip-flop JK **[11]** como mostra a Fig. 4.9a. É importante observar que este gerador de fases é implementado com transistores com a dimensão W maior do que os utilizados nos demais circuitos que estão constituídos por transistores de tamanho W_N=2,5μm e W_P=6,5μm.



(a)



(b)

Fig. 4.9 (a) Gerador de Fases.
(b) Sinais de "Clock" Obtidos na Simulação.

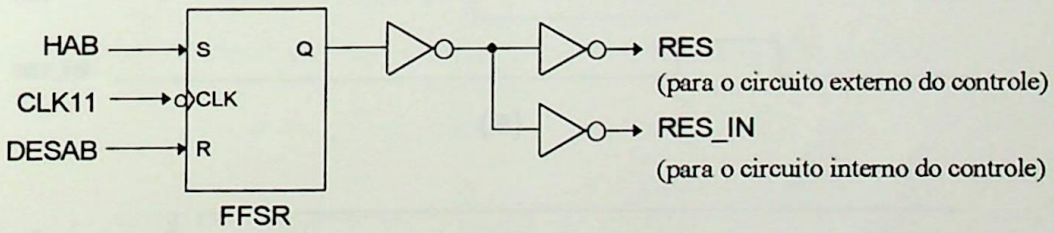
O tamanho dos transistores do flip-flop JK, da porta AND e dos cinco primeiros inversores, contando de cima pra baixo, é W_N=30μm e W_P=75μm. O tamanho dos transistores dos inversores restantes é W_N=40μm e W_P=100μm. Isto

foi feito para acelerar a resposta dos "cloks" e para reduzir o "fan-in" gerado pelos circuitos que serão conectados nestas saídas. A Fig. 4.9b mostra as curvas geradas. Observe-se que estas formas de onda estão de acordo com as formas de onda da Fig. 4.6.

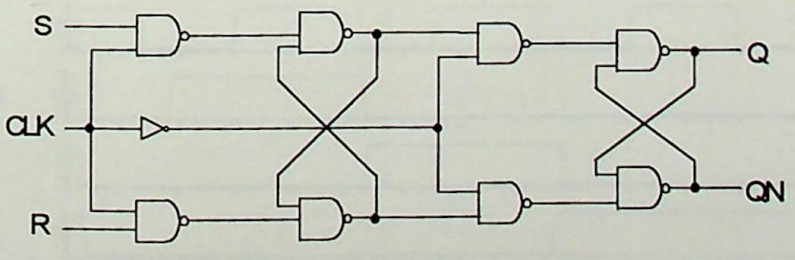
A saída dos "clocks" CLK1_PSX e CLK1_PSY temporizam os registradores de deslocamento Paralelo-Série X e Y respectivamente. O "clock" CLK2B temporiza as CBM's. O "clock" CLK2B_SP temporiza o registrador de deslocamento Série-Paralelo e os clocks" restantes temporizam os blocos internos do Bloco de Controle, como mostrado na Fig. 4.8.

4.4.2 O Bloco RESET.

O circuito gerador do Reset é constituído basicamente por um flip-flop SR, como o mostrado na Fig. 4.10. Assim como o circuito gerador de fases, os seus transistores possuem a dimensão W maior **[11]**.



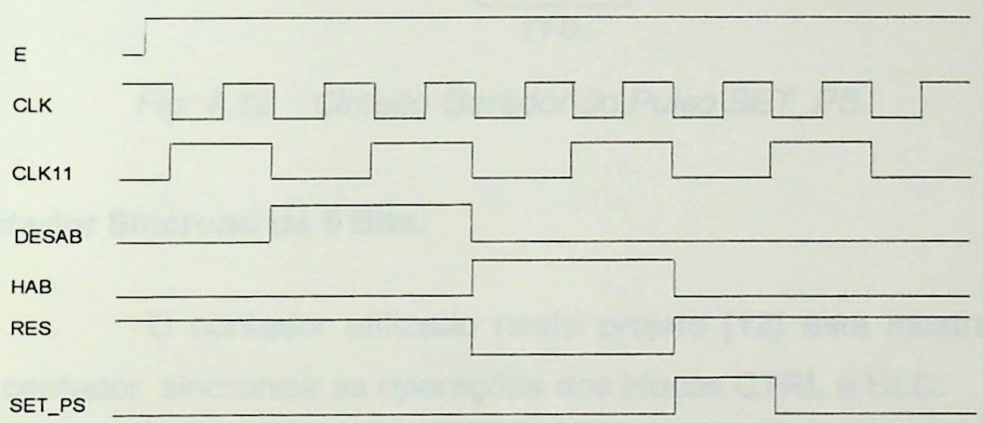
(a) inversores: W_N= 40μm ; W_P= 100μm.



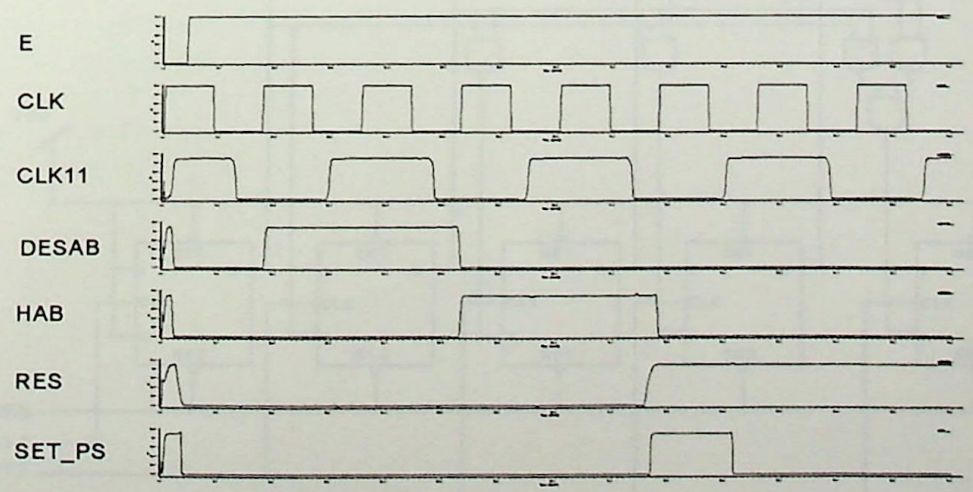
(b) W_N= 30μm ; W_P= 75μm.

Fig. 4.10 (a) Circuito do Reset .
(b) Flip-Flop SR.

O nível lógico E=1 (que dispara a multiplicação) chega na entrada S (habilita) do flip-flop SR já na forma de um único pulso (gerado por um bloco igual ao bloco Set_PS) passando antes por dois flip-flops D que atrasam este sinal dois ciclos do "clock" CLK11, observar Fig. 4.8. Estes atrasos sincronizam o sinal E com o "edge" negativo do "clock" CLK11. O mesmo sinal E=1 chega na entrada R (desabilita) do flip-flop SR, na forma de um único pulso e atrasado de um ciclo. Isto é feito para garantir a ***zeragem inicial ou Reset*** de todos os flip-flops do circuito do multiplicador antes do inicio de uma multiplicação. As formas de onda esperadas estão mostradas na Fig. 4.11a, e as formas de onda obtidas na simulação estão mostradas na Fig. 4.11b.



(a)



(b)

Fig. 4.11 (a) Formas de Onda Esperadas da Geração do RES e SET_PS.
(b) Formas de Onda Simuladas da Geração do RES e SET_PS.

4.4.3 O Bloco SET_PS.

Este pulso, mostrado na Fig. 4.11, habilita os registradores de deslocamento paralelo-série imediatamente após o sinal de reset RES ir para o nível lógico 1. O comprimento do pulso SET_PS não deve ser maior que um ciclo do "clock" CLK. O circuito que gera este pulso é encontrado em **[11]** e mostrado na Fig. 4.12.

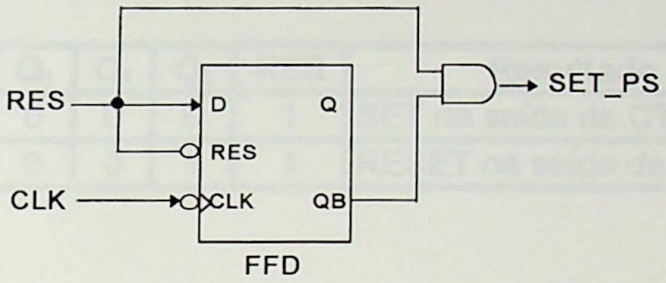


Fig. 4.12 Circuito Gerador do Pulso SET_PS.

4.4.4 Contador Síncrono de 5 Bits.

O contador utilizado neste projeto **[12]** esta mostrado na Fig. 4.13. Este contador sincroniza as operações dos blocos CTRL e HLD.

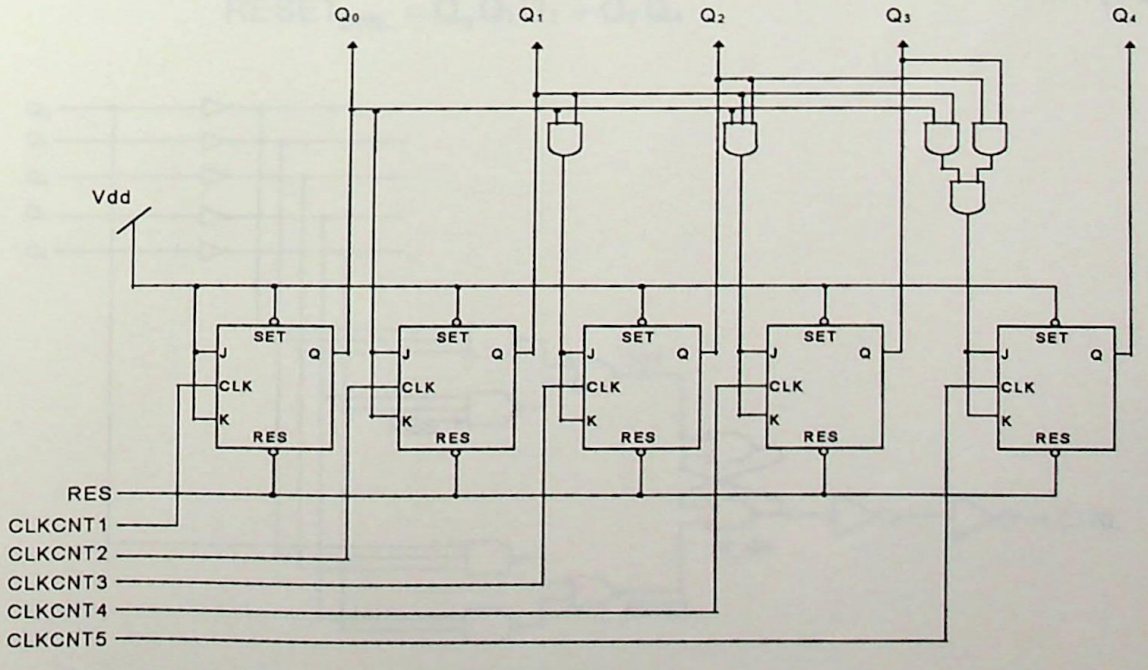


Fig. 4.13 Contador Síncrono de 5 Bits.

4.4.5 O Bloco CTRL.

O bloco CTRL tem a função de gerar um único pulso de duração de um ciclo do "clock" CLK1 assim que o sinal RES vai para nível lógico alto, com isto são monitoradas as chaves CH1 até CH8 (ver Fig. 3.10 do Capítulo 3). A implementação desta lógica está mostrada na Fig. 4.14. A lógica necessária para se gerar este pulso é a seguinte:

Período	Q_4	Q_3	Q_2	Q_1	Q_0	RES	Resultado
0T	0	0	0	0	0	1	SET na saída de CTRL
1T	0	0	0	0	1	1	RESET na saída de CTRL

Então:

$$SET_{CTRL} = \overline{Q}_0\overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4 RES$$

$$SET_{CTRL} = \overline{\overline{\overline{\overline{\overline{Q}_0\overline{Q}_1\overline{Q}_2} + \overline{\overline{Q}_3\overline{Q}_4}RES}}} \qquad (4\text{-}1)$$

e

$$RESET_{CTRL} = Q_0\overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4$$

$$RESET_{CTRL} = \overline{\overline{\overline{\overline{Q_0\overline{Q}_1\overline{Q}_2} + \overline{\overline{Q}_3\overline{Q}_4}}}} \qquad (4\text{-}2)$$

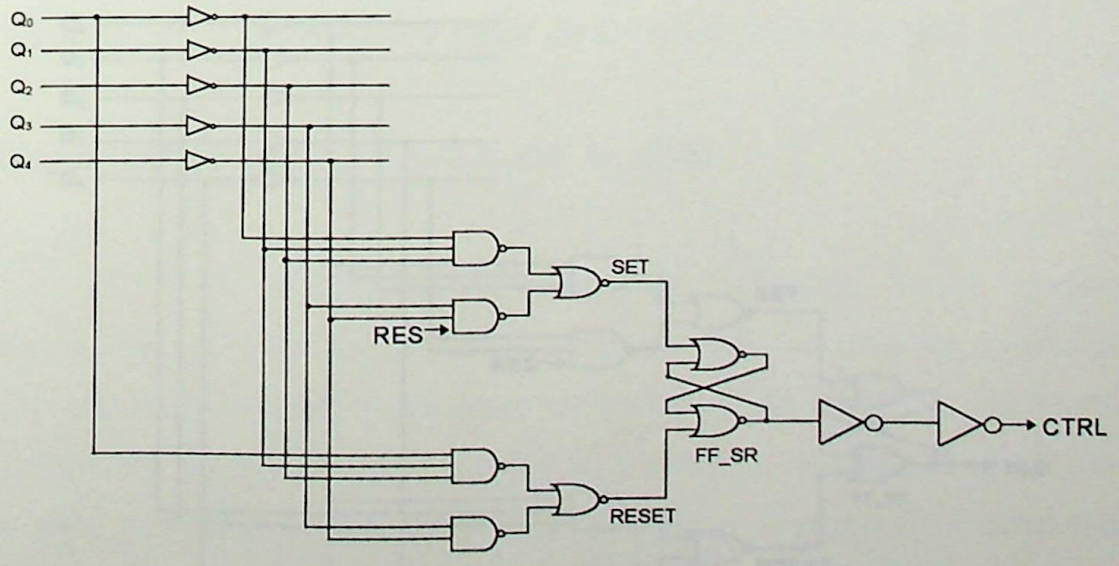


Fig. 4.14 Circuito do Bloco CTRL .

4.4.6 O Bloco HLD.

A função deste bloco é a de reter a informação armazenada no registrador série-paralelo, exatamente no final da multiplicação, isto é feito gerando-se um sinal "0" no período 22 do "clock" CLK1 (valor do número de ciclos que demora a multiplicação de dois operandos de 8 bits). A implementação desta lógica está mostrada na Fig. 4.15. A lógica necessária é :

Período	Q_4	Q_3	Q_2	Q_1	Q_0	RES	Resultado
0T	0	0	0	0	0	1	SET na saída de HLD
22T	1	0	1	1	0	1	RESET na saída de HLD

Então:

$$SET_{HLD} = \overline{Q}_0\overline{Q}_1\overline{Q}_2\overline{Q}_3\overline{Q}_4RES$$

$$SET_{HLD} = \overline{\overline{\overline{\overline{\overline{Q}_0\overline{Q}_1\overline{Q}_2} + \overline{\overline{Q}_3\overline{Q}_4}RES}}} \qquad (4\text{-}3)$$

e

$$RESET_{HLD} = \overline{Q}_0Q_1Q_2\overline{Q}_3Q_4$$

$$RESET_{HLD} = \overline{\overline{\overline{\overline{Q}_0Q_1Q_2} + \overline{Q}_3Q_4}} \qquad (4\text{-}4)$$

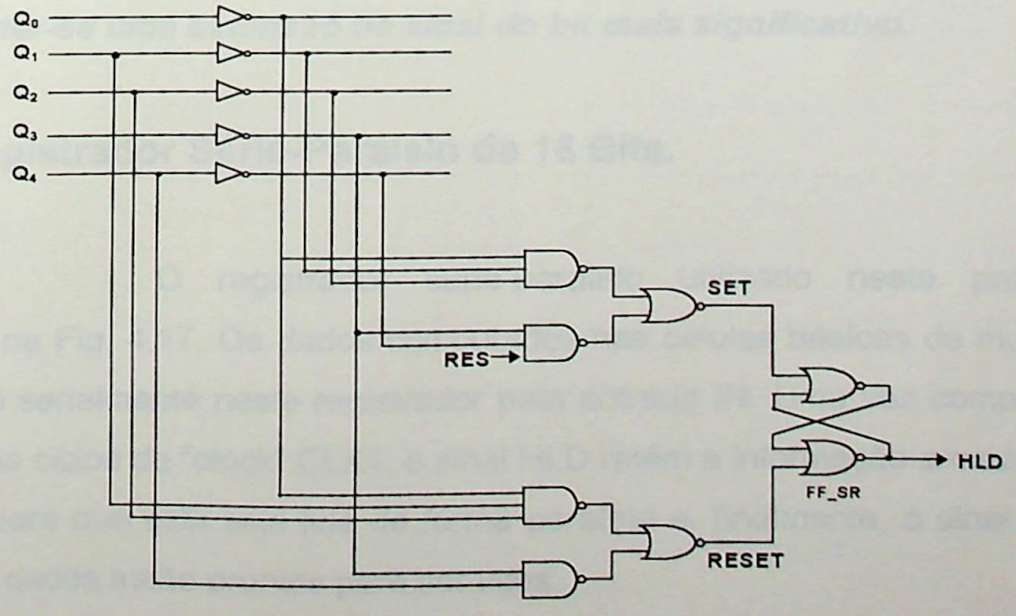


Fig. 4.15 Circuito do Bloco HLD.

4.5 Registrador Paralelo-Série de 8 Bits.

Uma vez que o pulso SET_PS habilita os registradores paralelo-série, as palavras de dado X_7 ... X_0 e Y_7 ... Y_0 atingem os flip-flops JK. A palavra armazenada agora pode ser lida serialmente um bit a cada pulso do "clock" CLK1, iniciando-se com o bit menos significativo. O circuito do registrador paralelo-série, usado neste projeto, esta mostrado na Fig. 4.16.

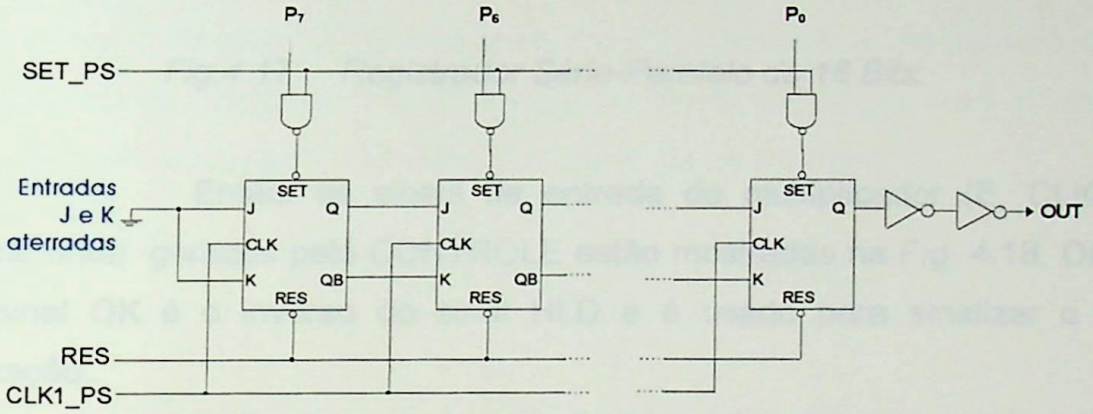


Fig. 4.16 Registrador Paralelo-Série de 8 Bits.

Uma observação importante é que as entradas J e K do primeiro flip-flop foram aterradas, logo quando as entradas estão em nível baixo a saída Q não muda (ver tabela de verdade da Fig. 4.3), isto foi feito para poder obter-se uma extensão de sinal do bit mais significativo.

4.6 Registrador Série-Paralelo de 16 Bits.

O registrador série-paralelo utilizado neste projeto está mostrado na Fig. 4.17. Os dados computados nas células básicas de multiplicação ingressam serialmente neste registrador pela entrada IN. Uma vez completados os vinte e dois ciclos do "clock" CLK1, o sinal HLD retém a informação armazenada nos flip-flops para que esta seja lida de forma paralela e, finalmente, o sinal OK indica que estes dados estão prontos para ser lidos.

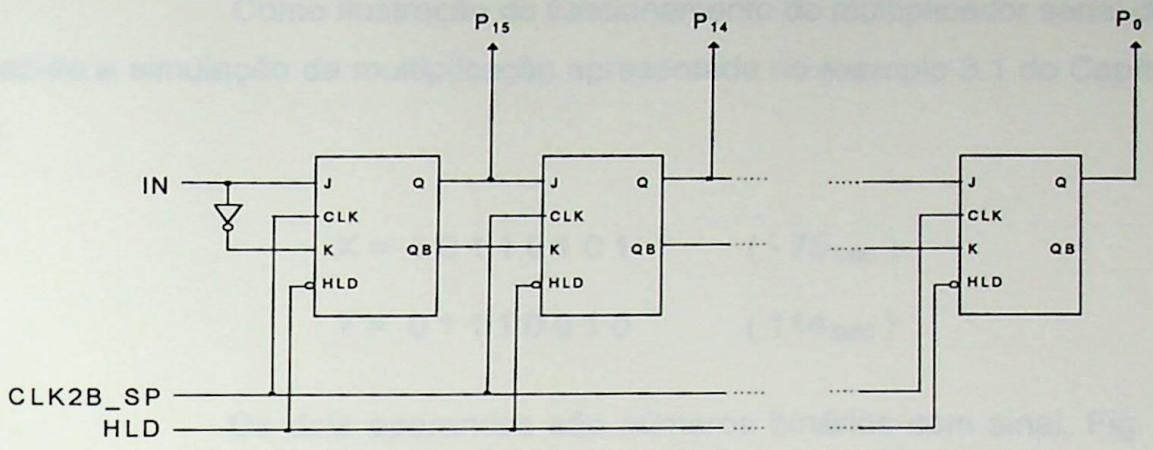


Fig.4.17 Registrador Série-Paralelo de 16 Bits.

Então, os sinais de entrada do multiplicador (E, CLK) e as formas de onda geradas pelo CONTROLE estão mostradas na Fig. 4.18. Observar que o sinal OK é o inverso do sinal HLD e é usado para sinalizar o fim da multiplicação.

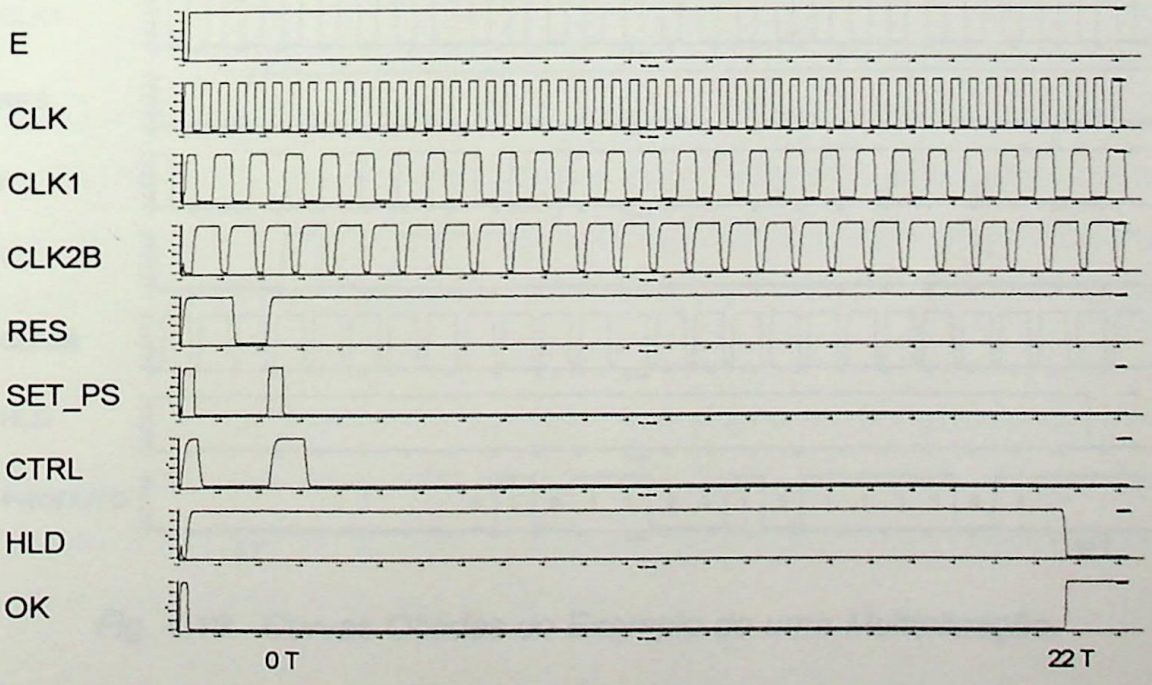


Fig. 4.18 Formas de Onda Geradas pelo Bloco de Controle.

Como ilustração do funcionamento do multiplicador serial de oito bits, fez-se a simulação da multiplicação apresentada no exemplo 3.1 do Capítulo 3. Onde:

$$X = 1\,0\,1\,1\,0\,1\,0\,1 \qquad (-75_{DEC})$$

$$Y = 0\,1\,1\,1\,0\,0\,1\,0 \qquad (114_{DEC})$$

Os dois operandos são números binários com sinal, Fig. 4.19, observe-se que o bit de sinal do operando X é "1" e portanto é um número negativo e o bit de sinal do operando Y é "0" e portanto é um número positivo. O resultado final de esta multiplicação é um número negativo de desseseis bits:

$$P = 1101\ 1110\ 1001\ 1010\ (-8550_{DEC})$$

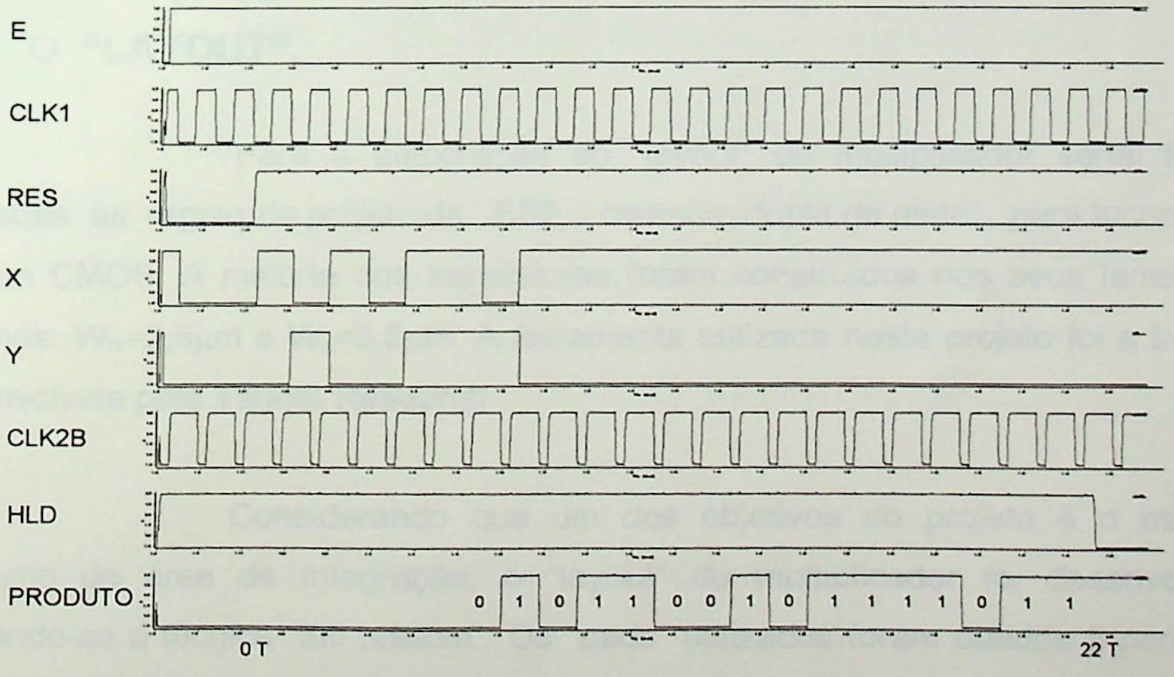


Fig. 4.19 Curvas Obtidas do Exemplo de uma Multiplicação.

Pode-se observar na Fig. 4.20 que, a simulação deste exemplo para todos os casos foi satisfatória.

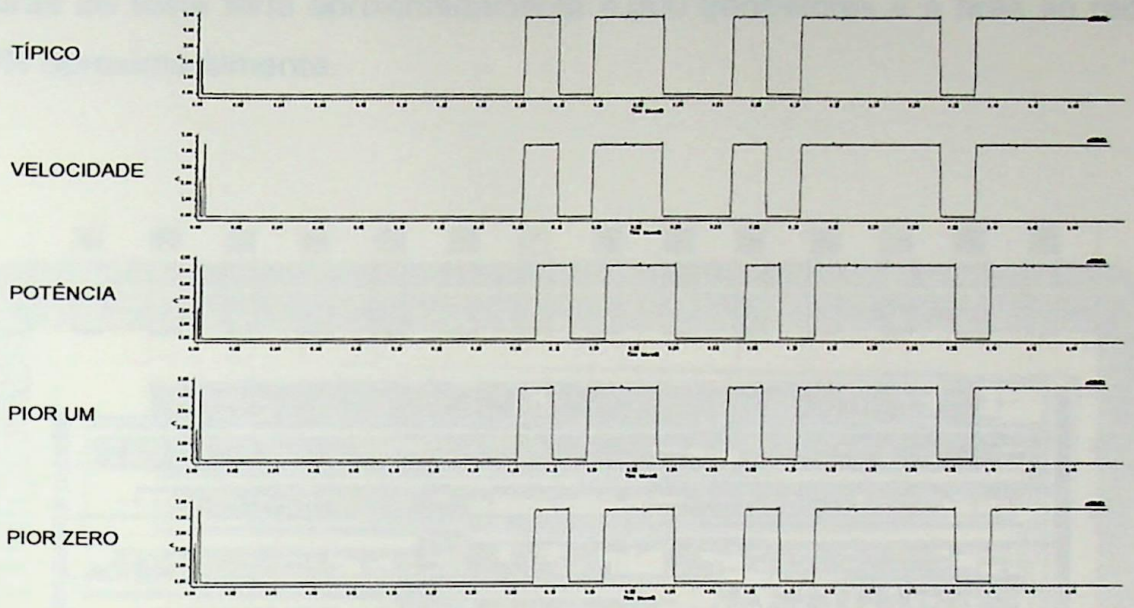


Fig. 4.20 Diferentes Casos de Simulação do Exemplo de uma Multiplicação.

4.7 O "LAYOUT".

Para a elaboração do "layout" do multiplicador serial foram utilizadas as regras de projeto da ES2, camada dupla de metal, para tecnologia 1,0 μm CMOS. A maioria dos transistores foram construídos nos seus tamanhos mínimos: W_N=2,5μm e W_p=6,5μm. A ferramenta utilizada neste projeto foi a L-Edit, desenvolvida pela Tanner Research.

Considerando que um dos objetivos do projeto é o mínimo consumo de área de integração, o "layout" do multiplicador foi desenvolvido utilizando-se a técnica "full custom". Os "pads" utilizados foram obtidos à partir da biblioteca fornecida pela ES2 para a Tanner.

O "layout" do Multiplicador Serial de 8 bits, mostrado na Fig. 4.21, conta com aproximadamente 5.500 transistores integrados em uma área de 4,8 x 3,5 mm^2 , incluindo-se os "pads". Foram colocados no "chip" estruturas de teste para eventuais medições, sendo assim, o protótipo sem estas

estruturas de teste teria aproximadamente 4.500 transistores e a área se reduziria em 20% aproximadamente.

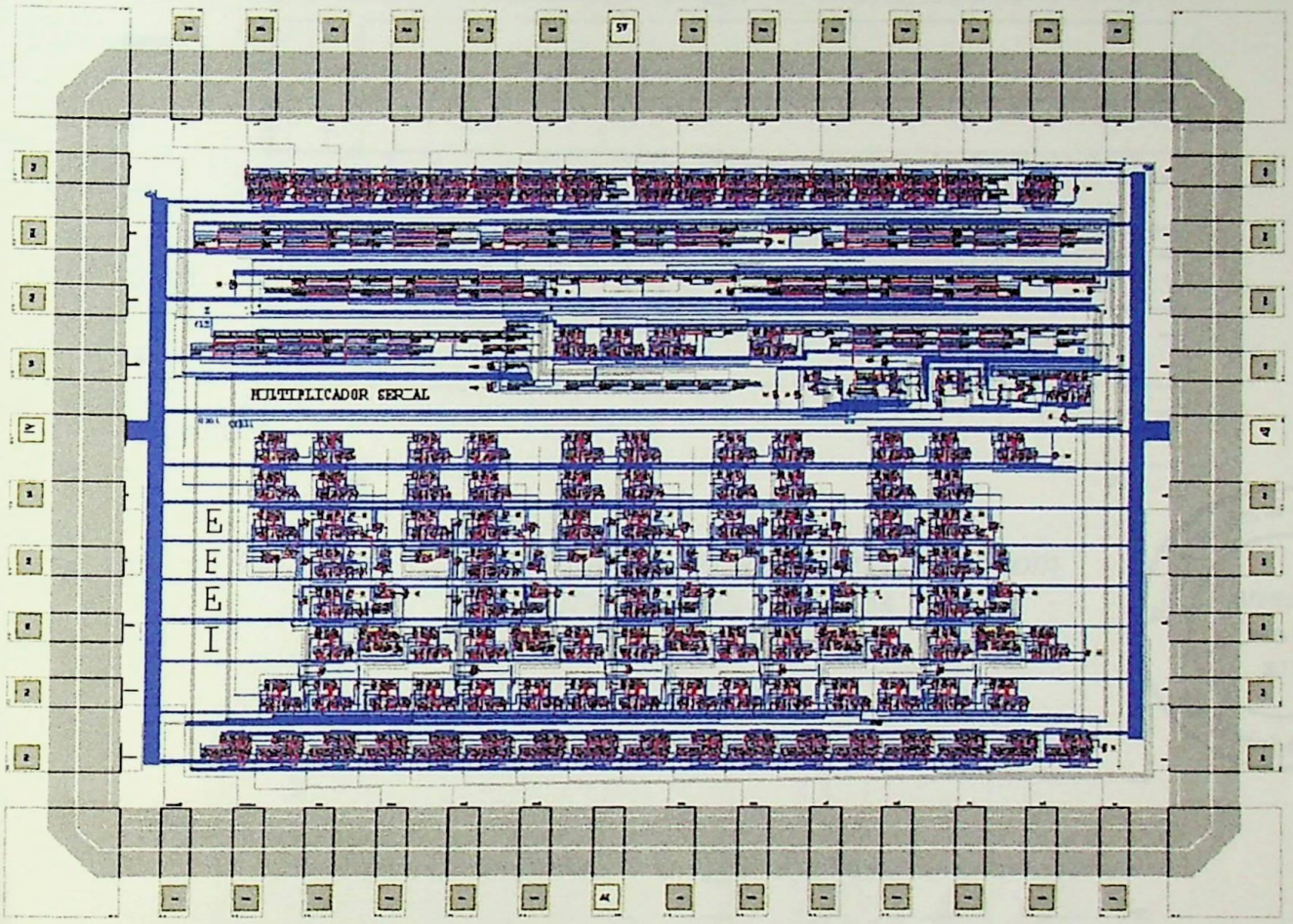


Fig. 4.21 Layout do Multiplicador Serial de 8 Bits.

4.8 Simulações com Capacitâncias Parasitas.

Foram obtidas as curvas das Figs. 4.22 e 4.23 a partir do netlist gerado com o extrator de parâmetros. Este netlist contém todas as capacitâncias parasitas geradas pelas diferentes camadas do "layout". Pode-se comparar estas curvas com as obtidas nas Figs. 4.19 e 4.20 e concluir que o multiplicador funciona corretamente mesmo com os atrasos adicionais gerados por estas capacitâncias.

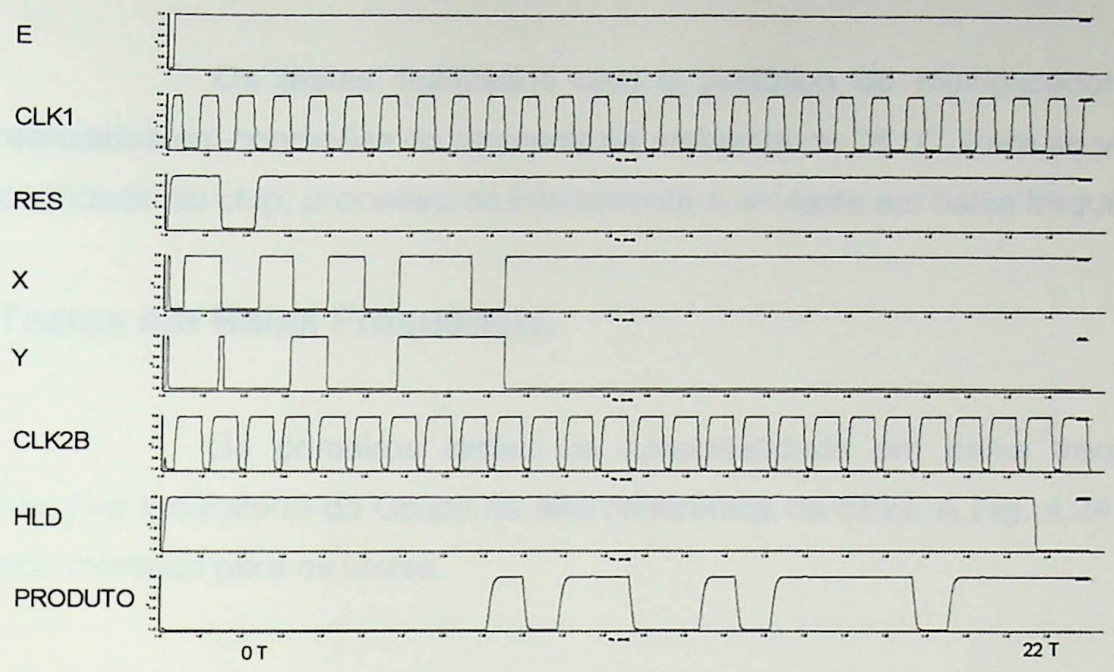


Fig. 4.22 Curvas Extraídas do Netlist do Layout com Capacitâncias Parasitas.

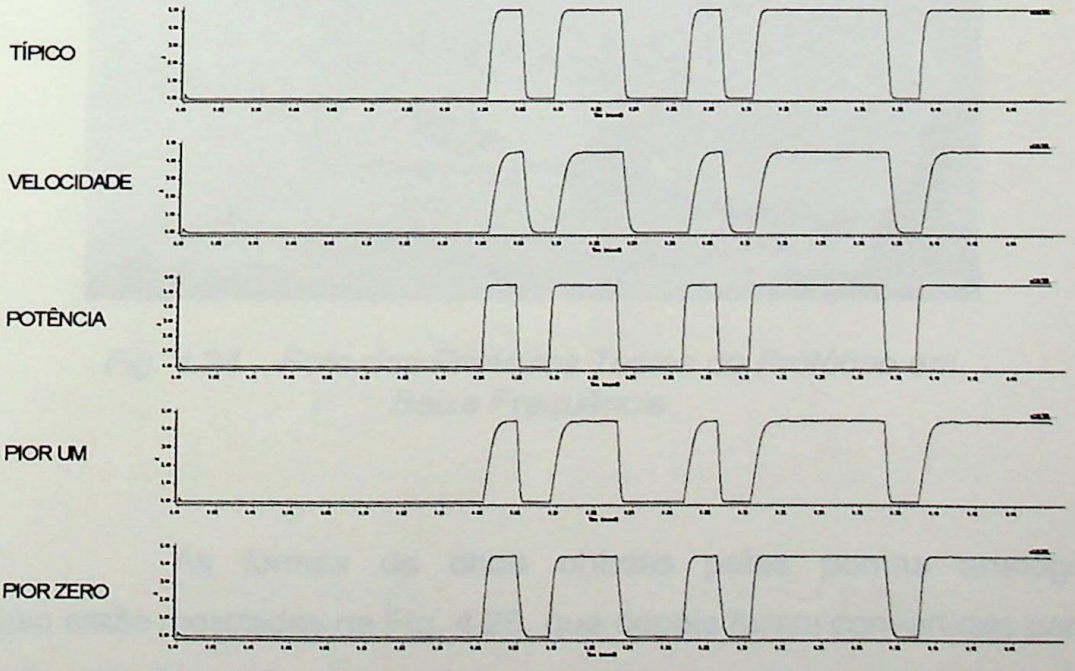


Fig. 4.23 Os Diferentes Casos de Simulação com Capacitâncias Parasitas.

4.9 Testes.

Os testes realizados com o protótipo do multiplicador serial foram realizados em condições de temperatura ambiente de 26 °C. Para se verificar a funcionalidade do chip, procedeu-se inicialmente a um teste em baixa frequência.

4.9.1 Testes em Baixa Frequência.

Os primeiros testes de funcionalidade em baixa frequência foram feitos no laboratório do Grupo de Microeletrônica da EFEI. A Fig. 4.24 ilustra a bancada montada para os testes.

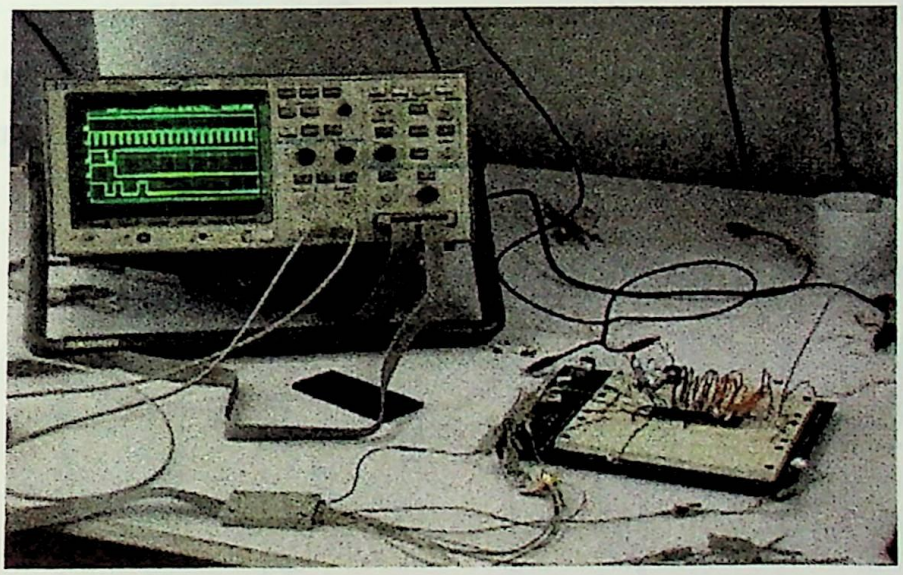


Fig. 4.24 Foto dos Primeiros Testes do Protótipo em Baixa Frequência.

As formas de onda obtidas pelas pontas analógicas do osciloscópio estão mostradas na Fig. 4.25, que depois foram convertidas para sinais digitais pelo próprio osciloscópio; assim, o pequeno ruído existente foi eliminado para depois se proceder às medições de atraso.

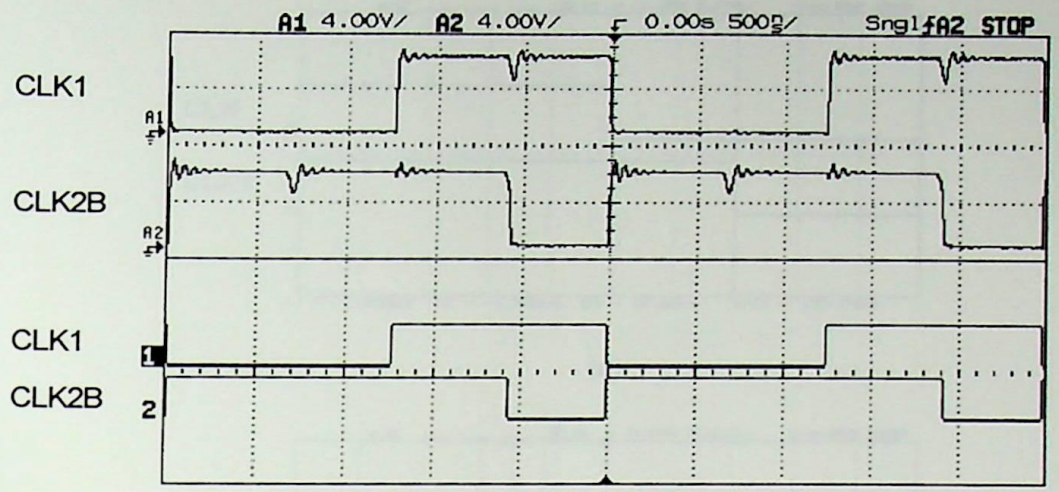


Fig 4.25 Sinais dos "clocks" Analógicos e Depois Digitalizados.

Uma vez digitalizados os sinais dos "clocks", pode-se fazer as medidas dos atrasos. Na Fig. 4.26 observam-se estes sinais a uma frequência de "clock" CLK de 820 KHz, e os atrasos estão mostrados na Fig. 4.27. No caso da Fig. 4.27a, o atraso obtido foi de 10,0 ns e no caso da Fig. 4.27b foi de 7,5 ns.

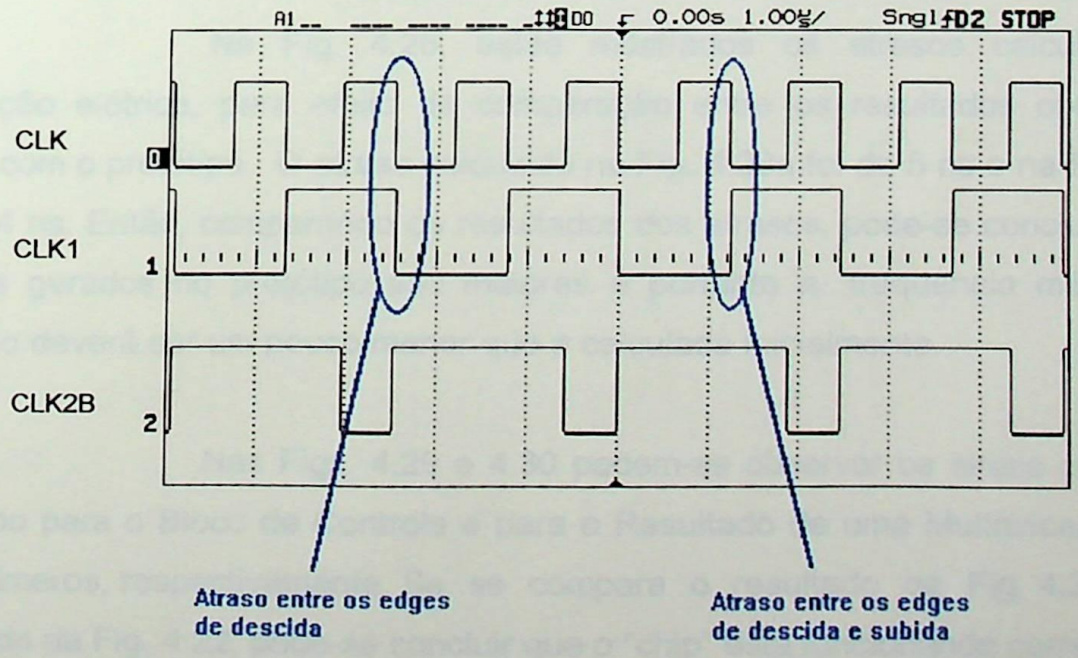
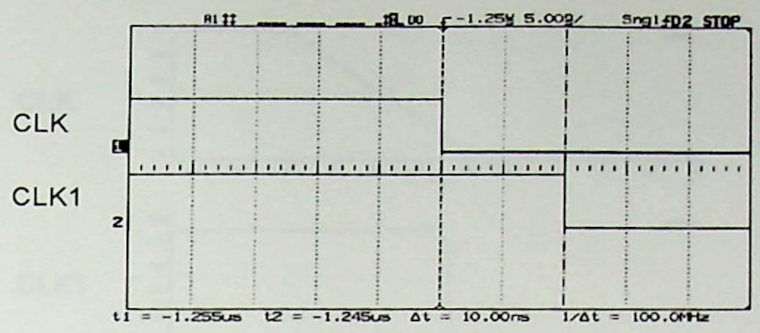
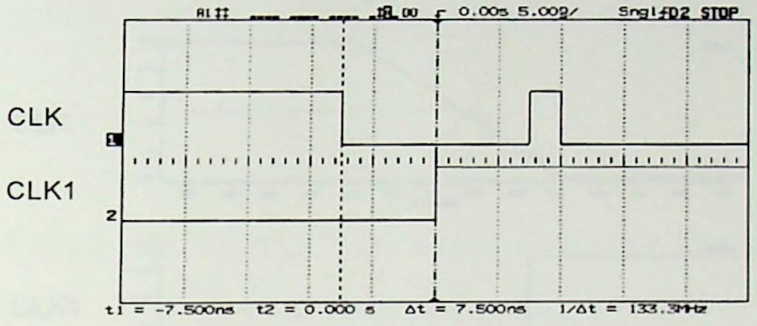


Fig. 4.26 Sinais dos "clocks" Obtidos nos Testes.



(a)



(b)

Fig. 4.27 (a) Atraso Entre as "Edges" de Descida.
(b) Atraso entre as "Edges" de Descida e Subida.

Na Fig. 4.28, estão mostrados os atrasos calculados na simulação elétrica, para efeito de comparação entre os resultados obtidos nos testes com o protótipo. O atraso calculado na Fig. 4.28a foi de 5 ns e na Fig. 4.28b foi de 4 ns. Então, comparando os resultados dos atrasos, pode-se concluir que os atrasos gerados no protótipo são maiores e portanto a "frequência máxima" de trabalho deverá ser um pouco menor que a calculada inicialmente.

Nas Figs. 4.29 e 4.30 podem-se observar os sinais obtidos do protótipo para o Bloco de Controle e para o Resultado de uma Multiplicação entre dois números, respectivamente. Se se compara o resultado da Fig. 4.30 com o resultado da Fig. 4.22, pode-se concluir que o "chip" está funcionando corretamente. Além desta multiplicação, foram feitas outras para se testar o correto funcionamento do protótipo.

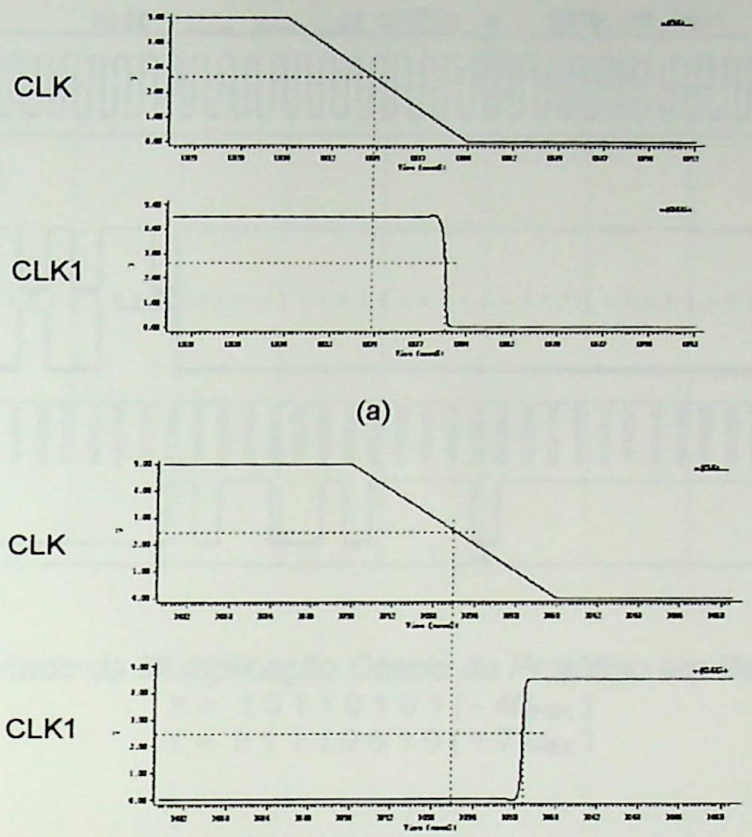


Fig. 4.28 (a) Atraso entre os "edges" de Descida.
(b) Atraso entre os "edges" de Descida e Subida.

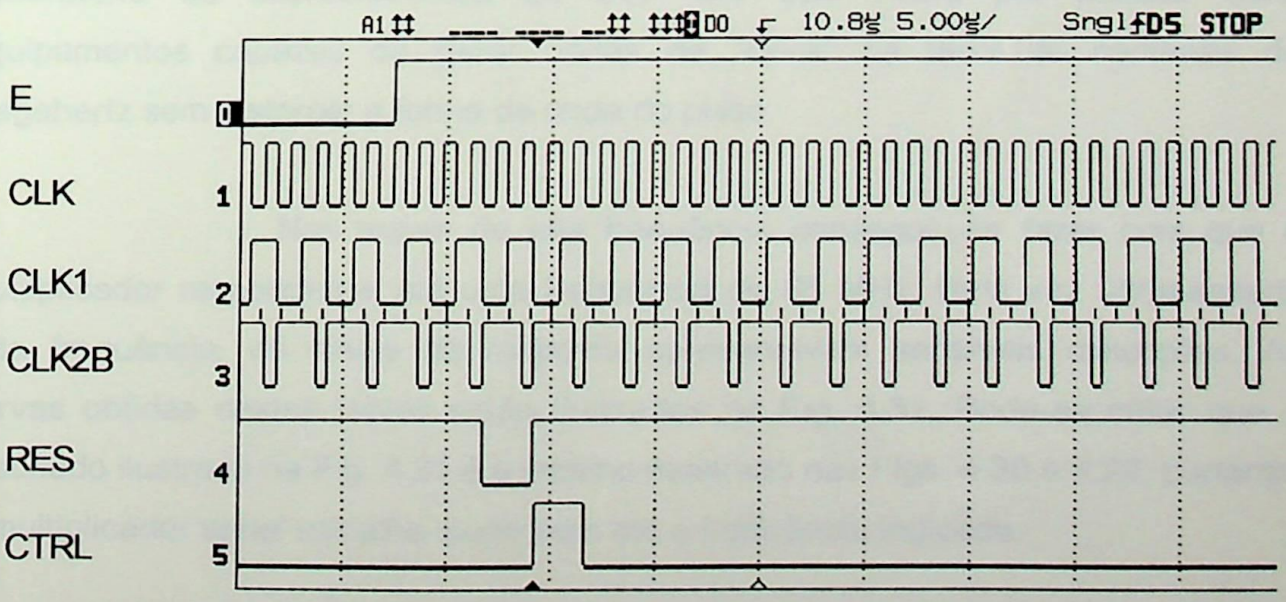


Fig. 4.29 Sinais do Bloco de Controle Obtidos do Protótipo em Baixa Frequência.

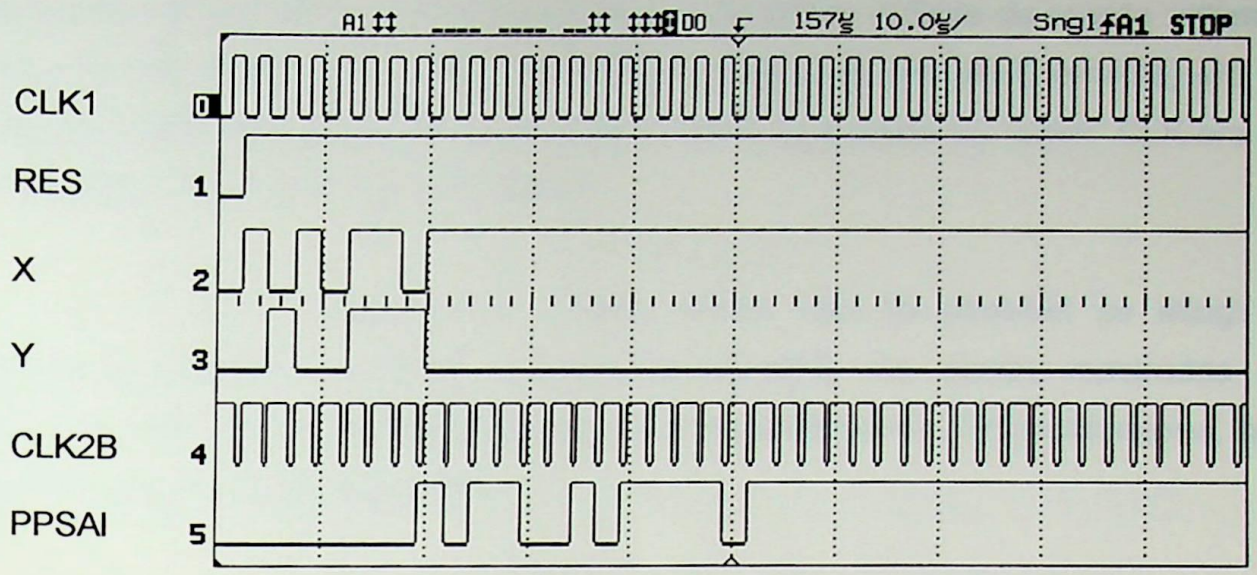


Fig. 4.30 Resultado da Multiplicação Obtido do Protótipo em Baixa Frequência.
X = 1 0 1 1 0 1 0 1 (- $4B_{HEX}$)
Y = 0 1 1 1 0 0 1 0 (+ 72_{HEX})

4.9.2 Testes em Alta Frequência.

Os testes em alta frequência tiveram que ser realizados no Laboratório de Microeletrônica da USP em São Paulo por possuir este, equipamentos capazes de gerar ondas de "clock" na faixa de centenas de megahertz sem distorcer a forma de onda do pulso.

Nos testes de alta frequência conseguiu-se fazer com que o multiplicador respondesse até uma frequência de 85 MHz. Uma vez ultrapassada esta frequência, os sinais de resposta apresentavam sensíveis distorções. As curvas obtidas nestes testes estão ilustradas na Fig. 4.31. Pode-se notar que o resultado ilustrado na Fig. 4.31 é o mesmo mostrado nas Figs. 4.30 e 4.22; portanto, o multiplicador serial trabalha muito bem até a frequência indicada.

A simulação indicava uma correta operação até 125 MHz. Então, deve-se considerar que os testes foram feitos em "protoboard", gerando

capacitâncias parasitas adicionais e reflexão de sinais. A fonte de tensão utilizada marcava, no instante dos testes, 5V e 70mA. O sinal de entrada E (enable) era de 3,3V em nível alto e 1,5V em nível baixo. O sinal de entrada de "clock" CLK era de 4V em nível alto e 0,7V em nível baixo.

Pelas razões citadas acima, não foi possível se atingir a frequência máxima projetada, que era de 125 MHz. As curvas mostradas na Fig. 4.31 representam a média obtida pelo osciloscópio em 32 amostragens, que conduziram a curvas mais nítidas.

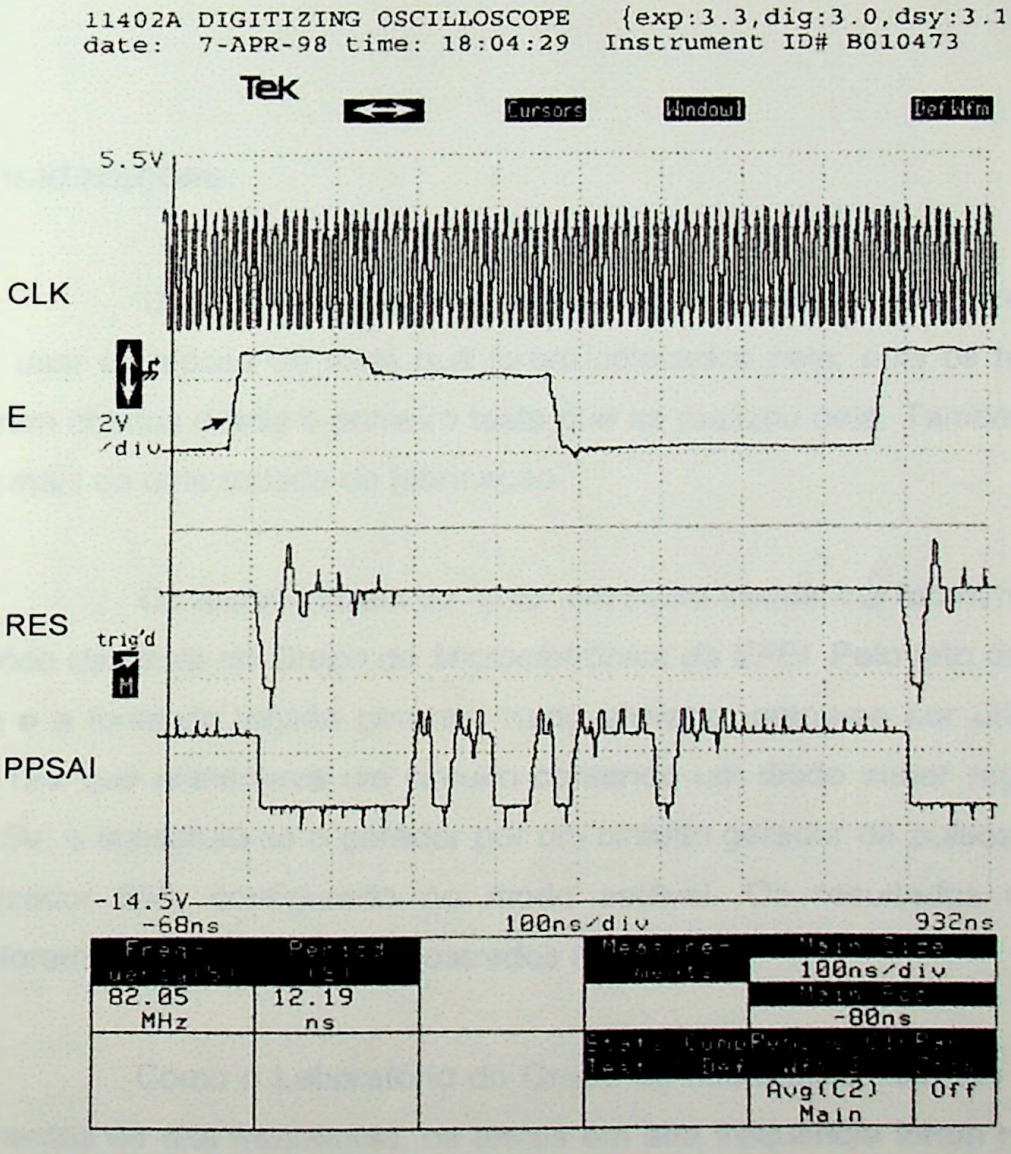


Fig. 4.31 Resultado da Multiplicação Obtido do Protótipo em Alta Frequência.

CAPÍTULO 5

CONSIDERAÇÕES E CONCLUSÕES FINAIS

5.1 Considerações.

Uma observação importante é que no primeiro protótipo, não foi necessário usar os blocos de teste que foram colocados nele, pois os resultados corretos foram obtidos desde o primeiro teste que se realizou nele. Também não foi necessária mais de uma rodada de fabricação.

Os testes iniciais do "chip" em baixa frequência foram realizados no *Laboratório de Teste do Grupo de Microeletrônica da EFEI*. Pelo fato do gerador de funções e a fonte de tensão gerarem ruído elevado, optou-se por utilizar uma bateria de 12V que alimentava um circuito contendo um diodo zener regulando a tensão em 5V, e substituiu-se o gerador por um circuito gerador de pulsos baseado no temporizador 555, configurado no modo astável. Os resultados em baixa frequência foram satisfatórios como mostrados no Cap. 4.

Como o Laboratório do Grupo de Microeletrônica não dispunha de equipamentos de alta frequência, os testes em alta frequência foram realizados no *Laboratório de Microeletrônica da Universidade de São Paulo*. Outra alternativa para os testes de alta frequência seria o *Centro Tecnológico de Informática de*

Campinas, CTI, mas o gerador de funções, em altas frequências, deformava as ondas quadradas inviabilizando qualquer resultado.

A Fig. 5.1 mostra a foto do protótipo que foi fabricado na foundry ES2 da França por meio do PMU (Programa Multiusuário Brasileiro) coordenado pelo CTI. O integrado demandou cinco meses para ser difundido, alterando o cronograma original previsto para este trabalho. A Fig. 5.2 mostra uma foto do "die" tirada com ajuda de um microscópio, ajustado para um aumento de cinquenta vezes.

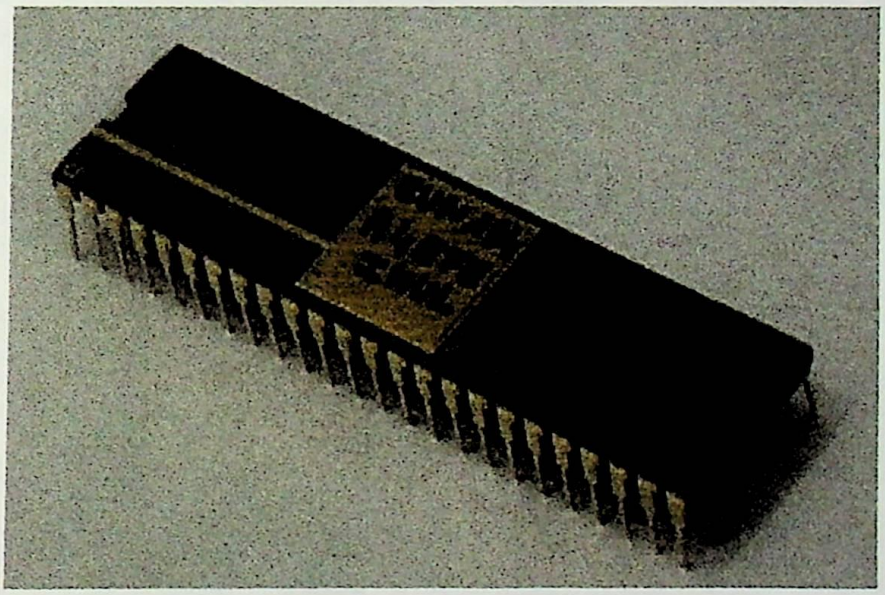


Fig. 5.1. Foto do Protótipo.

Esta dissertação deu origem a três trabalhos. O primeiro intitulado, *"Optimization of Cheng's Cell and its Application to a Serial Multiplier Based on the Booth's Modified Algorithm"*, foi apresentado no XII Congresso da SBMicro realizado em Caxambu M.G. em agosto de 1997 **[16]**. O segundo, com o título *"Optimization of Cheng's Cell and its Application to a 125 MHz Serial Multiplier Based on the Booth's Modified Algorithm"* , foi apresentado no Primer Workshop de Microelectrónica realizado em Rosario-Argentina em outubro do mesmo ano **[17]**. O

terceiro trabalho, "*A 85-MHz 8 x 8-bit Serial Pipelined Multiplier Based on The Cheng's Cell* " foi enviado ao IEEE.

Buscou-se, neste trabalho atingir outros objetivos como o aprendizado e a validação do software da Tanner Research. Também foi elaborado um tutorial que se encontra na biblioteca do Grupo de Microeletrônica da EFEI para orientar futuros trabalhos. O software mostrou-se bastante eficiente para o trabalho com realização de circuitos relativamente grandes, tanto na captura esquemática como na edição de "layout".

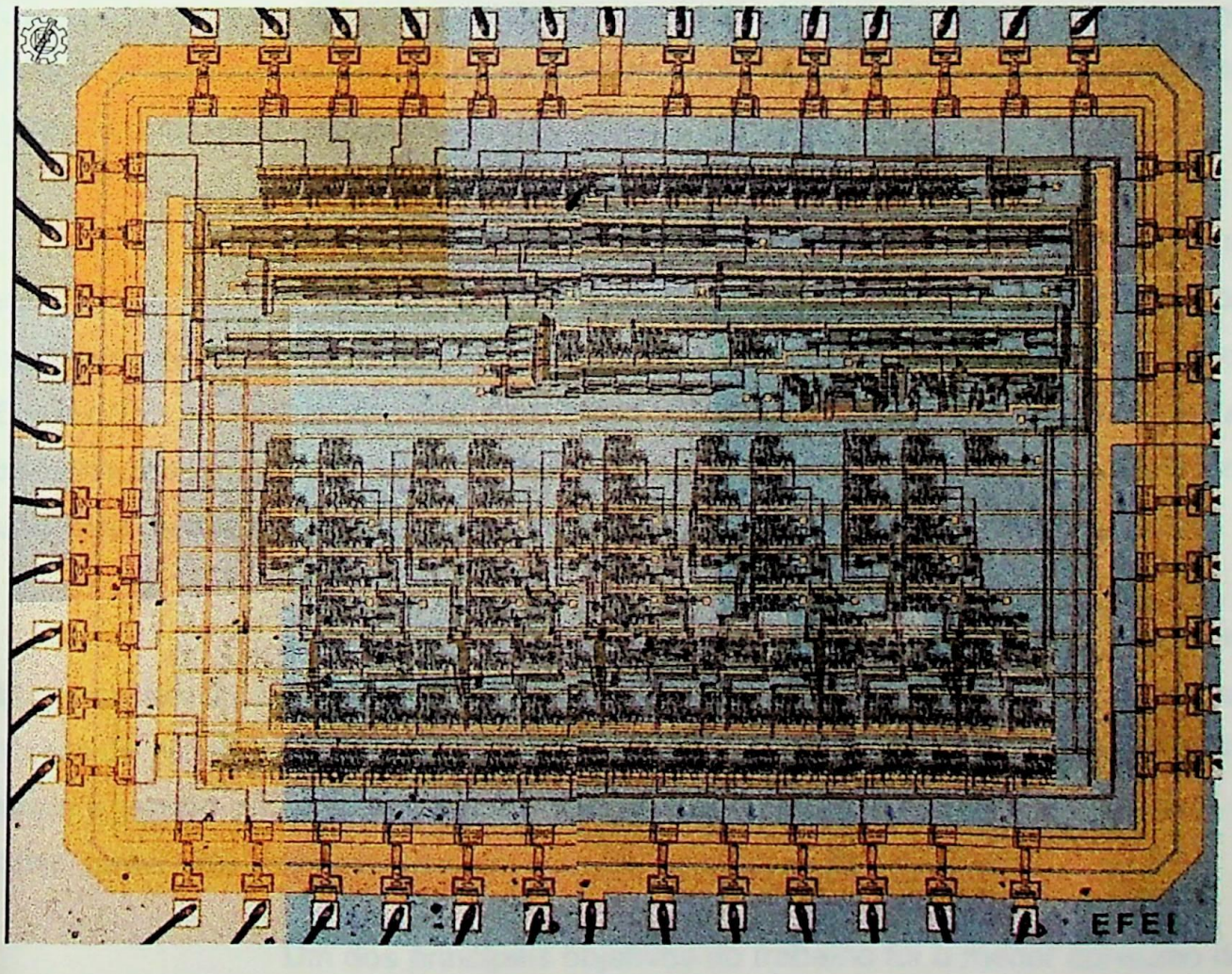


Fig. 5.2. Foto do "Die" Tirada com Microscópio.

5.2 Conclusões.

Após o estudo e a analise dos resultados conseguidos, chegou-se às conclusões que são apresentadas na sequência.

A modificação da *Célula de Multiplicação Serial de Cheng*, apresentada no Cap. 3, torna-a mais rápida e 100% confiável. A célula original elaborada por E. K. Cheng apresentava alguns erros durante a operação de multiplicação como foi demonstrado no Cap. 3.

A escolha do Algoritmo de Booth Modificado, mostrou ser uma boa opção, pois ele mantém a modularidade da célula de multiplicação e ao mesmo tempo acelera o processo da multiplicação. Com isso pode-se facilmente trabalhar com palavras maiores que 8 bits.

A frequência máxima de trabalho do "clock" obtida na simulação foi de 125 MHz na tecnologia CMOS de 1μm da ES2; portanto, a taxa de multiplicação seria superior a 2,7 milhões de multiplicações por segundo. Entretanto, os testes em alta frequência mostraram que o multiplicador serial trabalha sem problemas até uma frequência em torno de 85 MHz, implicando em uma taxa de multiplicação em torno de 1,9 milhões de multiplicações por segundo.

A queda na frequência indicada acima é devido principalmente às capacitâncias originadas pelo "protoboard" utilizado assim como pelos cabos das pontas de prova. Adicionalmente outros fatores que contribuíram para esta redução foram a tensão de entrada baixa do "clock" que ficou em torno de 3,3V, embora o "chip" tenha sido projetado para uma tensão de entrada de "clock" de 5V.

Um dos principais objetivos do trabalho foi o menor consumo de área de integração possível. Considerando-se que este multiplicador serial seja parte de um sistema digital, a área consumida sem os "pads" é de 3,2 x 2,3 mm^2, que é bastante pequena se comparada com a área consumida com "pads" que é de 4,8 x 3,5 mm^2 .

É importante ressaltar que sempre existiu uma busca por integrar todo tipo de sistemas, e para isto é imprescindível o menor consumo de área. Embora o multiplicador paralelo seja comprovadamente o mais veloz dos multiplicadores, ele ainda ocupa uma área maior do que o multiplicador serial. Portanto, em projetos nos quais a velocidade do processamento não é tão fundamental, mas sim a área de integração, os multiplicadores seriais se traduzem na melhor opção pois consomem uma área reduzida e possuem uma alta performance.

Então, com a relação área x performance garantida, sugere-se a continuidade de estudos de multiplicadores seriais ainda mais velozes com ajuda de algoritmos mais poderosos que o de Booth. Com os avanços dos processos da microeletrônica, reduzindo o comprimento de canal, as velocidades de operação dos multiplicadores seriais crescerão acentuadamente, tornando-os capazes de operar em frequências que até agora estavam proibidos.

ANEXO

COMPARAÇÃO ENTRE AS TÉCNICAS "FULL CUSTOM" E "FPGA"

1. CONSIDERAÇÕES SOBRE FPGA's.

Os FPGA's (Field Programmable Gate Array) são dispositivos programáveis cuja arquitetura interna está baseada em "Gate Arrays". As conexões entre os blocos e as trilhas são configuráveis pelo usuário, a través de elos. Os elos usados na arquitetura dos FPGA's são elos do tipo antifusiveis ou transistores controlados por RAM estática.

A configuração de um circuito FPGA com SRAM's é atualmente a mais utilizada já que o dispositivo pode ser reprogramado a qualquer momento e todas as vezes que o usuário quiser.

2. RESULTADOS OBTIDOS COM A TÉCNICA "FPGA".

A Fig. A1 mostra o circuito implementado com o software MAX+PLUS II da Altera Corporation, e a Fig. A2 mostra o resultado obtido de uma multiplicação, pode observar-se que é o mesmo resultado da Fig. 4.19 do Capítulo 4. Adicionalmente, a Fig. A2 mostra os estados finais das saídas do registrador

série-paralelo, quando a saída HLD vai para o nível lógico zero. A frequência máxima de trabalho obtida com a técnica FPGA é de 4.2 MHz.

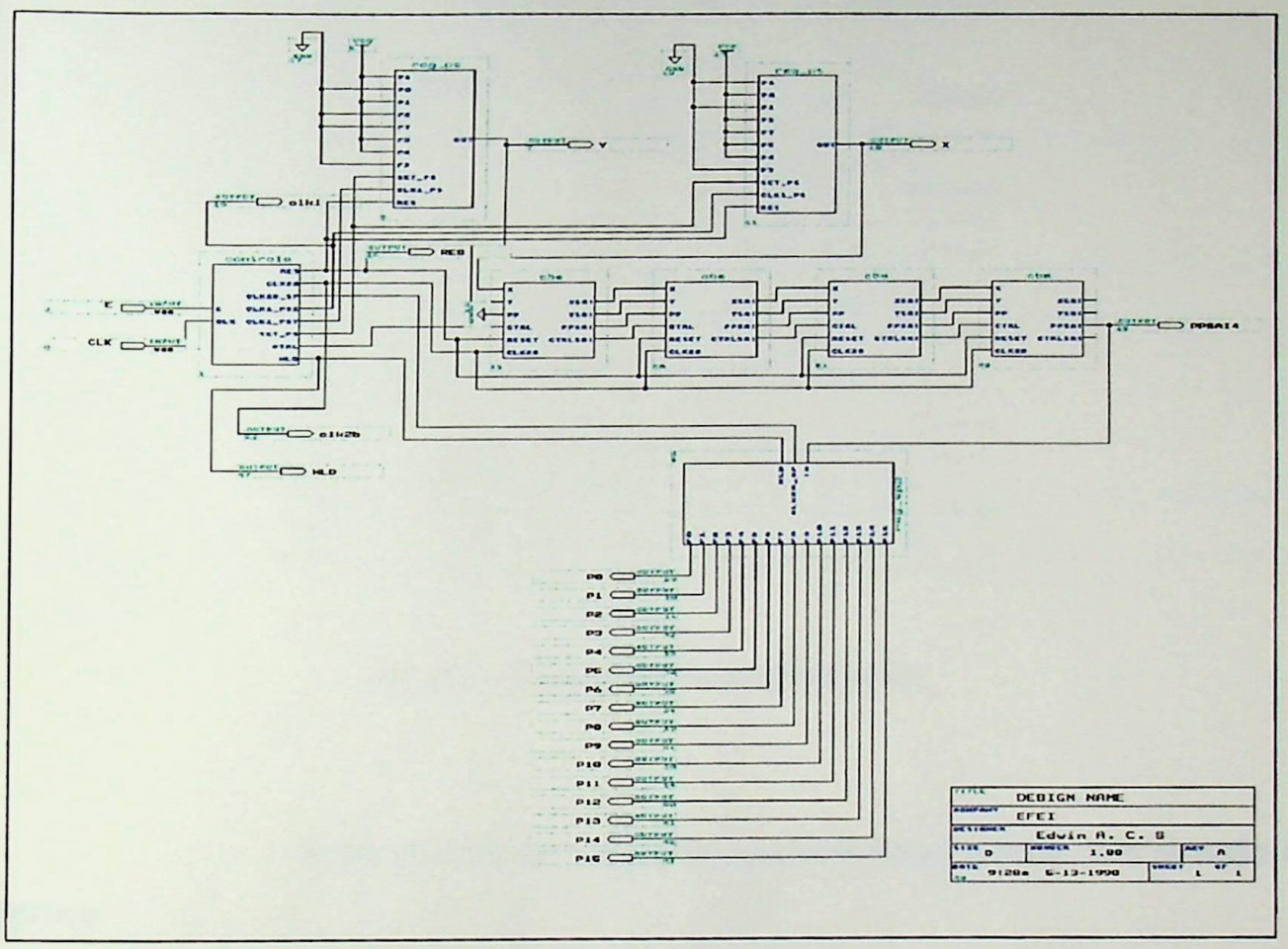


Fig. A1 - Circuito do Multiplicador Serial Implementado em FPGA.

3. OBSERVAÇÕES.

No exemplo ilustrado do multiplicador serial de oito bits utilizando a célula modificada, se conseguiu uma taxa de multiplicação superior a 1,9 milhões de multiplicações por segundo com a frequência de 85 MHz para Full Custom. Para FPGA se chegou a uma taxa de 95,5 mil multiplicações por segundo com a frequência de 4,2 MHz.

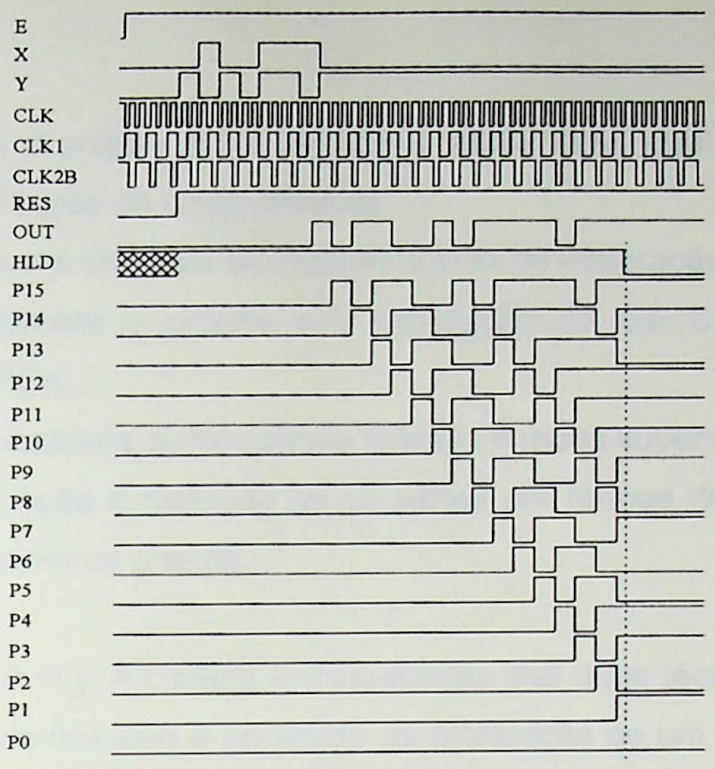


Fig. A2 - Resultado da Multiplicação.

Pode-se citar as seguintes características gerais com relação à técnica:

FPGA :

- A matriz de gates é programável pelo usuário, por isso um projeto pode ser modificado a qualquer hora e rapidamente;
- Não existe eficiência na utilização da área integrada;
- Alta conveniência no seu uso, com relação à rapidez com que um projeto pode ser integrado;
- Perdida de performance com relação à velocidade de resposta do circuito implementado, quando é comparado com outras técnicas de integração;
- Custo elevado, em comparação a outros circuitos integrados para aplicações específicas (ASIC's).

"Full Custom":

- Em caso de que o projeto sofra correções, estas terão que passar por todo o processo de fabricação do CI novamente;
- Esta técnica é a mais eficiente em relação à área de integração;
- O tempo que demora o projeto e a fabricação de um CI é elevado se é comparado ao FPGA;
- A velocidade de resposta ou frequência máxima é muito superior à de um FPGA;
- O custo de fabricação é reduzido se se pensa em termos de área no caso do projeto ser relativamente grande.

A Fig. A3 ilustra a comparação das duas técnicas em relação à frequência máxima de trabalho e ao tempo de fabricação de um multiplicador serial de oito bits.

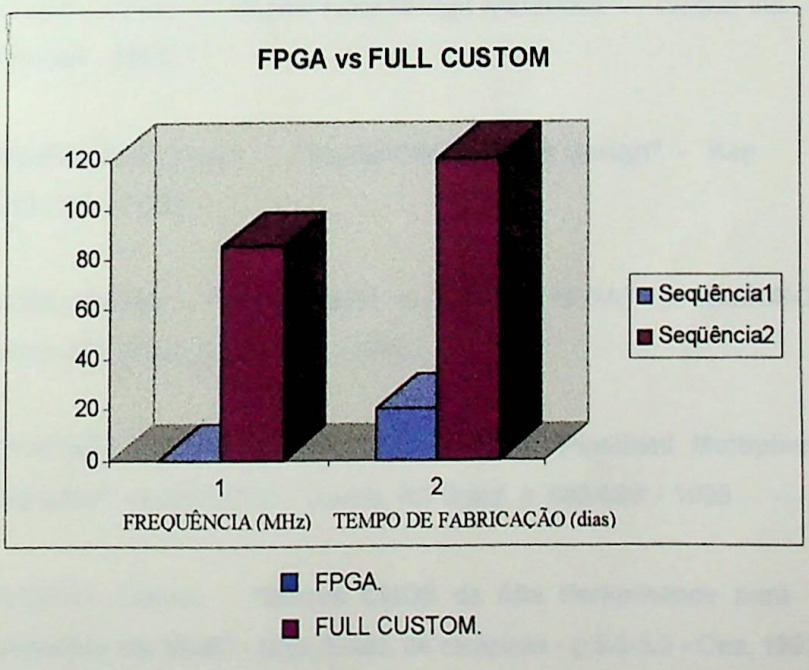


Fig. A3 - Comparação das Duas Técnicas.

Bibliografia Referenciada

[1] BAUGH, Charles R., WOOLEY, Bruce A. - **"A Two's Complement Parallel Array Multiplication Algorithm"** - IEEE Transactions on Communications - v.C-22 - p.1045-1047 - 1973.

[2] KAI, Hwang, FAYÉ, A. Briggs - **"Computer Architecture and Parallel Processing"** - McGraw-Hill - p.21 - 1984.

[3] LYON, R. F. - **"Two's Complement Pipeline Multipliers"** - IEEE Transactions on Communications - p.418-425 - April 1976.

[4] NEIL, H. E. Weste, KAMRAM, Eshraghian - **"Principles of CMOS VLSI Design"** - Addison-Wesley - p.339-348 - 1985.

[5] CHU, Y. - **" Digital Computer Design Fundamentals"** - New York. McGraw Hill - p.24-35. - 1984

[6] TAYLOR, J. Fred - **"Digital Filter Design Handbook"** - Dekker Inc. p.573-583 - 1983.

[7] ANNARATONE, Marco - **"Digital CMOS Circuit Design"** - Kap p.209-229 - 1986.

[8] SUZIM, Altamiro - **"Multiplicador de 8 Bits em FPGA"** - X SBMICRO - Canela, RS Brazil, p. 634-643 - 1995 .

[9] ROYANNEZ, P., GREINER, A. - **"Booth Multi - Pipelined Multiplier Generator"** - X SBMICRO - Canela, RS Brazil, p. 686-689 - 1995 .

[10] CALDEIRA, Laércio - **"Blocos CMOS de Alta Performance para Aplicações em VLSI"** - Univ. Estad. de campinas - p.5.8-5.9 - Dez. 1993.

[11] STOUT, David, KAUFMAN, Milton - **"Handbook of Microcircuit Design and Application"** - McGraw-Hill - p.4-1 - 4-13; 9-1 - 9-4 - 1980.

[12] MALVINO, Albert - **"Digital Computer Electronics"** - McGraw-Hill - p.136-139 - 1977.

[13] KAI, Hwang, FAYÉ, A. Briggs - **"Computer Architecture and Parallel Processing"** - McGraw-Hill - p.170-174 - 1984.

[14] CAPPELLO R. Peter, STEIGLITZ, Kenneth - **"A VLSI Layout for a Pipelined Dadda Multiplier"** - ACM Transactions on Computer Systems, Vol. 1, No. 2 - p. 157-174 - May 1983.

[15] RABEY M. Jan - **"A Design Perspective"** - Prentice Hall Electronics and VLSI Series - p. 240-242 - 1996.

[16] Sánchez Edwin, Caldeira L. - **"Optimization of Cheng's Cell and its Application to a Serial Multiplier Based on the Booth's Modified Algorithm"**- SBMicro Caxambú Agosto 1997.

[17] Sánchez Edwin, Caldeira L. - **"Optimization of Cheng's Cell and its Application to a 125 MHz Serial Multiplier Based on the Booth's Modified Algorithm"**- Primer Workshop de Microelectrónica Rosario - Argentina Outubro 1997.

Bibliografia Recomendada

BOOTH, A. D. - **"A Signed Binary Multiplication Technique"** - J. Mech. Appl. Math. 4 - p.236-240 - 1951.

GLASSER, A. Lance, DOBBERPUHL, W. Daniel - **"The Design And Analysis of VLSI Circuits"** Addison-Wesley - p.52-55 - 1985.

JACKSON, B. Leland, KAISER, F. James, McDONALD, S. Henry **"An Approach To The Implementation Of Digital Filters"** - IEEE Transactions on Audio and Electroacoustics, v.AU-16, n.3 - p.413-421 - set. 1968.

WARE, A. Frederick et al. - **"64 Bit Monolithic Floating Point Processors"** IEEE Journal Of Solid-State Circuits, v.SC-17, n.5 - p.898-907 - out. 1982.

[12] MALVINO, Albert - **"Digital Computer Electronics"** - McGraw-Hill - p.136-139 - 1977.

[13] KAI, Hwang, FAYÉ, A. Briggs - **"Computer Architecture and Parallel Processing"** - McGraw-Hill - p.170-174 - 1984.

[14] CAPPELLO R. Peter, STEIGLITZ, Kenneth - **"A VLSI Layout for a Pipelined Dadda Multiplier"** - ACM Transactions on Computer Systems, Vol. 1, No. 2 - p. 157-174 - May 1983.

[15] RABEY M. Jan - **"A Design Perspective"** - Prentice Hall Electronics and VLSI Series - p. 240-242 - 1996.

[16] Sánchez Edwin, Caldeira L. - **"Optimization of Cheng's Cell and its Application to a Serial Multiplier Based on the Booth's Modified Algorithm"**- SBMicro Caxambú Agosto 1997.

[17] Sánchez Edwin, Caldeira L. - **"Optimization of Cheng's Cell and its Application to a 125 MHz Serial Multiplier Based on the Booth's Modified Algorithm"**- Primer Workshop de Microelectrónica Rosario - Argentina Outubro 1997.

Bibliografia Recomendada

BOOTH, A. D. - **"A Signed Binary Multiplication Technique"** - J. Mech. Appl. Math. 4 - p.236-240 - 1951.

GLASSER, A. Lance, DOBBERPUHL, W. Daniel - **"The Design And Analysis of VLSI Circuits"** Addison-Wesley - p.52-55 - 1985.

JACKSON, B. Leland, KAISER, F. James, McDONALD, S. Henry **"An Approach To The Implementation Of Digital Filters"** - IEEE Transactions on Audio and Electroacoustics, v.AU-16, n.3 - p.413-421 - set. 1968.

WARE, A. Frederick et al. - **"64 Bit Monolithic Floating Point Processors"** IEEE Journal Of Solid-State Circuits, v.SC-17, n.5 - p.898-907 - out. 1982.

GARNER, L. Harvey - **"The Residue Number System"** - IRE Transactions On Electronic Computers - p.140-147 - jun. 1959.

KUTSUWA, Toshiro, MUN, Minho, EBATA, Katsuhiko - **"Configuration And Evaluation Of 2's Complement Multiplication-Division Arrays"**
IEEE Transactions on Circuits And Systems, v.CAS-34, n.3 - p.304-308 - mar. 1987.

DONTHI, V. Ravindra, SINGH, Harpreet - **"Bit Sequential Multiplier Using Carry-look-ahead Technique"** - Int. J. Electronics, v.55, n.3 - p.411-416 - 1983.

HENLIN, A. Dennis et al. - **"A 16 Bit x 16 Bit Pipelined Multiplier Macrocell"**
IEEE Journal Of Solid-State Circuits, v.SC-20, n.2 - p.542-547 - Abril 1985.

HATAMIAN, Mehdi, CASH, L. Glenn - **"A 70-Mhz 8-bit x 8-bit Parallel Pipelined Multiplier in 2.5-μm CMOS"** - IEEE Journal Of Solid-State Ciecuits, v.SC-21, n.4, p.505-513 - ago. 1986.

LEE Y. Joseph - **"A High-Speed High-Density Silicon 8x8-bit Parallel Multiplier"** - IEEE Journal Of Solid-State Ciecuits, v.SC-22, n.1 - p.35-40 - February 1987.

MATTEME L. et al. - **"A C-Testable Booth Multiplier, Designed for a Silicon Compilation Enviroment"** - CH2473-7/87/0000/0354S01.00 IEEE - P.354-357 - 1987.

EFEI / BIBLIOTECA

ESTE LIVRO DEVE SER DEVOLVIDO NA ÚLTIMA DATA CARIMBADA.

09.12.98	26/09/2003	
10/07/99	10/12/2003	
02/07/09		
07/08/2001		
22/08/2002		
05/09/01		
13/12/01		
12/07/02		
23/07/02		
19.02.02		
06/03/2002		
21/03/2002		
09/04/02		
22 4 2002		
08.05.02		
23.05.03		
02.6.2003		
12.9.2003		
19.9.2003		