RELATÓRIO TÉCNICO

UNIDADE LÓGICA E ARITMÉTICA RÁPIDA DE 32 BITS

Mário Afonso da Silveira Barbosa
Alexandre Ribeiro Sales
NCE/UFRJ

NCE — 03/93
março

NCE

Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro

Tel.: 598-3212 - Fax.: (021) 270-8554
Caixa Postal 2324 - CEP 20001-970
Rio de Janeiro - RJ

UNIDADE LÓGICA E ARITMÉTICA RÁPIDA DE 32 bits

RESUMO

Este trabalho descreve o estudo e a implementação em LSI de duas Unidades Lógicas e Aritméticas de 32bits. O objetivo principal é a avaliação de desempenho de ALUs e a escolha da que será utilizada no microprocessador RISC com arquitetura SPARC, em desenvolvimento no NCE/UFRJ no âmbito do projeto MULTIPLUS. O circuito, que será fabricado no PMU CMOS7, foi implementado com tecnologia CMOS 1.2 micra, regras da ES2. No seu projeto foram utilizados como ferramenta de edição de layout "full custom" e simulação os sistemas MAGIC V6, IRSIM e SPICE, instalados numa estação de trabalho SUN IV.

FAST 32 BIT ARITHMETIC AND LOGIC UNIT

ABSTRACT

This paper describes the analisys and LSI implementation of two 32-bit Arithmetic and Logic Unities. The main goal of this work is to evaluate the performance of the ALU; the one will be used as part of a RISC microprocessor based on the SPARC architecture, under development at NCE\UFRJ within MULTIPLUS project. The circuit is being implemented in the Brazilian Multiproject using a 1,2μm CMOS ES2 technology. The full custom ALU layout has been designed using the Berkeley MAGIC V6 system and simulated using the IRSIM and SPICE tools running on a SUN IV workstation.

UNIDADE LÓGICA E ARITMÉTICA RÁPIDA DE 32 bits

Mario Afonso da Silveira Barbosa [1]

Alexandre Ribeiro Sales

Universidade Federal do Rio de Janeiro - Núcleo de Computação Eletrônica

Caixa Postal 2324 CEP: 20001 Rio de Janeiro - RJ

RESUMO

Este trabalho descreve o estudo e a implementação em LSI de duas Unidades Lógicas e Aritméticas de 32bits. O objetivo principal é a avaliação de desempenho de ALUs e a escolha da que será utilizada no microprocessador RISC com arquitetura SPARC, em desenvolvimento no NCE/UFRJ no âmbito do projeto MULTIPLUS. O circuito, que será fabricado no PMU CMOS7, foi implementado com tecnologia CMOS 1.2 micra, regras da ES2. No seu projeto foram utilizados como ferramenta de edição de layout "full custom" e simulação os sistemas MAGIC V6, IRSIM e SPICE, instalados numa estação de trabalho SUN IV.

ABSTRACT

This paper describes the analisys and LSI implementation of two 32-bit Arithmetic and Logic Unities. The main goal of this work is to evaluate the performance of the ALU; the one will be used as part of a RISC microprocessor based on the SPARC architecture, under development at NCE\UFRJ within MULTIPLUS project. The circuit is being implemented in the Brazilian Multiproject using a 1,2μm CMOS ES2 technology. The full custom ALU layout has been designed using the Berkeley MAGIC V6 system and simulated using the IRSIM and SPICE tools running on a SUN IV workstation.

---

1 - Engenheiro Eletrônico (UFRJ 1982), M.Sc. em Sistemas e Computação (COPPE /UFRJ 1987).

2 - Engenheiro Eletrônico (CEFET "CSF"-RJ 1992)

1 - INTRODUÇÃO:

O MULTIPLUS é um multiprocessador científico de alto desempenho em desenvolvimento no NCE/UFRJ [AUDE90]. Sua arquitetura é modular e capaz de suportar até 256 nós de processamento. Cada nó de processamento possui um microprocessador RISC [PATT85]de 32 bits com arquitetura SPARC [SUN87].

No âmbito do projeto MULTIPLUS está sendo desenvolvido o circuito integrado de um microprocessador RISC compatível com a arquitetura SPARC [BARB90]. Este CI será utilizado nos nós de processamento, em substituição ao CI comercial. Os diversos blocos que compõem este chip estão sendo fabricados através do PMU brasileiro. Este artigo descreve o desenvolvimento da ALU.

Este microprocessador terá um clock de 20MHz (T=50ns) dividido em quatro fases não sobrepostas de 12.5ns cada [BARB91]. Sua Unidade Lógica e Aritmética disporá de um tempo um pouco maior que uma fase de relógio para executar uma operação. Por segurança consideramos o tempo de operação da ALU 12.5ns.

Em uma ALU, o ponto crítico na obtenção de um alto desempenho é o atraso no caminho do CARRY do seu somador, o que descarta a utilização de lógicas tradicionais como a de "RIPPLE-CARRY" [WEST85] e nos faz buscar lógicas mais aprimoradas que nos permitam construir um somador rápido e ao mesmo tempo pequeno.

Após um estudo preliminar, foram selecionados três tipos de ALUs que atenderiam aos pré-requisitos. Destas apenas duas foram implementadas, cada uma utilizando uma lógica diferente no circuito de soma. A ALU do tipo "CONDICIONAL SUM" [ROTH89]é descrita no ítem 2. No ítem 3 é descrita ALU que utiliza lógica "BINARY LOOKAHEAD CARRY" [WEI90]. No ítem 4 é mostrada a implementação "full custom" de um chip contendo as duas ALUs para comparação e no ítem 5 são apresentados resultados e conclusões.

No projeto deste circuito foram utilizadas as ferramentas MAGIC de Berkeley, para edição de layout, e as ferramentas IRSIM e SPICE 3 modelo 2 para as simulação lógica/elétrica respectivamente. Todas estas ferramentas de trabalho estão instaladas numa estação de trabalho SUN IV.

2- ALU baseada em "BINARY LOOKAHEAD CARRY"

2.1- Descrição

O ponto crítico para a obtenção de uma ALU de bom desempenho está no tempo de geração do "carry" no seu somador, que cresce linearmente com o tamanho dos operandos. Uma das técnicas mais utilizadas na otimização deste tempo é a de Carry Lookahead, que consiste no cálculo em paralelo dos sinais de "carry". O carry no i-ésimo estágio pode ser expresso por:

$$C_i = g_i + p_i.C_{i-1} \quad (1)$$

onde: $g_i = A_i.B_i$ (sinal gerado) (2)

$p_i = A_i \text{ XOR } B_i$ (sinal propagado) (3)

No entanto, esta técnica só se mostra eficiente para operandos de até aproximadamente 4 bits, pois o número de bits do somador define o número de entradas das portas usadas na implementação da equação (1).

Foi definido então o operador de concatenação "o":

$$(g,p) \text{ o } (g',p') = (g + (p.g'), p.p') \quad (4)$$

Temos então:

$$C_i = G_i \quad (5)$$

onde

$$(G_i,P_i) = (g_o + p_o.C_{in}), p_o \quad \text{para } i = 0 \quad (6)$$

$$(g_i,p_i) \text{ o } (G_{i-1},P_{i-1}) \quad \text{para } i = 1 \text{ até } n$$

$$S_i = p_i \text{ XOR } C_{i-1} \quad \text{para } i = 1 \text{ até } n \quad (7)$$

$$p_o \text{ XOR } C_{in} \quad \text{para } i = 0$$

Dado o fato de que o operador "o" é associativo podemos escolher um número "m" entre 0 e i tal que a eq.(6) possa ser reescrita da seguinte maneira:

$$(G_{i,0},P_{i,0}) = (G_{i,m},P_{i,m}) \text{ o } (G_{m-i,0},P_{m-i,0}) \quad (8)$$

2.2- Implementação

Na figura 1 temos o diagrama em blocos da ALU. O circuito de precondição, a partir das entradas Ai e Bi, gera os sinais pi e gi bem como as funções lógicas ANDi, ORi e XORi. Na figura 2 temos o diagrama lógico e o layout deste circuito.

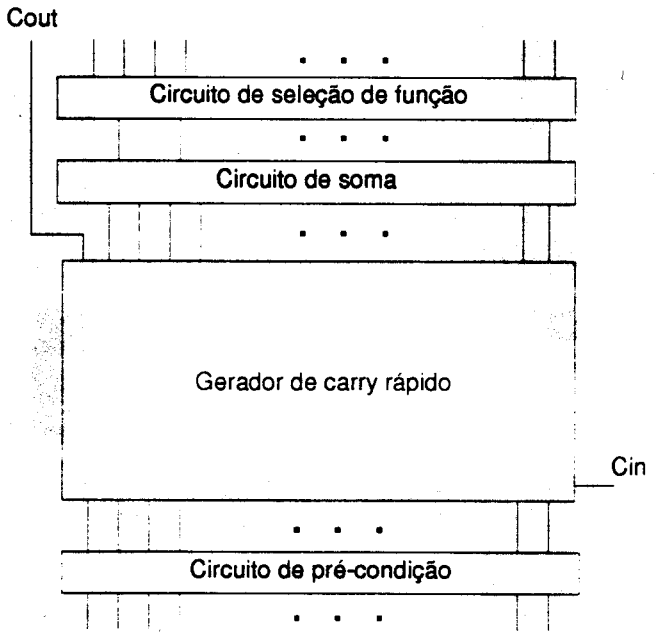
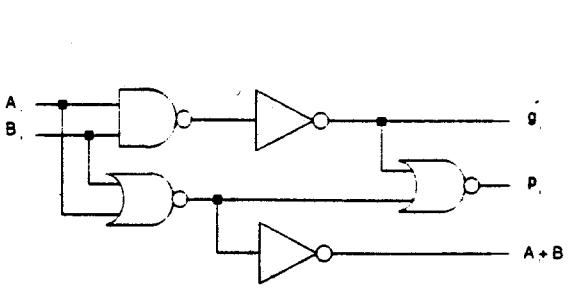
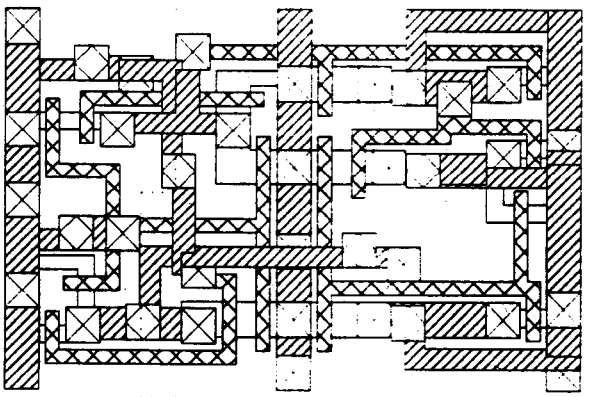


Fig. 1 - Diagrama em blocos da ALU





(a) Diagrama lógico

(b) Layout

Fig .2 - Circuito de pré-condição

O circuito de soma corresponde ao ou-exclusivo dos sinais p_i e C_{i-1}. O seu layout é mostrado na figura 3.

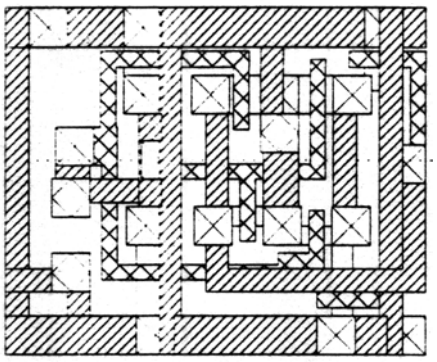


Fig.3 - Layout do circuito de soma

O circuito de seleção de função é um multiplexador que seleciona uma das funções AND, OR, XOR e SOMA na saída da ALU. A subtração é feita invertendo-se um dos operandos, atribuindo-se "1" a Cin e somando-se os operandos.

O circuito de geração de "carry" realiza as operações definidas em (4) e (6). Neste circuito foram utilizados três tipos básicos de célula: células pretas e brancas [BREN82] e "drivers". As células pretas realizam o operador "o" e são de dois tipos: células ba e bb. As células ba recebem entradas positivas e geram saídas negativas; as células bb recebenm entradas negativas e geram saídas positivas. Na figura 4 temos a célula ba e na figura 5 a célula bb.

As células do tipo bb recebem como entrada sinais invertidos enquanto que as do tipo bageram sinais invertidos. Alem disso, as células pretas recebem como entradas sinais gerados por outras células pretas. Para garantirmos a polaridade correta dos sinais nesessitamos de inversores que são chamados de células brancas wa. Alem disso, algumas vezes é necessário também propagar os sinais para os bits posteriores; estas células são chamadas de células brancas wb. Na figura 6 são mostradas as células brancas.

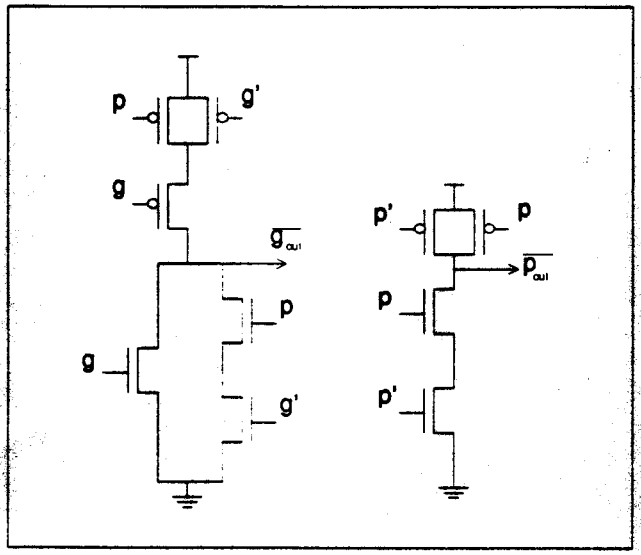
No caso da saída de uma célula preta se propagar para muitas outras células e torna-se decessário o uso de "drivers" de um ou mais estágios para diminuir o atraso do sinal.

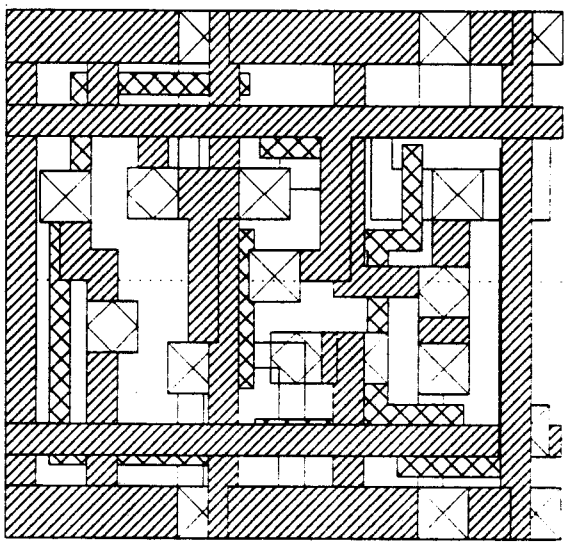
A figura 7 mostra como exemplo um gerador de carry de 6 bits.

A partir de uma análise de atraso para os três tipos de células (pretas, brancas e drivers) [WEI90], pode-se posicionar e interconectar essas células de forma a se obter o gerador de "carry" ótimo.

Com base na eq.(8), um gerador de "carry" de n bits pode ser dividido em dois outros (fig.8). Por sua vez, cada um desses blocos pode ser dividido em dois outros e assim susessivamente em uma construção recursiva.

É importante notar que em geral $R_{(n-m)}$ é mais alto do que $R_{(m)}$. Isto permite que se coloque drivers na saída do bit mais significativo de $R_{(m)}$ sem aumentar o tamanho do circuito.

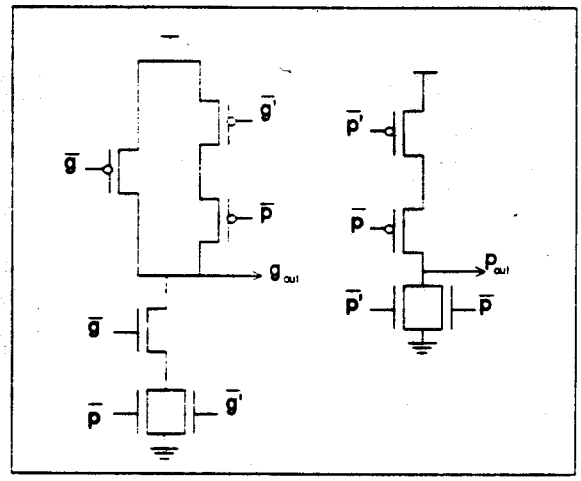


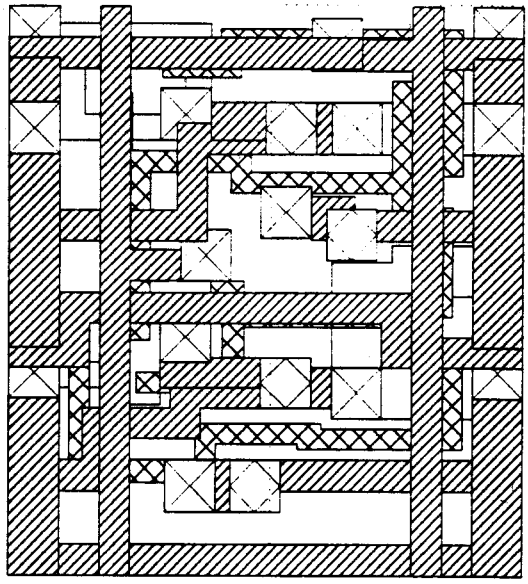


(a) Diagrama de transistores

(b) Layout

Fig.4 - Célula ba





(a) Diagrama de transistores

(b) Layout

Fig.5 - Célula bb

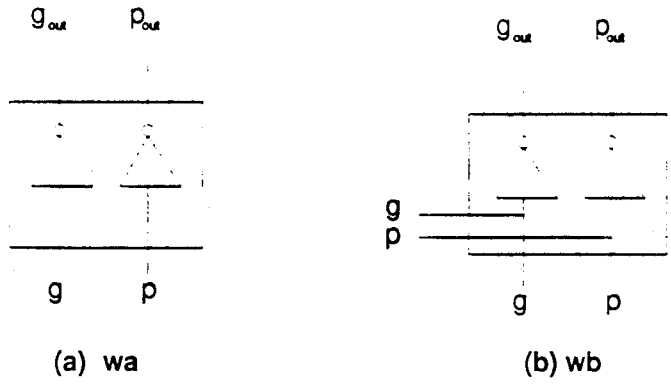


Fig.6 - Células brancas

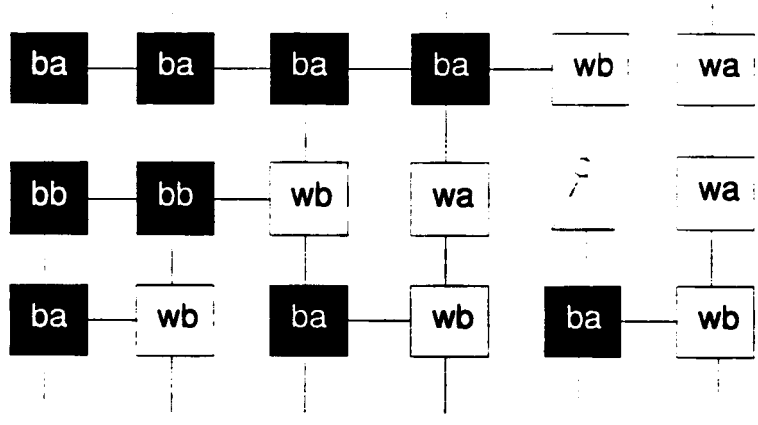


Fig.7 - Gerador rápido de carry de 6 bits

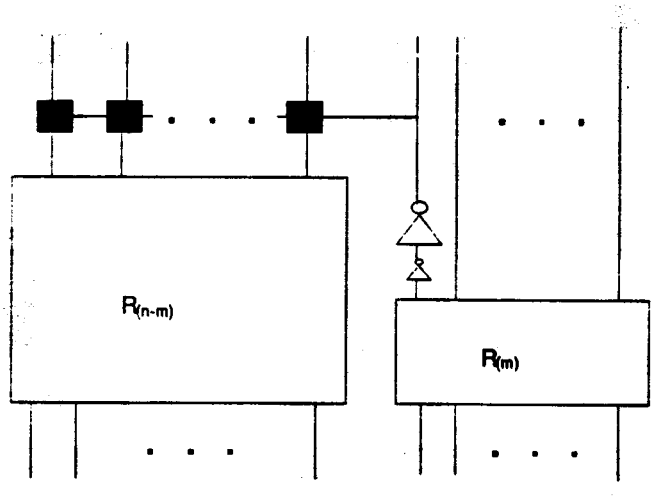


Fig.8 - Contrução recursiva de um gerador rápido de carry

Podemos observar que existem dois possíveis caminhos críticos em R(n). Um é a propagação do bit mais significativo (bit n), o outro é o atraso do bit m que driva todos os bits de R(n-m). É importante notar que ambos os caminhos levam à célula que está no extremo superior esquerdo do circuito. Portanto, se minimizarmos o atraso das entradas desta célula (determinando o valor ótimo para "m"), minimizaremos o atraso de todo o circuito. Com base nisto, [WEI90] definiu um método computacional de obtenção do "m" ótimo.

Utilizando este método computacional chegamos à forma final do circuito gerador de 'carry" e à ALU (fig.9).

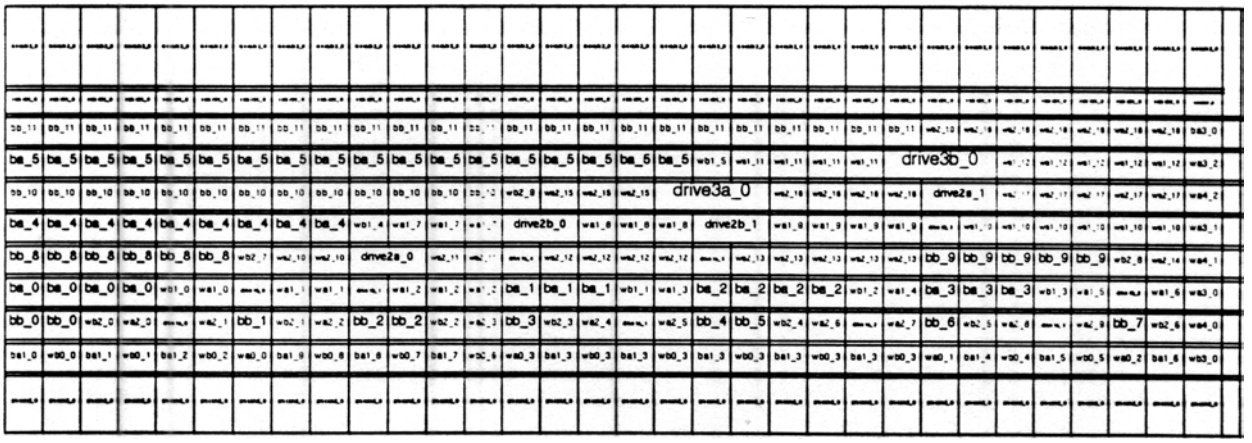


Fig.9 - ALU de 32 bits (Binary Lookahead Carry)

3- CIRCUITO DE SOMA CONDICIONAL[ROTH89]

3.1- Descrição

O circuito de soma condicional (fig.10) apresenta basicamente dois elementos distintos: Células de condição(CC) e Multiplexadores(M#).

A figura 10 mostra o esquema em blocos de um somador de 8-bits baseado em soma condicional.

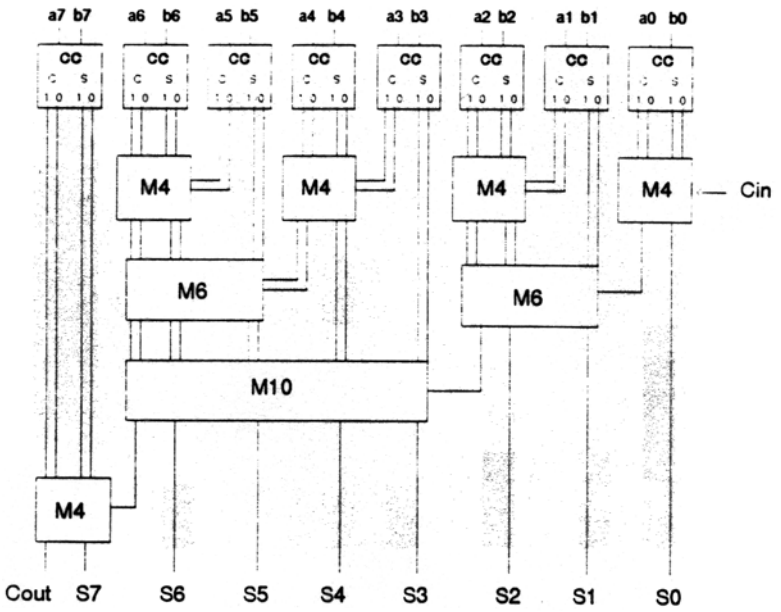


fig.10 - Somador de 8 bits

Para cada bit do somador existe uma célula de condição onde a soma e o "carry" são calculados duas vezes, uma delas considerando o "carry" proveniente do estágio anterior igual a 0 (Gerando então os sinais S_0 e C_0) e a outra considerando "carry" igual a 1 (Gerando os sinais S_1 e C_1).

A seleção das somas e "carries" corretos será feita níveis abaixo pelos multiplexadores que são controlados pelos "carry-out" dos bits anteriores . Portanto, a idéia básica que a lógica de soma condicional sugere é a de fazer a soma imediata e simultânea de todos os bits dos operandos, não se preocupando com os "carries" anteriores. Após a soma "duplicada" ter sido feita, uma lógica de chaves (mux) seleciona o valor final correto da soma.

A tabela da figura 11 mostra a operação de soma condicional do circuito da fig.10 [SKLA60].

i	7	6	5	4	3	2	1	0	carry anterior	intervalo de tempo
xi	0	1	1	0	1	1	0	1		
yi	1	0	1	1	0	1	1	0		
S	1	1	0	1	1	0	1	1	0	T1
C	0	0	1	0	0	1	0	0		
S	0	0	1	0	0	1	0	0	1	
C	1	1	1	1	1	1	1	0		
S	1	0	0	1	1	0	1	1	0	T2
C	0	1		0		1		0		
S	0	0	1	0	0	1	0		1	
C	1	1		1		1				
S	1	0	0	1	1	0	1	1	0	T3
C	0	1				1				
S	0	0	1	0	0				1	
C	1	1								
S	1	0	1	0	0	0	1	1	0	T4
C	0	1								
S	0								1	
C	1									
S	0	0	1	0	0	0	1	1	0	T5
C	1									
S									1	
C										

Fig.11 - Tabela de Soma Condicional

Na fig.11 estão representados por t0, t1, t2, t3 e t4 os intervalos de tempo dentro dos quais somas e "carries" condicionais são computados.

Durante o intervalo de tempo t0 todos os bits são somados duas vezes considerando os dois possíveis valores de "carry-in". Desta forma são gerados durante este intervalo de tempo os sinais S0 e C0 (Soma e "carry" considerando "carry-in" igual a 0) e S1 e C1 (Soma e "carry" para "carry-in" igual a 1) (veja na fig.10). Estes sinais são representados na tabela da fig.11 pelos dois pares soma e "carry" de cada coluna referente ao intervalo t0.

Em t1 são selecionados soma e "carry" para pares de colunas (colunas 1 e 2, 3 e 4, 5 e 6,...), basaeando-se nos dois possíveis valores do carry gerado no bit anterior. Esta seleção portanto é feita simultaneamente em blocos de dois bits(duas colunas). A primeira coluna é selecionada imediatamente pelo "carry-in" inicial do somador.

Continuando o processo, as somas e "carries" condicionais são selecionados simultaneamente em blocos de quatro bits durante o tempo t2, em blocos de oito durante o tempo t3 e assim por diante, dependendo do número de bits do somador. No final do processo as somas e "carries" obtidos já não serão mais condicionais e sim o valor exato da soma entre os operandos. O processo assume a forma de árvore binária podendo ser facilmente ampliado para um número de bits maior.

Apesar desta lógica de soma condiciomal proporcionar uma operação rápida do somador, otimizando o tempo de propagação do "carry", se ela fosse implementada com NAND's, NOR's e inversores, o somador se tornaria muito grande e consequentemente lento. No entanto a lógica fica bem menor e mais rápida se for implementeda com "Transmission gates".Desta forma, toda a lógica do somador foi construido utilizando Transmission Gates, inclusive a célula de condição.

3.2- Implementação

A célula de condição é o único bloco do circuito que realiza a soma dos operandos. Já mencionamos que esta soma é feita duas vezes considerando os dois possíveis valores de "carry-in", o que caracteriza a lógica como sendo condicional. Desta célula também se originam as operações lógicas AND, OR e XOR entre os operandos que são os sinais C0, C1 e S1 respectivamente. A figura 12 mostra o circuito completo da célula de condição utilizada no somador.

A maior parte do circuito de soma condicional é formada por blocos de multiplexadores que têm a responsabilidade de selecionar as somas e os "carries" do circuito durante o processo de soma. Já vimos que apenas na primeira etapa da lógica de soma condicional é que se opera a soma propriamente dita dos operandos. A partir dai a lógica se baseia totalmente em multiplexadores que irão selecionar, ao seu tempo, as somas e os "carries" corretos.Além destes multiplexadores existem os de seleção da operação na saída da ALU, que são idênticos ao do ítem 2.2.

De acordo com o funcionamento da lógica de Soma Condicional, sabe-se que em cada nível da soma são liberados dois "carries" que irão entrar em um multiplexador no nível superior de lógica e selecionar as somas e "carries" deste nível (de acordo com a tabela da fig 11). Como os "carries" que entram em um multiplexadores tem que controlar muitos transistores em série, (o número de transistores cresce de acordo com o nível no processo de soma) e sendo eles já provenientes de uma chave, tornou-se necessária a colocação de "buffers" antes de cada multiplexador com a função de restaurar o nível do sinal e ainda acionar a carga inerente aos multiplexadores. A figura 13 mostra os resultados obtidos no dimencionamento dos "buffers".

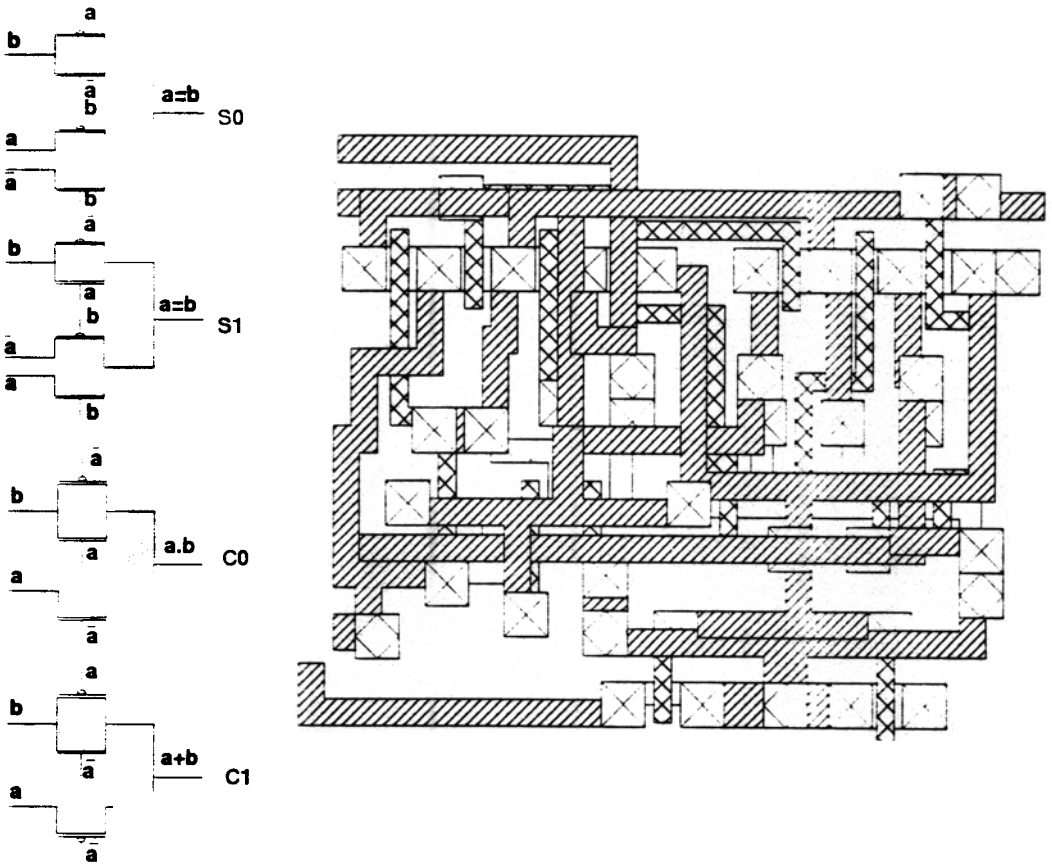


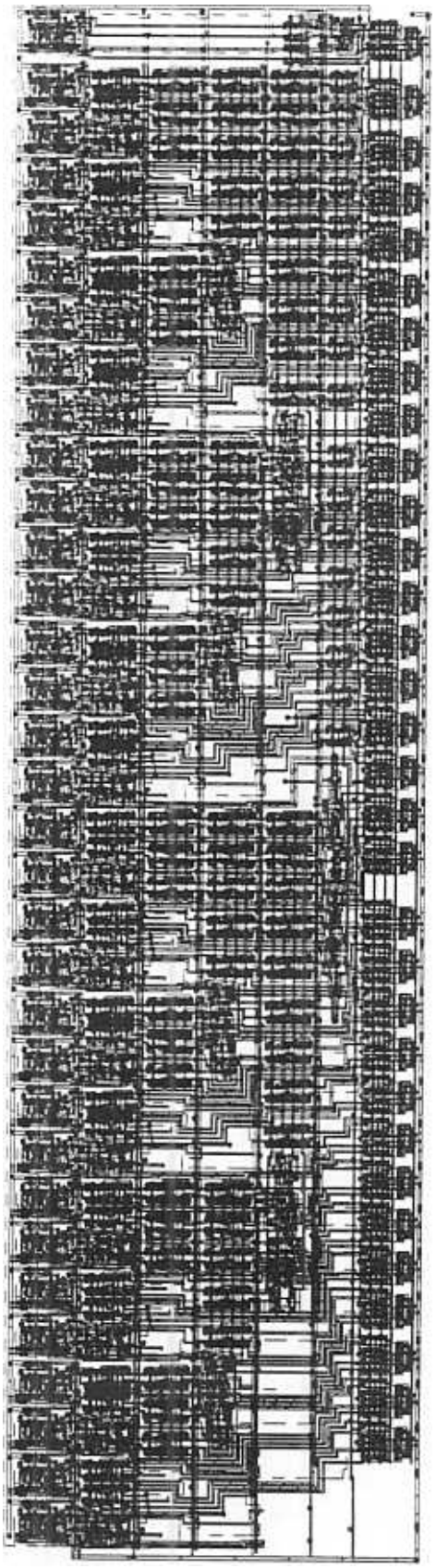
Fig . 12 - Célula de condição

no. do Buffer	bits de entrada do somador correspondente	no. de transisto_ res para serem drivados	no. de estagios do Buffer
1	2	2	1
2	4	3	2
3	8	5	2
4	16	9	3
5	32	17	4

<table>
<tr><td colspan="4">B3</td><td>MX</td><td>MX</td><td>MX</td><td>MX</td></tr>
<tr><td colspan="2">B2</td><td>MX</td><td>MX</td><td colspan="2">B2</td><td>MX</td><td>MX</td></tr>
<tr><td>B1</td><td>MX</td><td>B1</td><td>MX</td><td>B1</td><td>MX</td><td>B1</td><td>MX</td></tr>
<tr><td>CC</td><td>CC</td><td>CC</td><td>CC</td><td>CC</td><td>CC</td><td>CC</td><td>CC</td></tr>
</table>

fig. 13 - Dimencionamento dos "Buffers".

Na figura abaixo temos o layout da ALU de Soma Condicional.



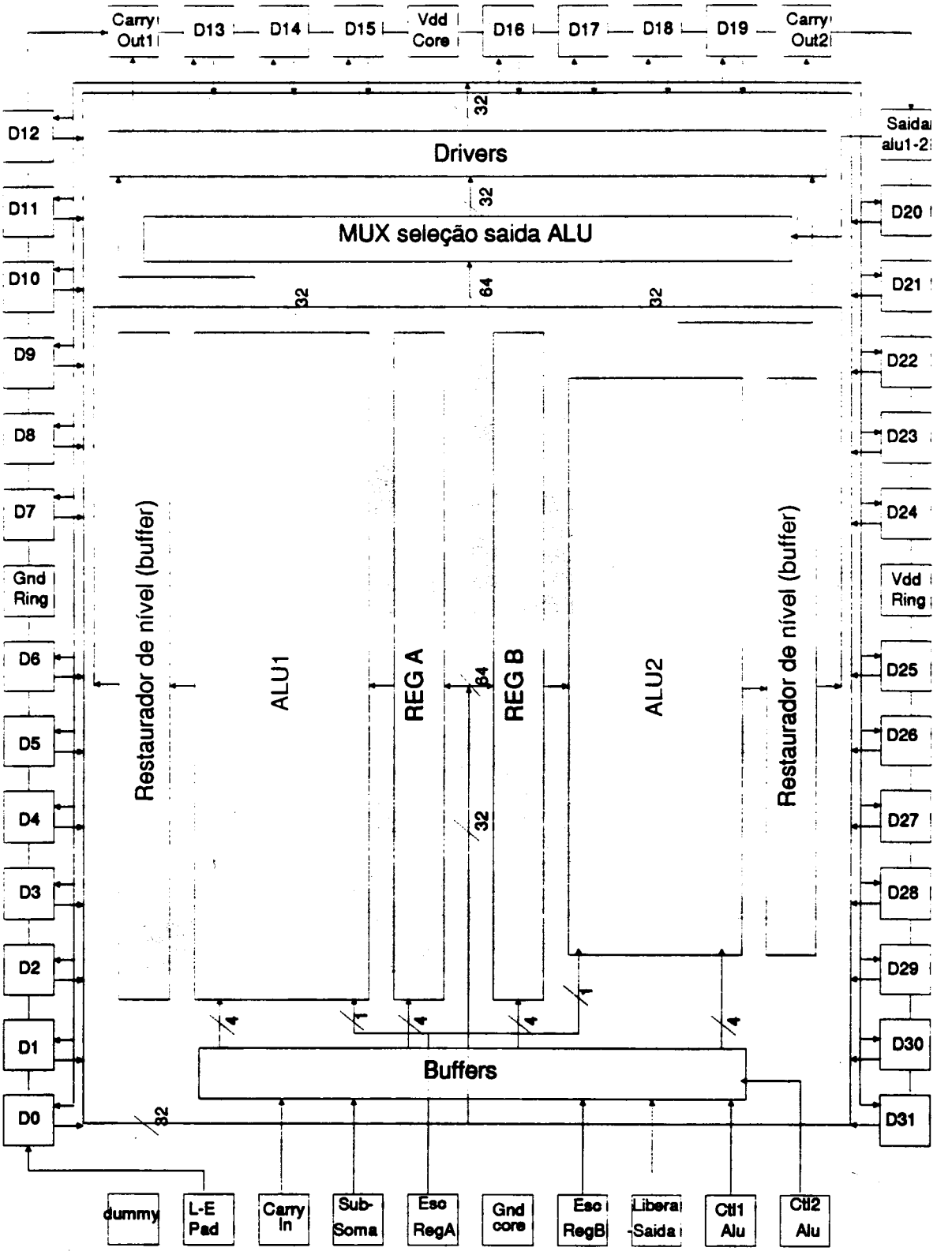
4 - Implementação do chip

O principal objetivo deste projeto foi a fabricação de um chip de teste com as duas ALUs para que se pudesse verificar o funcionamento de cada uma e finalmente selecionarmos a melhor para ser usada no processador RISC em desenvolvimento no NCE.

Na montagem final deste chip tivemos um problema: o número máximo de pinos por pastilha no PMU Brasileiro é 48. Cada uma das ALUs tem 64 entradas e 32 saídas, além dos controles. A solução encontrada foi utilizar PADs de Entrada\Saída para os dados, registradores para armazenagem dos operandos e multiplexadores de seleção de saída das ALUs. Com isto, o chip de teste apresentou um total de 47 PADs, sendo 32 de I/O para entrada de operandos e saída dos resultados das operações, 2 de saída de "Carry" das ALUs, 4 de alimentação (GND e Vdd), 1 de controle do MUX de seleção da ALU (saída alu1/2), 1 de controle dos PADs de I/O (L/E Pad), 2 de carga dos operandos (Esc RegA, Esc RegB), 1 de seleção entre soma e subtração (sub/soma), 2 de seleção das operações das ALUs (Ctl1 alu, Ctl2 alu),1 de entrada de "carry" (Carry in), e 1 que libera um dos operandos para que possamos verificar o atraso das ALUs sem que esteja incluido o tempo de carga nos registradores.

A operação de subtração é feita invertendo-se um dos operandos (através do sinal sub\soma), setando-se o sinal de "carry-in" e realizando uma operação de soma.

Na figura 14 temos o diagrama em blocos e na figura 15 o layout final do chip. O bloco ALU1 corresponde à ALU baseada em Soma Condicional e o bloco ALU2 corresponde à ALU baseada em "Binary Lookahead Carry".



CHIP PMU CMOS7

Fig.14 - Planta baixa do chip

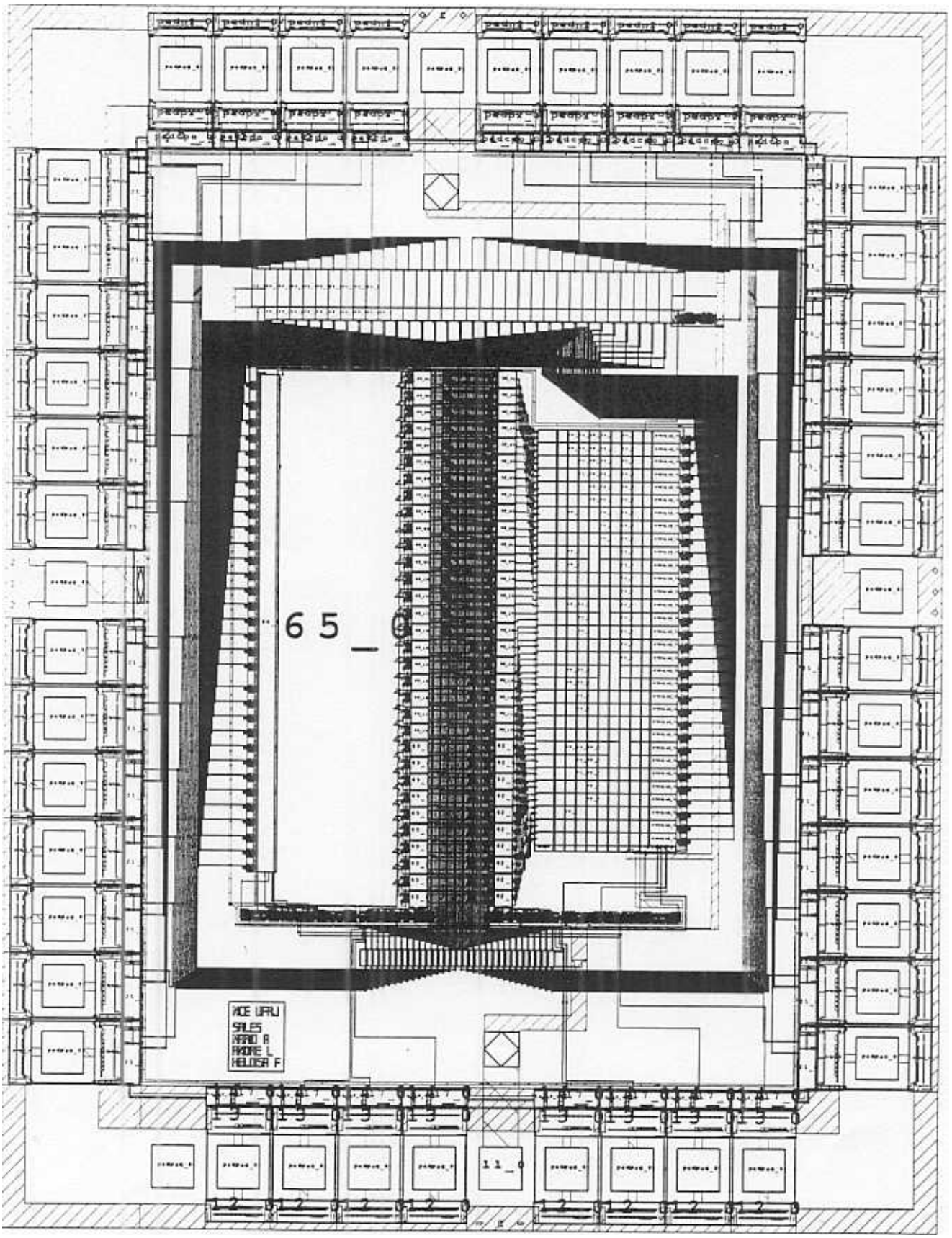


Fig.15 - Layout completo do chip

5 - CONCLUSÃO

O chip ocupou uma área total de 3110 X 2430 micra, incluindo os 48 PADs, na tecnologia ES2 CMOS 1.2 micra. O funcionamento lógico deste circuito foi totalmente verificado no simulador lógico IRSIM do MAGIC (fig.16). Este projeto será implementado no próximo PMU Brasileiro (PMU CMOS7).

A ALU1 é composta de 2128 transistores, ocupando uma área de 1363 X 369 micra. O tempo de atraso do sinal de "carry out", simulado no SPICE 3 modelo 2, foi de 7.3ns no pior caso (fig.17). A ALU2, com 2446 transistores, ocupou uma área de 1072 X 364 micra e apresentou um tempo de atraso de 4.5 ns no pior caso (fig.18).

Através dos resultados acima pudemos concluir que, pelo menos em termos de simulação, ambas as ALUs são rápidas, atendendo com folga às restrições de velocidade (atraso máximo de 12.5ns).

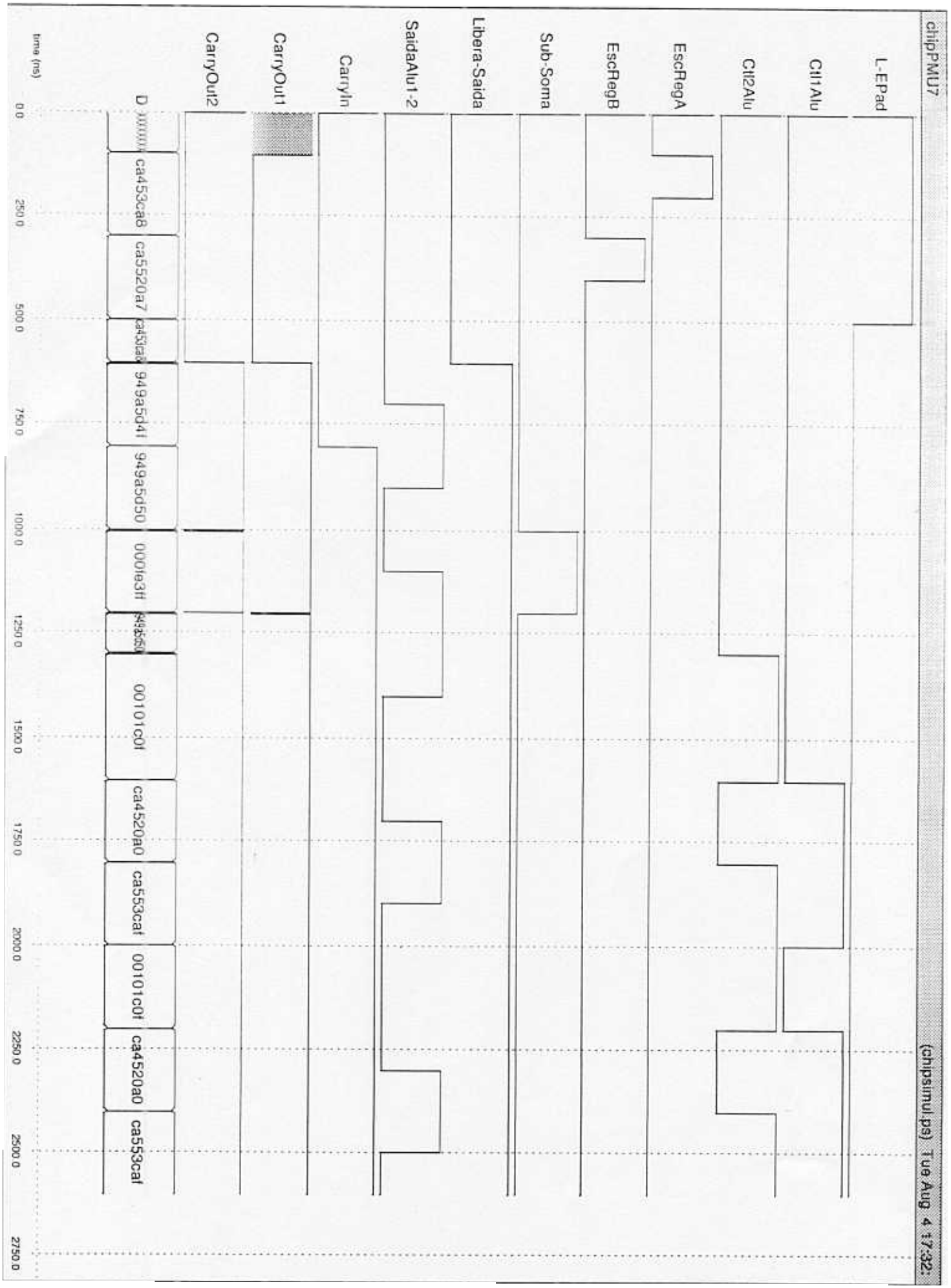


Fig.16 - Resultado da simulação lógica do chip

Fig.17 Resultado da simulação elétrica da ALU1

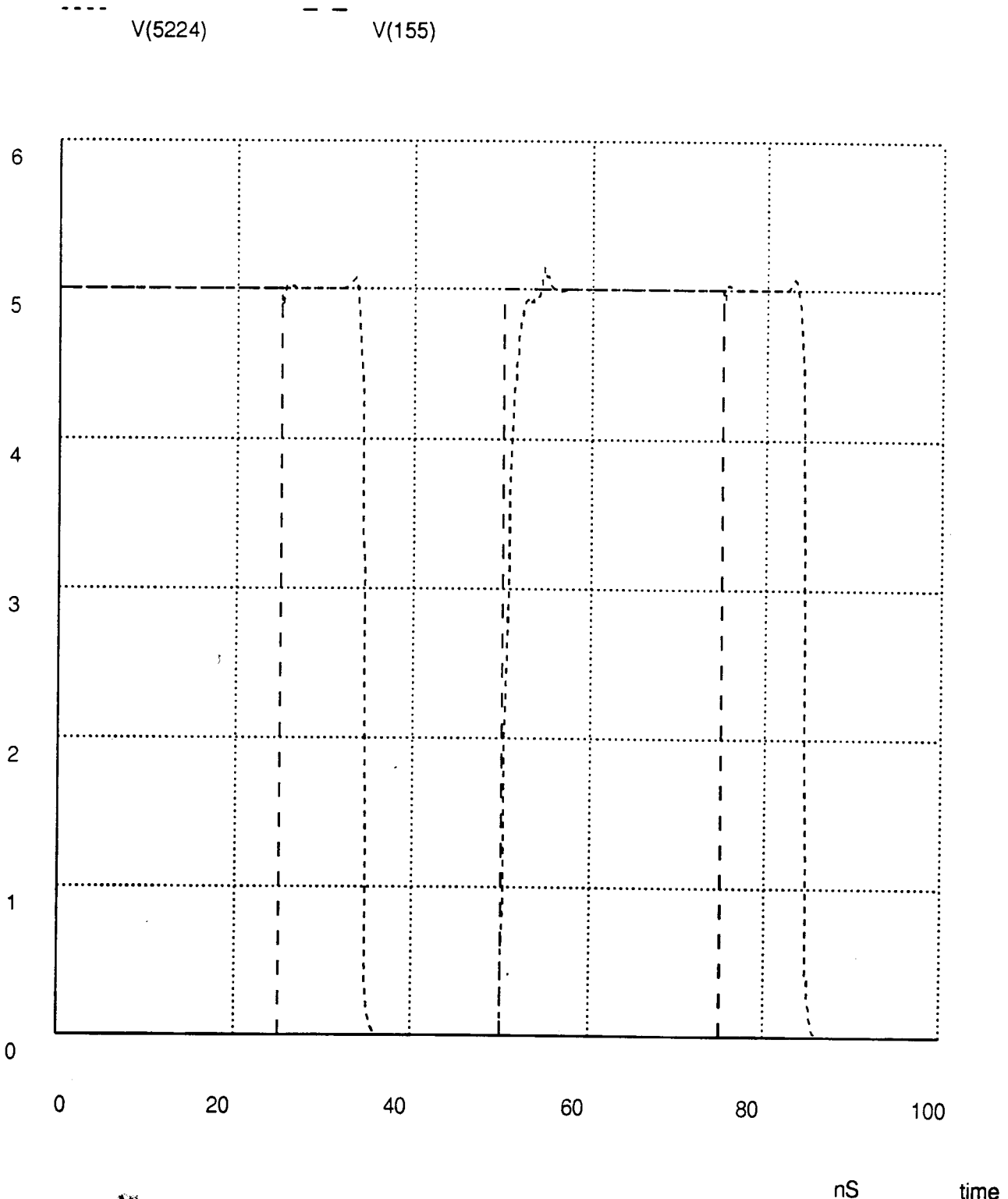
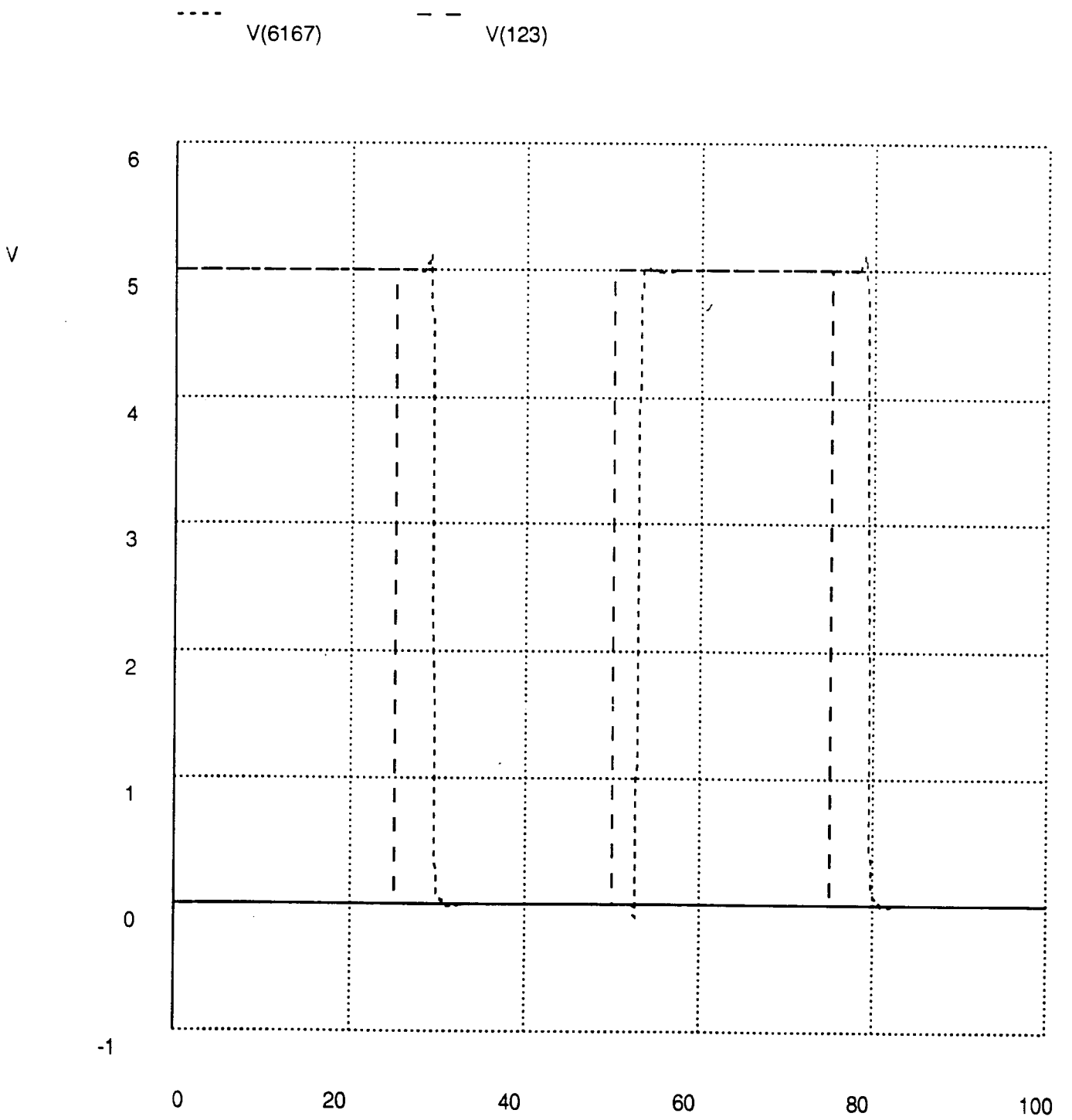


Fig.18 - Resultado da simulação elétrica da ALU2



VI - REFERÊNCIAS

[BARB91] - "Gerador de Clock de Quatro Fases Para Microprocessadores". Barbosa, M. A. S. & Façanha, H. S., Anais do VI Congresso da SBMicro, pp. 325-335, Julho 1991.

[BARB90] - "Implementação de Microprocessador RISC com Arquitetura SPARC". Barbosa, M. A. S., et al., Anais do V SBCCI, Outubro 1990, pp. 121 - 131.

[BREN82] - "A Regular Layout for Parallel Adders", Brent, R. P. and Kung,H. T., IEEE Transactions on computers, vol. C31, no. 3, pp. 260 - 264.Março 1982.

[AUDE90] - "Multiplus: Um Multiprocessador de Alto Desempenho". Aude, J. S. et al., Anais do X Congresso da SBC, Julho de 1990, pp. 93 - 105.

[PATT85] - "Reduced Instruction Set Cumputers", Patterson, D., Comunications of the ACM, Janeiro 1985, pp. 8-21.

[SUN87] - "The SPARC Arquitecture Manual", Sun Microsystems Inc., Outubro 1987.

[SKLA60] - "Conditional-Sum Addition Logic.", J. Sklansky, IRE Trans. Electron. Computers, vol.EC-9, pp. 226-231, Junho 1960.

[WEI90] - "Area-Time Optimal Adder Design", Belle W. Y. Wei & C. D. Thompson, IEEE Transations on Computers, vol.39, no.5, pp 666-675, Maio 1990.

[ROTH89] - "Realization of Transmission-Gate, Conditional-Sum (TGCS) Adders with Low Latency Time", Rothermel,A. et al., IEEE Journal of Solid-State Circuits, vol.24, no.3, pp. 558-561, Junho 1989.

[WEST85] - "Principles of CMOS VLSI Design: A Systems Perspective", N. West & K. Eshraghian, Addition-Wesley Publ. Co., Massachusetts, 1985.

[DOUG88] - "Basic VLSI Design Systems and Circuits", Douglas A. Pucknell & K. Eshraghian, Prentice Hall Silicon Systems Engineering Series Publ., second edition, 1988.