DISSERTAÇÃO DE MESTRADO

ROTEADOR NANOELETRÔNICO PARA REDES-EM-CHIP BASEADO EM TRANSISTORES MONOELÉTRON

Beatriz Oliveira Câmara da Fé

Brasília, março de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

FICHA CATALOGRÁFICA

CÂMARA, BEATRIZ OLIVEIRA
ROTEADOR NANOELETRÔNICO PARA REDES-EM-CHIP BASEADO EM TRANSISTORES MONOELÉTRON [Distrito Federal] 2017.
xvi, 75 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2017).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Nanoeletrônica	2. SET
3. NoC	4. Roteador
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CÂMARA, B. O. (2017). *ROTEADOR NANOELETRÔNICO PARA REDES-EM-CHIP BASEADO EM TRANSISTORES MONOELÉTRON*. Dissertação de Mestrado, Publicação 656/2017 DM/PGEA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 75 p.

CESSÃO DE DIREITOS

AUTOR: Beatriz Oliveira Câmara da Fé
TÍTULO: ROTEADOR NANOELETRÔNICO PARA REDES-EM-CHIP BASEADO EM TRANSISTORES MONOELÉTRON.
GRAU: Mestre em Engenharia de Sistemas Eletrônicos e Automação ANO: 2017

Beatriz Oliveira Câmara da Fé
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ROTEADOR NANOELETRÔNICO PARA REDES – EM – CHIP BASEADO EM TRANSISTORES MONOELÉTRON

BEATRIZ OLIVEIRA CÂMARA DA FÉ

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

JANAÍNA GONÇALVES GUIMARÃES, Dra., ENE/UNB
(ORIENTADORA)

MARCELO MENEZES DE CARVALHO, Dr., ENE/UNB
(EXAMINADOR INTERNO)

OMAR PARANAÍBA VILELA NETO, Dr., UFMG
(EXAMINADOR EXTERNO)

Brasília, 08 de março de 2017.

Dedicatória

À minha orientadora,
prof. Janaína Guimarães,
meu guia em terras inexploradas.

Agradecimentos

À minha mãe Áurea, por seu suporte e cuidados diários, sem os quais não teria sido possível realizar mais esta etapa.

Ao meu irmão Pedro, meu constante incentivador na busca de mais conhecimento.

À minha avó Maria de Jesus, cuja sabedoria adquirida em seus 90 anos de vida me inspira diariamente.

Aos meus tios Rubens e Aurenice, que são como pais para mim, por seu apoio e presença na minha vida.

À minha orientadora prof. Janaína Guimarães, pessoa e profissional exemplar, por ter possibilitado essa oportunidade e ter verdadeiramente me orientado nesta jornada.

Ao meu co-orientador, prof. José Camargo, por ter compartilhado comigo seu vasto conhecimento e por seu apoio durante esta etapa.

RESUMO

A contínua miniaturização do tamanho dos transistores abriu espaço para inovações tecnológicas e novas abordagens de desenvolvimento de sistemas. Dentre estas inovações pode-se destacar a tecnologia nanoeletrônica e os sistemas-em-*chip* (SoC). Os SoCs são limitados pelas suas interconexões e a abordagem de redes-em-*chip* (NoC) provê uma solução flexível e expansível para esse problema. O roteador é o módulo central na NoC e novas arquiteturas estão sendo desenvolvidas para melhor atender as necessidades de um SoC, que incluem baixo consumo de potência e menor área ocupada possível. Por sua vez o transistor monoelétron (SET) é um dispositivo nanoeletrônico que ocupa uma pequena área e dissipa pouca potência, sendo ideal para o desenvolvimento de um roteador nanoeletrônico.

Este trabalho propõe uma arquitetura digital de um roteador para NoC com topologia *Mesh* completamente baseado na tecnologia SET. São propostos módulos digitais básicos baseados na tecnologia SET, compilados em uma biblioteca para LTspice, e novas arquiteturas de uma memória SRAM e um registrador FIFO. Ao final os resultados serão comparados com a tecnologia CMOS, evidenciando as vantagens do roteador nanoeletrônico.

ABSTRACT

The continued reduction in transistor size has made room for technological innovations and new approaches to system development. Among these innovations the nanoelectronic technology and systems-on-chip (SoC) can be highlighted. SoCs are limited by their interconnections, and the network-on-chip (NoC) approach provides a flexible and scalable solution to this problem. The router is the central module in NoC and new architectures are being developed to better meet the needs of a SoC, which include low power consumption and the smallest possible occupied area. In turn, the single-electron transistor (SET) is a nanoelectronic device that occupies a small area and dissipates low power, being ideal for the development of a nanoelectronic router.

This work proposes a complete nanoelectronic circuit for an information router aiming at NoCs with Mesh topology. Basic digital modules based on the SET technology and new architectures of an SRAM memory and a FIFO register are proposed. At the end the results will be compared with the CMOS technology and the advantages of the nanoelectronic router will become evident.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

a	Espessura da barreira
A	Área das placas do capacitor
C	Capacitância
d	Distância entre as placas do capacitor
E	Energia total da partícula
E_F	Energia de Fermi
E_{vac}	Energia potencial do vácuo
h	Constante de Planck
I	Corrente
L_x, L_y, L_z	Dimensões 3D de um espaço confinado
$m*$	Massa efetiva
m_e	Massa do elétron
n_x, n_y, n_z	Números quânticos
Q	Carga
q_e	Carga do elétron
T	Probabilidade de tunelamento
V	Tensão
V_0	Energia constante de uma barreira de potencial
x	Posição

Símbolos Gregos

ε	Constante dielétrica
λ_F	Comprimento de onda de Fermi
Ψ	Função de onda

Outros

$e\Phi$	Função trabalho
$\hbar$	Constante de Dirac

Sobrescritos

$-$	Operação NOT

Siglas

CMOS	*Complementary Metal-Oxide-Semiconductor*
CNTFET	*Carbon Nanotube Field-Effect Transistor*
DEMUX	*Demultiplexer*
DRAM	*Dynamic Random-Access Memory*
EB	*Ellastic Buffer*
FIFO	*First-in, First-out*
GNRFET	*Graphene Nano-Ribbon Field-Effect Transistor*
IP	*Intellectual Property*
JT	Junção Túnel
LUT	*Look-Up Table*
MOS	*Metal-Oxide-Semiconductor*
MSB	*Most Significant Bit*
NAND	*Not-AND*
nanoNAND	*Nanoelectronic NAND*
NoC	*Network-on-Chip*
NWFET	*Nanowire Field-Effect Transistor*
OA	*Oblivious Arbiter*
PIPO	*Parallel-in, Parallel-out Register*
PISO	*Parallel-in, Serial-out Register*
QCA	*Quantum Cellular Automaton*
RAM	*Random-Access Memory*
RCLK	*Register Clock*
RoR	*Round-Robin Arbiter*
RWM	*Read-Write Memory*
SET	*Single-electron Transistor*
SIPO	*Serial-in, Parallel-out Register*
SISO	*Serial-in, Serial-out Register*
SoC	*System-on-Chip*
SR	*Shift-Register*
SRAM	*Static Random-Access Memory*
SRCLK	*Shift-Register Clock*
SRwOR	*Shift-Register with Output Register*
VLSI	*Very Large Scale Integration*

1 INTRODUÇÃO

A indústria da microeletrônica nasceu a partir da invenção do transistor em 1947 e do circuito integrado em 1958 [1]. Em 1965, Gordon Moore (co-fundador da *Intel Corporation*) observou que o número de transistores que podem ser colocados em determinada área dobrava a cada 12 meses [2]. Posteriormente esta afirmação ficou conhecida como Lei de Moore e ela se tornou o objetivo de miniaturização para a indústria de semicondutores, incentivando pesquisas e investimentos na área de tecnologia. Novas tecnologias foram desenvolvidas para a fabricação dos transistores, barateando o processo e por consequência popularizando eletrônicos que se tornaram obrigatórios no uso diário.

Atualmente o transistor MOS (*Metal-Oxide-Semiconductor*) é o dispositivo mais amplamente empregado no projeto de circuitos integrados. O seu sucesso se deve principalmente à pouca potência necessária para sua operação e ao seu processo de fabricação ser relativamente simples. O transistor MOS é fabricado em dimensões pequenas e elas têm sido constantemente reduzidas desde de sua criação. A corrida pela miniaturização gerou resultados impressionantes ao longo das décadas desde o surgimento da lei de Moore e em 2014 o menor transistor CMOS fabricado já atingia 14 nm [3].

Com a grande quantidade de transistores em cada *chip* de silício, teve início a era da tecnologia VLSI (*Very Large Scale Integration*). A capacidade de integração do transistor continua aumentando, porém com benefícios de performance e potência limitados. Uma alternativa para esse problema são as arquiteturas *multicore* ou os sistemas-em-*chip* (SoC). Os SoCs porém são limitados pela suas interconexões. Em um SoC as interconexões devem apresentar pequena dimensão, porém quanto menor essa dimensão maiores são a resistividade e a constante RC, levando à um aumento das correntes parasitárias. Uma boa parte da potênica do sistema é utilizada na energização dos fios e a maior parte do ciclo de *clock* é gasto em atrasos nos fios [4]. A tendência de miniaturização dos componentes do sistema não é acompanhada pela densidade dos fios de interconexão, afetando consideravelmente a aréa total do sistema. Por fim, a frequênica de comunicação entre os componentes é bem inferior aos períodos de *clock* dos elementos processadores, limitando a frequênica total do sistema [4].

A solução adotada pelos sistemas modernos de multiprocessamento heterogêneo é a rede-em-*chip* (NoC) [5]. A NoC aplica, em nível do *chip* de sílicio, os princípios bem estabelecidos de redes, colocando a arquitetura de interconexão dentro do *chip*. Uma NoC é uma rede de interconexão de núcleos de processamento baseada em roteadores que permite a redução das interconexões físicas, que é um grande gargalo no desempenho de circuitos eletrônicos atuais.

O elemento central da NoC é o roteador. Ele é responsável por fazer com que os pacotes de dados cheguem ao seu devido destino e deve ser capaz de suportar todas as permutações de entrada-saída ao mesmo tempo, além de resolver contenções para todas as saídas. A ideia de uma

comunicação segmentada é explorada com o intuito de ter maior escalabilidade e flexibilidade, ou seja, uma rede de dados composta de enlaces de comunicação e roteadores que são implementados no *chip*. Por fazer parte de um SoC, é importante que o roteador da NoC possua a menor área possível, além de dissipar pouca potência. O roteador para NoC pode ser otimizado através do uso de outra tecnologia que também é consequência da lei de Moore: a tecnologia nanoeletrônica.

A contínua miniaturização dos transistores fez com que eles chegassem na escala nanométrica. No domínio nanométrico, porém, há uma predominância da física quântica sobre a física clássica, e transistores que tem seu funcionamento baseado na física clássica, como o CMOS, passam a ser afetados de modo não desprezível pela quântica. Em contrapartida a nanoeletrônica usa a física quântica ao seu favor, baseando o funcionamento dos seus dispositivos nela. Dentre os vários dispositivos em desenvolvimento, o transistor monoelétron (SET) é muito promissor [1]. Seu funcionamento é baseado no controle do fluxo de um único elétron ou de um pequeno grupo de elétrons através do tunelamento, apresentando um consumo muito reduzido de potência durante sua operação e um excelente controle de corrente [1]. O SET também ocupa uma pequena área e apresenta uma característica de rápida operação. Por todos esses motivos o SET é uma opção atrativa para desenvolver circuitos com escala de integração THz.

Há 20 anos o grupo de nanoeletrônica do LDCI/UnB vem se dedicando ao estudo de diversos dispositivos nanoeletrônicos, e em particular o SET. Ao longo desse período o SET foi primeiramente estudado como um dispositivo individual, evoluindo para o desenvolvimento de circuitos pequenos e, mais recentemente, sistemas mais complexos. Através desse estudo têm sido exploradas novas arquiteturas, otimizadas para o SET, e técnicas de simulação. Sistemas implementados com o SET são capazes de operar na casa dos THz, porém, de modo semelhante aos SoCs, são limitados pelas suas interconexões, que não são capazes de operar na mesma frequência. Inspirado pelos SoCs, as NoCs se destacaram na busca por formas inteligentes de resolver esse problema. Este tema vêm sendo explorado recentemente pelo grupo, e *Pês et. al.* [6] desenvolveu uma NoC baseada em SET, porém com *Look-Up Table* (LUT) no lugar do roteador. Deste modo este trabalho visa o desenvolvimento de um roteador baseado em SET para NoCs.

1.1 OBJETIVOS

Este trabalho tem como objetivos:

1. Propor, desenvolver, implementar e validar um roteador para NoC com topologia *Mesh* desenvolvido apenas com tecnologia SET.
2. Validar a arquitetura de projeto digital hierárquico.

Para atingir esses objetivos módulos digitais básicos implementados com SET foram criados e validados. Ao final do trabalho são realizadas algumas comparações com a tecnologia CMOS e também são analisadas as dificuldades encontradas.

As contribuições desse trabalho são:

- Apresentação de uma nova arquitetura de roteador.
- Desenvolvimento do primeiro roteador implementado em tecnologia SET.
- Apresentação de arquiteturas originais para os seguintes módulos do roteador:
  - ◇ memória SRAM
  - ◇ registrador FIFO
- Criação de uma biblioteca de circuitos implementados em SET para LTspice.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 5 capítulos e 2 apêndices. O conteúdo de cada um é explicado brevemente a seguir.

1. **Introdução:** apresenta o tema e motivação do trabalho, os objetivos a serem alcançados e as contribuições deste trabalho.
2. **Fundamentação Teórica:** este capítulo apresenta os conceitos básicos mais importantes para o entendimento completo do trabalho, divididos nas seguintes categorias:
   - Nanoeletrônica
   - Redes-em-Chips
   - Arquitetura Digital
3. **Metodologia:** são apresentados a metodologia de desenvolvimento do trabalho, um breve histórico da evolução do modelo final do roteador, considerações sobre a simulação dos circuitos e a metodologia de avaliação do funcionamento dos mesmos.
4. **Análise e Resultados:** em uma abordagem *top-down*, este capítulo apresenta as arquiteturas do roteador e seus componentes e os resultados de validação de cada um. No final é feita uma análise comparativa com a tecnologia CMOS.
5. **Conclusão:** este capítulo apresenta as considerações finais dessa dissertação e trabalhos futuros.
6. **Apêndice A:** neste apêndice é apresentado o desenvolvimento e implementação de todos os módulos digitais básicos utilizados na criação do roteador.
7. **Apêndice B:** este apêndice apresenta o código do SET para SPICE utilizado na simulação dos circuitos implementados.

2 FUNDAMENTAÇÃO TEÓRICA

Nanotecnologia é a ciência do muito pequeno [1]. Ela se refere à qualquer tecnologia que faz uso de materiais ou dispositivos nanoscópicos, i.e., objetos da ordem de nanômetros ($10^{-9}m$) [1]. Em dimensões tão pequenas, prevalece a mecânica quântica sobre a física clássica, e sua compreensão é um requerimento básico para se trabalhar com a nanotecnologia. Este capítulo trata sobre os conceitos de nanoeletrônica, NoCs e arquitetura digital que norteiam este trabalho.

2.1 NANOELETRÔNICA

A nanoeletrônica é a aplicação da nanotecnologia à eletrônica. No caso do dispositivo SET ela se ocupa desde o modo como é controlado o fluxo de elétrons em um único dispositivo de dimensões nanométricas e a sua fabricação, ao *design* e montagem de circuitos que utilizem estes dispositivos de modo eficiente [7]. Ao contrário da microeletrônica, a teoria semiclássica (física clássica modificada pela mecânica quântica) não é suficiente para explicar o funcionamento da nanoeletrônica. Os fenômenos que regem a nanoeletrônica são quânticos e, portanto, um estudo da área exige conhecimento básico de mecânica quântica.

Por ser uma área em desenvolvimento, a nanoeletrônica oferece vários novos dispositivos com diferentes abordagens e soluções para a eletrônica em geral. Alguns desses dispositivos são: autômatos celulares de ponto quântico (QCA), transistores de nanotubo de carbono (CNTFET), transistores de nanofitas de carbono (GNRFET), transistores de nanofio de silício (NWFET) e transistores monoelétron (SET) [8].

Este trabalho foca no dispositivo SET. O funcionamento do SET é baseado no controle do fluxo de um único elétron (ou de um número muito pequeno de elétrons), de modo que o transistor liga e desliga toda vez que um elétron é adicionado ao sistema [7]. Esse dispositivo apresenta como vantagens dimensões extremamente reduzidas, baixo consumo de potência e pouco ruído, possibilitanto eletrônica de alta velocidade e alta densidade. Por outro lado ele exibe baixo ganho, é altamente susceptível a efeitos de cargas aleatórias (cargas de *offset*), e tem sua operação deteriorada a altas temperaturas [1]. Nesta seção serão explicados alguns conceitos básicos e dispositivos quânticos que regem o funcionamento do SET.

2.1.1 Tunelamento

O tunelamento é um efeito quântico, no qual uma partícula com energia total E, inferior à energia de uma barreira de potencial V_0, é transmitida através dela [9] (figura 2.1). Na física clássica, a partícula irá ultrapassar a barreira de potencial com 100% de probabilidade caso $E > V_0$ e será refletida com 100% de certeza caso $E < V_0$.

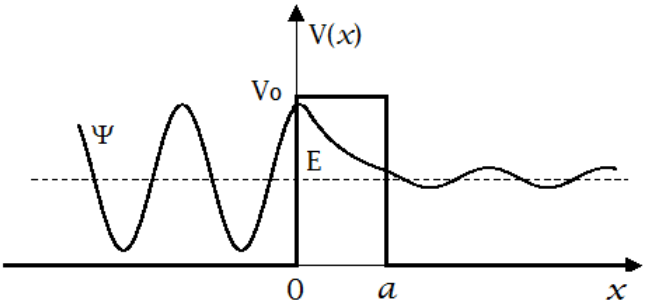

Figura 2.1: Tunelamento de uma partícula com função de onda Ψ através de uma barreira de potencial com comprimento a

No domínio da mecânica quântica, devido à natureza dual partícula-onda do elétron, a transmissão ou não da partícula através da barreira é regida pela probabilidade de tunelamento T (equação 2.1), que é função das soluções particulares da equação de Schrödinger independente do tempo para um determinado perfil de potencial e, portanto, varia de acordo com o material e a espessura da barreira a [10]. Para a maioria dos valores de energia $0 < T < 1$ há uma probabilidade não nula de que o elétron seja transmitido pela barreira [1], como pode-se observar no gráfico da probabilidade de tunelamento pela energia (figura 2.2). A figura 2.2 também evidencia que os valores resultantes da física clássica são casos limite em que $E \ll V_0$ e $E \gg V_0$.

$$T = \frac{4E(E - V_0)}{V_0^2 \sin^2(ka) + 4E(E - V_0)} \tag{2.1}$$

$$k^2 = \frac{2m^*(E - V_0)}{\hbar^2} \tag{2.2}$$

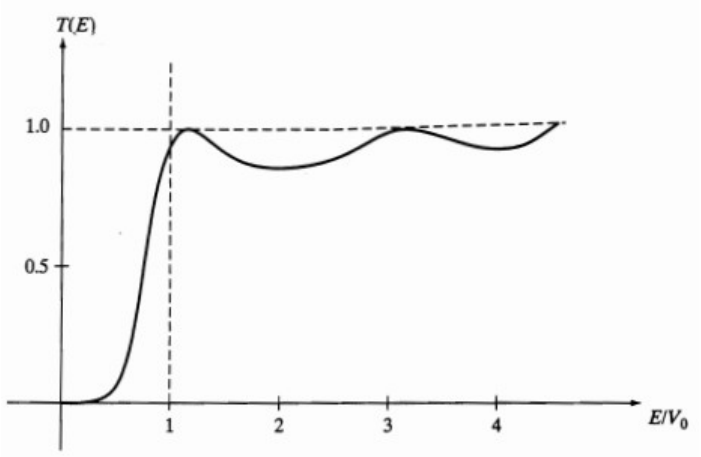

Figura 2.2: Probabilidade de Tunelamento x Energia para uma barreira de potencial [1].

A probabilidade de tunelamento decai exponencialmente em função do comprimento a da barreira e nanodispositivos que transportam carga baseados no tunelamento devem consequentemente possuir barreiras extremamente finas para funcionar adequadamente.

2.1.2 Junção Túnel

A barreira de potencial se dá pela junção de dois materiais diferentes, tipicamente metal-isolante-metal (figura 2.3). A figura 2.3 apresenta o perfil de energia potencial para uma junção metal-isolante-metal, onde $e\Phi$ (função trabalho) é a energia necessária para que os elétrons sejam liberados da superfície do metal. Se o isolante for fino o suficiente, o fluxo de corrente através dessa junção se dará por tunelamento (subseção 2.1.1), e a ela damos o nome de junção túnel (JT). O comportamento da junção túnel se assemelha ao de um capacitor de fuga (de um ponto de vista semiclássico, também chamado de nanocapacitor) e portanto é modelado como um capacitor ideal C em paralelo com uma resistência R_t (resistência de tunelamento) [1, 10]. A figura 2.4 apresenta o esquemático do modelo e o símbolo da junção túnel.

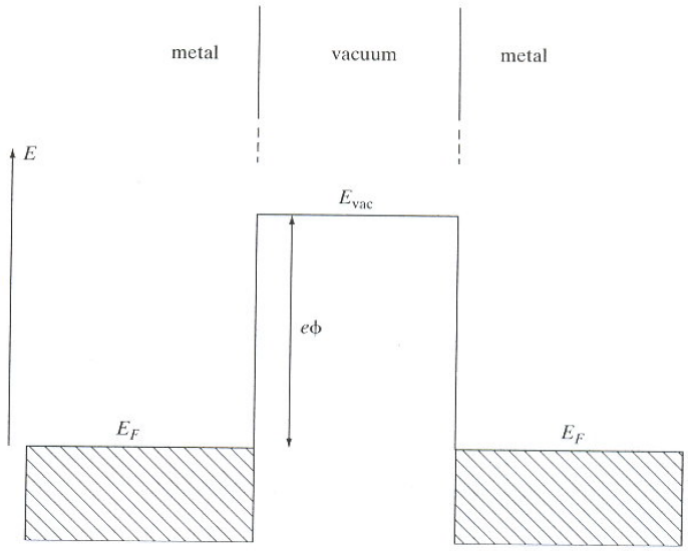

Figura 2.3: Perfil de potencial de uma junção túnel metal-isolante-metal

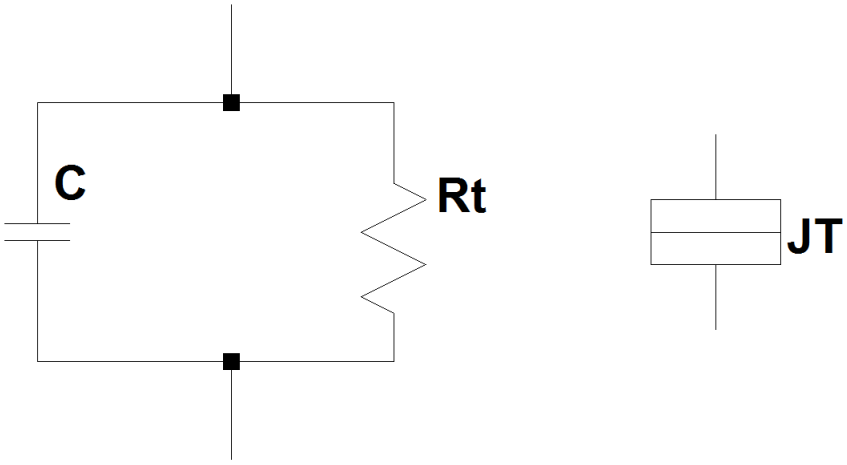

Figura 2.4: Modelo e símbolo da junção túnel

Para que este modelo represente adequadamente uma junção túnel, deve-se ter especial atenção quanto aos valores de C e R_t. Essencialmente a junção metal-isolante-metal funciona fisica-

mente como um capacitor, porém com dimensões nano. Isso significa que as equações 2.3 e 2.4 se aplicam ao modelo, porém A e d são extremamente pequenos (d neste caso é o comprimento da barreira de potencial), o que leva à valores da ordem de femto-ato Faradays [1].

$$Q = CV \tag{2.3}$$

$$C = \frac{\varepsilon A}{d} \tag{2.4}$$

A resistência de tunelamento R_t é dada pela Lei de Ohm (equação 2.5), onde V é a tensão aplicada sobre a junção e I é a corrente resultante devido ao tunelamento. A resistência de tunelamento não é uma resistência ôhmica clássica, mas sim resultado de um fenômeno mecânico-quântico [7]. Conceitualmente é R_t que permite que os elétrons atravessem a junção isolante como eventos discretos, e ela está relacionada com a espessura da barreira de potencial. Para garantir que o tunelamento ocorra através do capacitor, R_t deve ser grande, mas não grande o suficiente para impedir o tunelamento. Como regra geral a equação 2.6 deve ser satisfeita, onde R_0 é a resistência quântica [1].

$$R_t = \frac{V}{I} \tag{2.5}$$

$$R_t \gg R_0 = \frac{h}{q_e^2} \simeq 25.8\ k\Omega \tag{2.6}$$

2.1.3 Ponto Quântico e Ilha Quântica

Pontos quânticos são regiões de material nanoscópico tridimensionais (figura 2.5) com dimensões de comprimento L_x, L_y e L_z pequenas o suficiente para confinar elétrons. Por ser uma onda de matéria, o elétron está sujeito ao princípio de confinamento que estabelece que o confinamento de uma onda leva à existência de estados discretos com energias discretas, i.e., à quantização [11].

Para que um material nanoscópico seja um ponto quântico, é necessário que $L_x, L_y, L_z \leq \lambda_F$, onde λ_F é o comprimento de onda de Fermi do elétron [10]. Assim:

$$E = \frac{\hbar^2\pi^2}{2m_e}\left[\left(\frac{n_x}{L_x}\right)^2 + \left(\frac{n_y}{L_y}\right)^2 + \left(\frac{n_z}{L_z}\right)^2\right] = E_{n_x,n_y,n_z} \tag{2.7}$$

onde n_x, n_y, n_z representam os números quânticos para cada uma das três dimensões de confinamento do elétron e E_{n_x,n_y,n_z} representa energias discretas. Como um elétron possui um comprimento de onda de Fermi λ_F da ordem de nanômetros, então pontos quânticos devem possuir dimensões da ordem de alguns até centenas de nanômetros [1]. Neste aspecto, pode-se dizer que o ponto quântico é zero-dimensional. O ponto quântico é chamado de ilha quântica ou ilha de Coulomb quando há fluxo de corrente por tunelamento através de si.

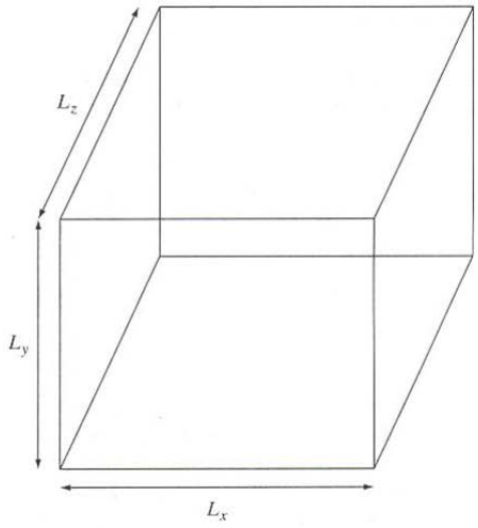

Figura 2.5: Região de confinamento tridimensional com $L_x, L_y, L_z \leq \lambda_F$

2.1.4 Bloqueio de Coulomb

O bloqueio de Coulomb é um fenômeno exclusivo da nanoescala. Ele é responsável por suprimir o fluxo de corrente por tunelamento em um dispositivo para valores de tensão inferiores a limiares específicos (estados de energia) quantizados. O dispositivo mais simples onde é possível observar este efeito é a junção túnel polarizada por uma tensão V. Em seu modelo semiclássico como nanocapacitor (subseção 2.1.2), a energia armazenada em seu campo eletrostático inicialmente (E^i) e a energia armazenada após o tunelamento de um único elétron (E^f) são dadas por:

$$E^i = \frac{Q^2}{2C} \tag{2.8}$$

$$E^f = \frac{(Q + q_e)^2}{2C} \tag{2.9}$$

onde q_e representa a carga de um único elétron.

Para que ocorra o tunelamento, a variação de energia no nanocapacitor $\Delta E > 0$, ou seja, deve haver um armazenamento de energia no capacitor, tornando a ocorrência do evento energeticamente favorável. Têm-se então que a corrente por tunelamento ocorrerá apenas quando

$$|V| > \frac{|q_e|}{2C} \tag{2.10}$$

Este efeito é conhecido como bloqueio de Coulomb e a figura 2.6 ilustra sua característica $I \times V$ em um nanocapacitor. Considerando que $q_e V$ é energia, têm-se que a energia necessária para adicionar uma carga ao capacitor, i.e., a energia de carregamento do capacitor é dada pela equação 2.11:

$$E_c = \frac{q_e^2}{2C} \tag{2.11}$$

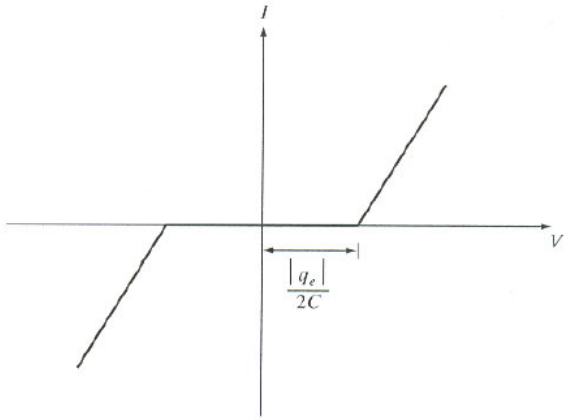

Figura 2.6: Bloqueio de Coulomb em um nanocapacitor.

O circuito de ponto quântico é definido como um circuito em que o ponto quântico é conectado a fonte de tensão através de isolantes finos, o que forma essencialmente duas junções túnel (figura 2.7). O tunelamento de életron permite que um número discreto n de elétrons seja acumulado na ilha de um circuito de ponto quântico (equação 2.12).

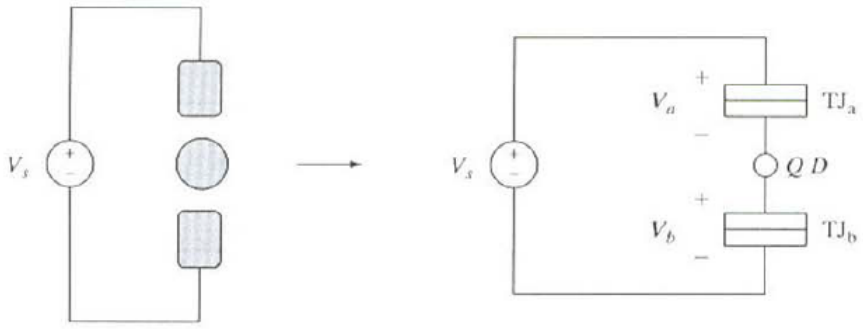

Figura 2.7: Circuito de ponto quântico.

$$Q = n \times q_e, n = 1, 2, 3, ... \tag{2.12}$$

Essa configuração também é conhecida como barreira dupla de tunelamento, que pode ser simplificada como um poço de potencial finito na região entre as duas barreiras das junções túnel (figura 2.8).

Dada a definição de ponto quântico e devido ao princípio de confinamento (subseção 2.1.3), a energia dentro do poço será quantizada com valores que são múltiplos inteiros da equação 2.11. Por consequência, temos que, para cada novo elétron a ser adicionado à ilha, a tensão fornecida V_S deverá ser [1]:

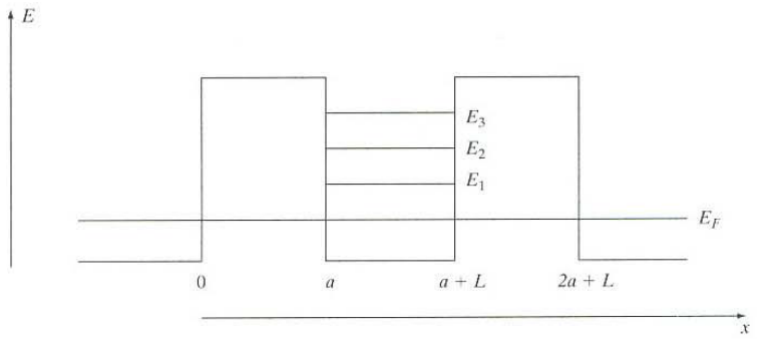

Figura 2.8: Barreira dupla de tunelamento

$$|V_S| > \frac{me}{2C}, m = 1, 3, 5, ... \tag{2.13}$$

Devido à quantização de energia na ilha, o bloqueio de Coulomb assume o formato de uma escada (escada de Coulomb), conforme ilustra a figura 2.9, quando uma diferença de tensão é aplicada a um circuito de ponto quântico. Isso ocorre pois, a cada vez que um elétron tunela para dentro da ilha, a energia da ilha aumenta em $e^2/2C$, reestabelecendo o bloqueio de Coulomb até que a tensão aumente de acordo com a equação 2.13, ou até que um elétron tunele para fora da ilha.

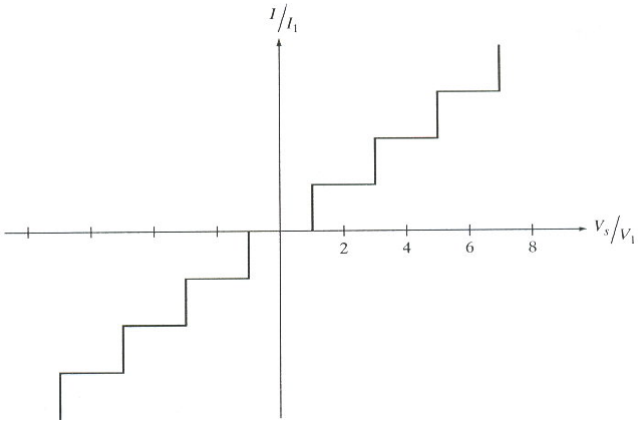

Figura 2.9: Escada de Coulomb

2.1.5 Transistor Monoelétron

O transistor mono-elétron é composto de um ponto quântico (ou ilha) situado entre duas junções túnel e conectado a um terminal de porta (figura 2.10). As junções túnel isolam a ilha dos conectores externos ligados a fonte de tensão fonte-dreno e o terminal de porta é isolado da ilha através de uma capacitância ideal (que não permite tunelamento) de porta [1].

A tensão de porta permite um controle adicional sobre o fluxo de corrente através das duas junções túnel. Ao variar V_{g} modifica-se energia de Fermi (energia do nível ocupado mais ene-

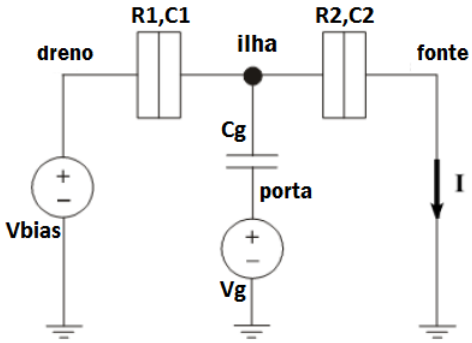

Figura 2.10: Esquemático do transistor mono-elétron.

gético) na ilha e, portanto, a tensão V_{s} necessária para geração de fluxo de corrente no SET. A figura 2.11 ilustra a diminuição da energia de Fermi na ilha quando $V_{\mathrm{g}} > 0$, fazendo com seja necessário uma tensão V_s menor para superar o bloqueio de Coulomb .

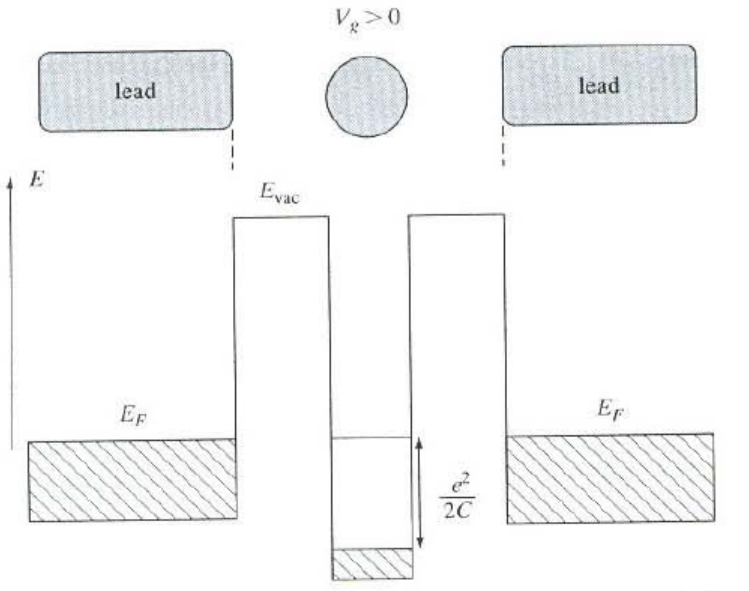

Figura 2.11: Efeito de $\mathrm{V_g}$ na energial de Fermi na ilha.

O bloqueio de Coulomb agora assume o formato de um diamante quando observado do ponto de vista das tensões V_{S} e V_{g} (figura 2.12). As áreas hachuradas do diamante são regiões onde não há fluxo de corrente, i.e., onde ocorre o bloqueio de Coulomb. O número dentro do diamante hachurado indica quantos elétrons existem dentro na ilha de modo estável. Por exemplo, o diamante com o número 0 indica que, para os valores de V_S e V_g dentro de sua área, não há fluxo de corrente e não existe nenhum elétron dentro da ilha. Para que haja corrente, V_g ou V_S devem ser modificados de modo a superar o bloqueio de Coulomb. A numeração dos demais diamantes informa quantos elétrons podem tunelar através da ilha e quantos elétrons podem permanecer na ilha. Por exemplo, o diamante $0, 1$ indica que um elétron pode tunelar através da ilha, mas nenhum elétron pode permanecer na ilha. O gráfico formado pelos diversos diamantes no plano $V_{\mathrm{S}} \times V_{\mathrm{g}}$ é denominado de diagrama de estabilidade e informará as regiões determinadas pelos

valores de tensão que são mais prováveis de se encontrar carga na ilha.

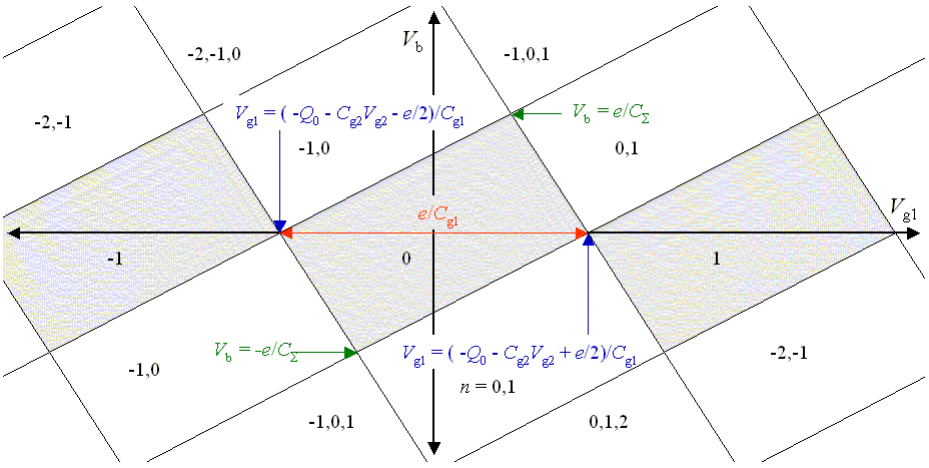

Figura 2.12: Diagrama de estabilidade de Coulomb

2.2 REDES-EM-CHIPS

Vários dos atuais aparelhos eletrônicos (celulares, *tablets*, etc.) são compostos de centenas a milhares de elementos processadores (IP) integrados em um único *chip* de silício, também conhecidos como *Systems on Chip* (SoC) [5]. Em um espaço tão reduzido, é um desafio estabelecer as conexões físicas e lógicas entre todos os IPs de forma funcional e eficiente. As formas de conexões tradicionais não são suficientes para resolver estes problemas. Trilhas dedicadas (ponto-a-ponto, figura 2.13b) conectam apenas dois componentes e seu número cresce exponencialmente com o número de elementos, tornando seu uso impossível em um SoC. Barramentos compartilhados (*buses*, figura 2.13a) deterioram com mais de dez elementos conectados, apresentam limitações quanto à largura de banda, consomem muita potência, geram ruídos e não conseguem acompanhar a escalabilidade dos atuais sistemas [12].

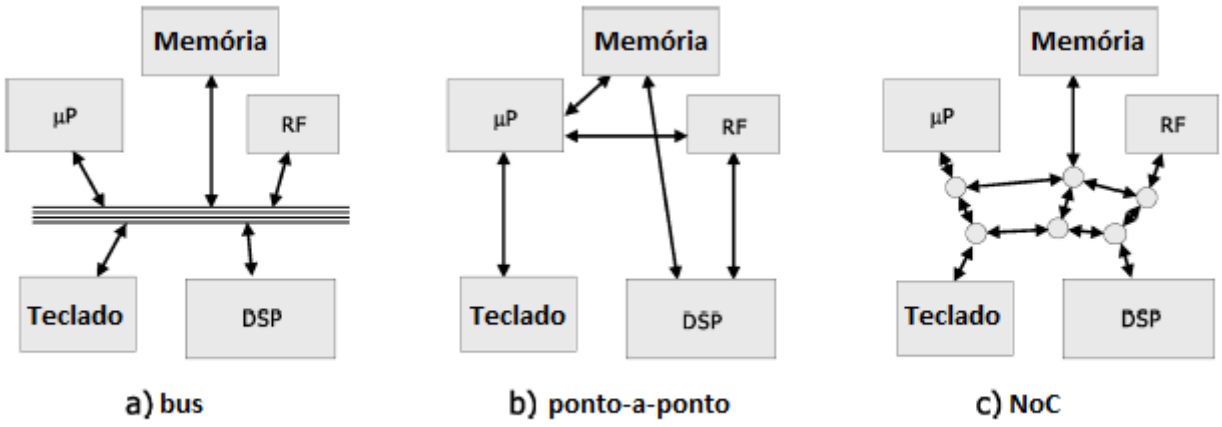

Figura 2.13: Conexões ponto-a-ponto, barramento e NoC [13]

Uma resposta para esse problema são as redes-em-chip (figura 2.13c) [4, 5, 12, 13]. A ideia de NoC é inspirada nas redes de computadores tradicionais, de modo que ela aplica os mesmos princípios bem estabelecidos dessas redes ao nível do *chip* de silício. O *design* de uma NoC é um problema multidimensional que envolve componentes tanto de *hardware* quanto de *software*, como topologias, interface de rede, roteadores e algoritmos de roteamento [5].

2.2.1 Topologias de Rede

A topologia de rede define a conexão entre os IPs através de enlaces, referindo-se à forma física (ou geométrica) da rede e o modo como os elementos estão conectados entre si. Existem várias topologias para NoCs, dentre as quais pode-se citar *SPIN, Mesh, Torus, Ring* e *Butterfly* [13]. A figura 2.14 ilustra três dessas topologias: *SPIN, Mesh* e *Torus*. Os quadrados pretos são roteadores, enquanto os quadrados brancos são IPs.

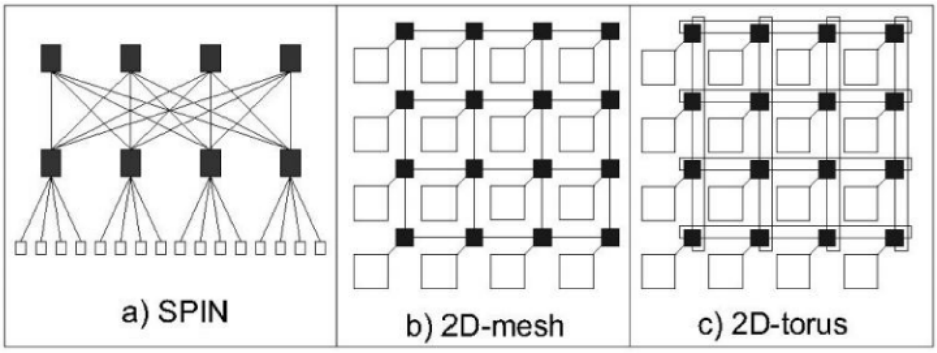

Figura 2.14: Topologias *SPIN, Mesh* e *Torus*

Dentre essas topologias, a *Mesh* é de especial interesse. É possível observar na figura 2.14b que cada IP é diretamente conectado a um único roteador, e um típico roteador, que não esteja nas bordas da rede, é conectado em quatro direções (norte, sul, leste e oeste) a outros quatro roteadores. A topologia *Mesh* tem sido muito utilizada por pesquisadores devido ao seu *layout* eficiente, boas propriedades elétricas e simplicidade de endereçamento de recursos *on-chip* [14]. Por esses motivos este trabalho utiliza a rede *Mesh* como premissa para o desenvolvimento do roteador nanoeletrônico. Porém é importante ressaltar que, devido à sua natureza modular, este roteador pode ser facilmente adaptado para outras topologias.

2.2.2 Roteador

Os roteadores são o coração das NoCs, e sua função é transferir dados de fonte para destinatário através da conexão de várias entradas e saídas de modo a permitir a implementação arbitrária de topologias de rede [5]. São os roteadores que implementam as camadas física de transporte de uma NoC e eles afetam a velocidade, a vazão, a área e a energia da NoC no sistema.

De modo genérico, um roteador é composto por três partes: a lógica de entrada, o tecido de interconexão (*crossbar*) e a lógica de saída (figura 2.15) [15].

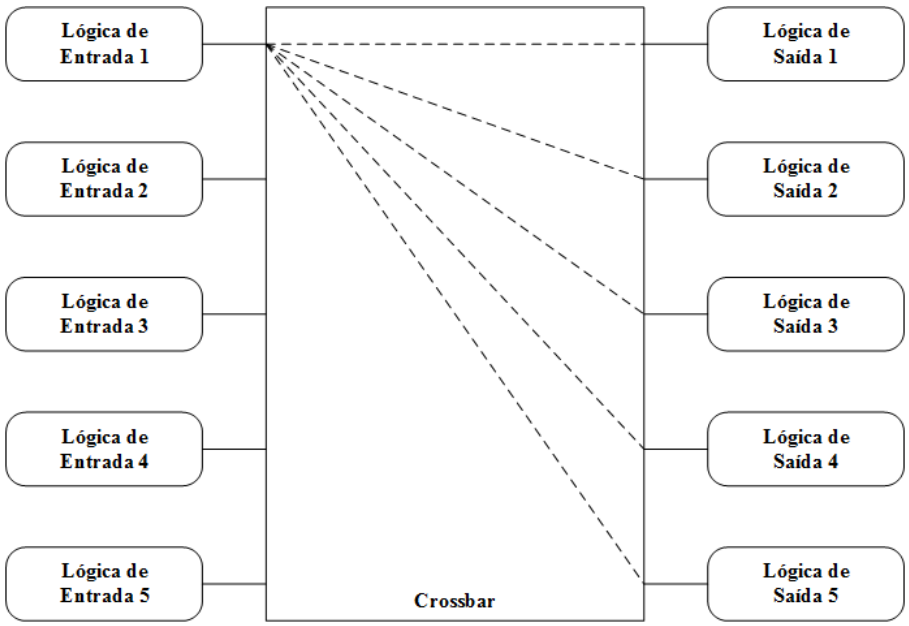

Figura 2.15: Modelo genérico de um roteador

A lógica de entrada deve ser capaz de extrair o endereço dos dados de entrada e possuir *buffers* para o armazenamento temporário dos mesmos durante a execução de todo o processo de roteamento. O *crossbar* costuma ser composto por um elemento demultiplexador, que é o responsável por conectar a entrada à saída correta. A lógica de saída costuma possuir um árbitro, que é responsável por alocar o canal de saída de acordo com os pedidos das várias entradas à ele conectadas.

2.3 ARQUITETURA DIGITAL

O roteador é um dispositivo essencialmente digital. Como tal ele é composto por módulos de menor complexidade, chegando ao nível das portas lógicas, que são os elementos mais básicos da arquitetura digital [16]. A subseção a seguir trata da porta NAND nanoeletrônica, bloco básico sobre o qual foi desenvolvido o roteador deste trabalho. Diversos módulos digitais básicos foram desenvolvidos a partir da porta NAND nanoeletrônica, os quais já são muito bem estabelecidos na literatura em sua versão CMOS. Por esse motivo tais módulos serão apresentados apenas nos Apêndices, visto que suas arquiteturas são simples e diretas, mas faz-se necessário validá-los, dado que seu funcionamento para compor o roteador proposto neste trabalho advém do SET, e não do CMOS. Uma lista com todos esses módulos é apresentada a seguir:

- Porta NOT
- Porta OR
  - ◇ Porta OR de 2 Entradas

    - ◇ Porta OR de 3 Entradas
    - ◇ Porta OR de 4 Entradas
- Porta AND
    - ◇ Porta AND de 2 Entradas
    - ◇ Porta AND de 3 Entradas
    - ◇ Porta AND de 4 Entradas
- Porta NAND de 3 Entradas
- Porta NOR de 4 Entradas
- Porta XOR
- Decodificador Binário
- Demultiplexador
- *Latch* SR com *Clear* Assíncrono
- *Latch* D com *Clear* Assíncrono
- *Latch* D com *Preset* Assíncrono
- *Flip-Flop* D com *Clear* Assíncrono
- *Flip-Flop* D com *Preset* Assíncrono
- Divisor de Frequência
- PISO
- Contador Binário

2.3.1 Porta NAND Monoelétron

Em lógica, um conjunto de operadores possui a propriedade de completude funcional quando todos os demais operadores podem ser definidos a partir desse conjunto [17].Toda função booleana pode pode ser expressa através da combinação do conjunto mínimo de operadores funcionalmente completo constituído por: AND (E), OR (OU) e NOT (NÃO). Em 1913 Sheffer [18] provou que o conjunto mínimo de operadores funcionalmente completo booleano poderia ser reduzido a uma única operação binária primitiva, a operação NAND (NÃO-E), também conhecida como conectivo de Sheffer. Portanto, o bloco básico digital deste projeto é a porta NAND de duas entradas e todos os demais elementos digitais nos circuitos foram criados a partir dela.

A porta NAND utilizada neste trabalho é nanoeletrônica, i.e., ela é constituída por SETs. Há algumas arquiteturas de portas NAND nanoeletrônicas propostas na literatura [19, 20, 21]. A

proposta escolhida é a porta NAND de [21], visto que suas configurações a tornam capaz de operar a temperatura ambiente. Esta arquitetura, que é uma porta programável, é apresentada na figura 2.16 e seus parâmetros encontram-se na tabela 2.1. As tensões V_a e V_b são as entradas de sinais digitais A e B da porta NAND. A tensão $V3$, chamada de *select input*, seleciona a função NOR na saída da porta quando seu valor é alto $(0,5\ V)$, e seleciona a função NAND quando seu valor é baixo $(0\ V)$.

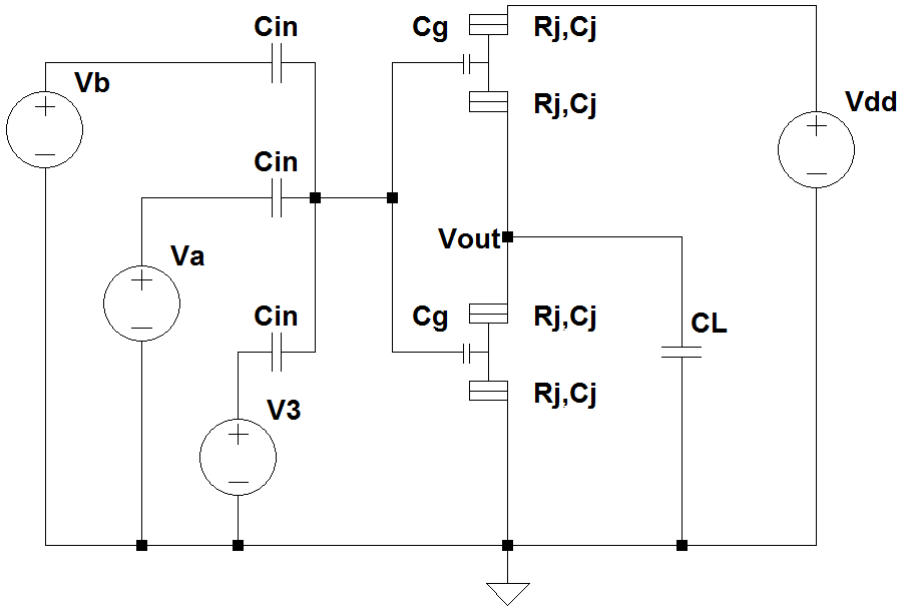

Figura 2.16: Porta NAND nanoeletrônica.

Tabela 2.1: Parâmetros da porta NAND nanoeletrônica

Parameter	Value
R_j	1 MΩ
C_j	0.001 aF
C_{in}	2 aF
C_g	0.15 aF
C_L	0.25 aF
V_{dd}	0.5 V
V_3	0 V

2.3.2 Registrador de Deslocamento de 8 bits com Registrador de Saída

Um registrador é a coleção de dois ou mais *flip-flops* D governados pelo mesmo sinal de *clock* e é utilizado para o armazenamento de um bit [16]. O registrador de deslocamento (SR) é um registrador de n-bits com capacidade de deslocar a posição do dado armazenado em um bit a cada pulso de *clock*. Esse deslocamento pode ser unidirecional, i.e., apenas para a direita ou para a esquerda, ou bidirecional, caso em que uma entrada de controle é utilizada para especificar a direção de deslocamento. Existem quatro tipos básicos de registradores de deslocamento:

- *Serial-in, Serial-out* (SISO): novos bits são apresentados na entrada serial e são movidos para o próximo registrador a cada pulso de *clock*. Um bit é apresentado na saída serial após n pulsos de *clock*, onde n é o número total de registradores no SR. Desse modo um SISO de n bits pode ser usado para atrasar um sinal em n períodos de clock (figura 2.17).
- *Serial-in, Parallel-out* (SIPO): neste SR cada um dos seus registradores internos oferecem uma saída, ou seja, há uma saída para cada bit do SR, tornando-os disponíveis para outros circuitos. Este SR é utilizado para fazer conversão serial-paralelo de dados (figura 2.18).
- *Parallel-in, Serial-out* (PISO): neste SR há uma entrada disponível para cada bit e, a cada pulso de *clock* os registradores carregam os novos bits ou simplesmente desloca os bits já armazendos, de acordo com a entrada de controle *Load/Shift*. Este SR portanto é utilizado para fazer conversão paralelo-serial de dados (figura 2.19).
- *Parallel-in, Parallel-out* (PIPO): se, além da saída serial no PISO, for acrescentada uma saída para cada bit, têm-se um SR *parallel-in, parallel-out*. Este SR é o mais genérico e pode ser utilizado no lugar de qualquer um dos SR citados anteriormente, por possuir entradas e saídas paralelas e seriais. O PIPO é portanto considerado o registrador de deslocamento universal (figura 2.20).

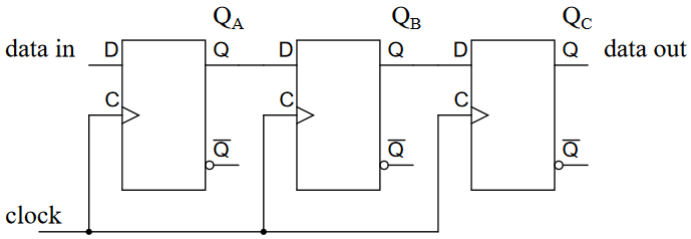

Figura 2.17: Exemplo de um registrador SISO [22]

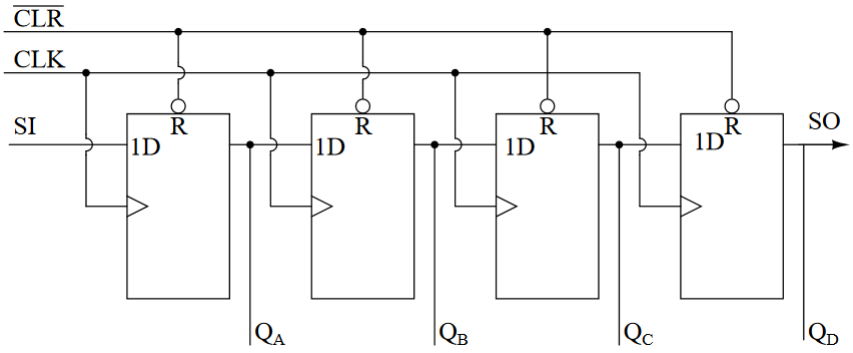

Figura 2.18: Exemplo de um registrador SIPO [22]

Em alguns casos, os SR são acoplados a registradores de armazenamento com sinal de *clock* separado do sinal principal do SR, para persistência de dados. É necessário um registrador de armazenamento (i.e. um *flip-flop* D) para cada bit do SR. Esses SR são chamados de registradores de deslocamento com registrador de saída (SRwOR). Neste trabalho são utilizados SRwOR do tipo SIPO de 8 bits.

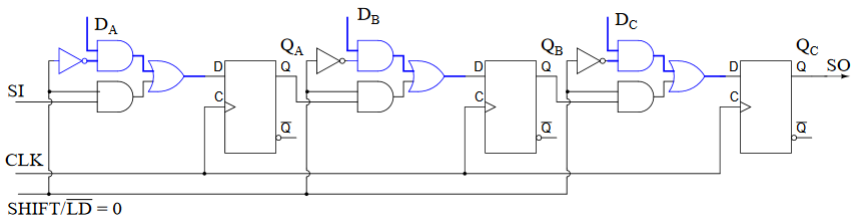

Figura 2.19: Exemplo de um registrador PISO [22]

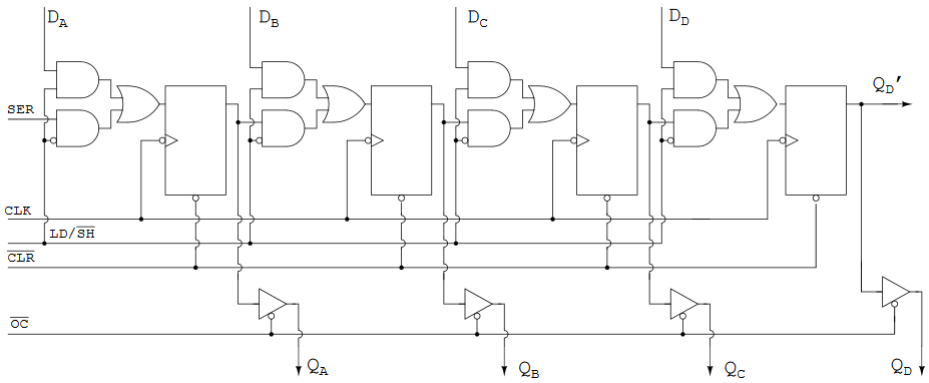

Figura 2.20: Exemplo de um registrador PIPO [22]

2.3.3 Memória SRAM

Memórias capazes de armazenar e oferecer a informação armazenada a qualquer momento são chamadas de memórias de leitura/escrita (RWM). Se o tempo para ler ou escrever um bit na RWM for independente da localização desse bit na memória, ela passa a ser chamada de memória de acesso aleatória (RAM) [16]. Memórias RAM podem ser estáticas (SRAM) ou dinâmicas (DRAM). A SRAM mantém o armazenamento dos dados enquanto seu chip estiver sendo alimentado. A DRAM necessita que seus dados sejam lidos e reescritos periodicamente (refresco) para que o armazenamento seja mantido.

A SRAM possui apenas duas operações possíveis: a de escrita e a de leitura. Ela é tipicamente constituída por *arrays* de células de memória, que são *latches* ou *flip-flops* acrescidos de uma lógica para possibilitar as operações de leitura e escrita quando tal célula for selecionada. Os *arrays* são selecionados através de um endereçamento normalmente realizado por decodificadores. A SRAM pode ser classificada como *single-port*, quando possui apenas um endereçamento, uma seleção de escrita/leitura e uma entrada de dados, ou como *dual-port* quando possui endereçamento duplo, duas seleções de escrita/leitura e duas entradas de dados.

2.3.4 Registrador *First-in, First-out* (FIFO)

Registradores FIFO (também chamados de memória FIFO) operam com base no princípio que o dado mais antigo na fila, i.e., o que chegou primeiro, será o primeiro dado a ser processado. Neste sentido registradores FIFO costumam ser utilizados para transferência de dados de um

domínio de *clock* para outro [16]. O modelo mais simples de FIFO é o registrador SISO, em que o primeiro bit a chegar é o primeiro bit a sair do sistema.

Uma implementação usual do FIFO é o FIFO circular. Neste caso, ao invés de mudar a posição dos dados, como no SISO por exemplo, têm-se um ponteiro de escrita e um de leitura, que indicam em qual posição (ou endereço) deve ser realizada a operação. Quando um ponteiro atinge o final da memória ele simplesmente volta para a posição inicial, e novos dados são escritos sobre os dados antigos.

Normalmento FIFOs são compostos por memórias SRAM como dispositivo de armazenamento, possuem a lógica dos ponteiros de leitura e escrita, e uma lógica para as *flags full* e *empty*, que indicam se a memória está cheia ou vazia.

2.3.5 *Buffer* Elástico

O *buffer* elástico (EB) é o tipo de registrador mais primitivo capaz de implementar o protocolo *ready/valid handshake* [5]. Para tanto é necessário que o EB esteja presente nas interfaces tanto no remetente (*sender*) quanto no destinatário (*receiver*) (figura 2.21). O EB aceita novos dados e transfere dados disponíveis apenas quando os sinais de *ready* e *valid* forem ambos iguais a 1.

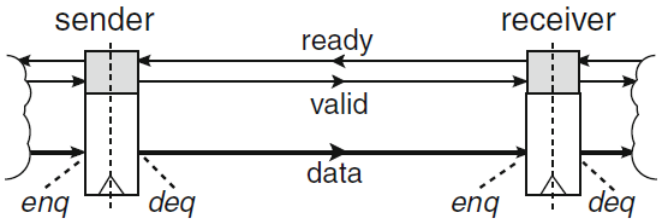

Figura 2.21: *Buffer* elástico nas interfaces do *sender* e *receiver* [5]

2.3.6 Árbitro *Round Robin*

Os árbitros são componentes essenciais de um roteador. São eles os responsáveis por resolver múltiplos pedidos de um mesmo recurso [4]. Cada porta de saída de um roteador é compartilhada por todas as portas de entrada (agentes) e um árbitro é necessário para distribuir o acesso à esse recurso para uma porta de entrada de cada vez. A figura 2.22 apresenta o símbolo de um árbitro. As entradas *ri* são chamadas de *request lines* e é através delas que o agente solicita acesso ao recurso sendo arbitrado. O árbitro seleciona um desses pedidos através das *grant lines gi*. As entradas *hi* são as *hold lines* e são responsáveis por permitir que um *grant* seja mantido por uma quantidade arbitrária de tempo.

A propriedade chave de um árbitro é a sua justiça. Normalmente a justiça é definida em três categoria:

- Justiça fraca: todos os pedidos serão eventualmente atendidos;

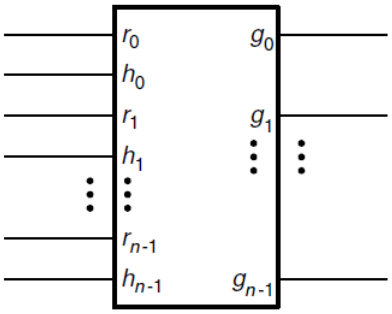

Figura 2.22: Símbolo de um árbitro [4]

- Justiça forte: os pedidos serão atendidos frequentemente igualmente;
- Justiça FIFO: os pedidos são atendidos na ordem em que os pedidos são feitos.

Os árbitros também podem ser classificados quanto à sua prioridade. Árbitros de prioridade fixa são considerados injustos. Árbitros de prioridade variável iterativa mudam sua prioriadade a cada ciclo e são considerados justos. Seu tipo de justiça depende do gerador do bit de prioridade *pi*. Um árbitro com gerador de prioridade cujo pricípio de operação seja o pedido que acabou de ser servido deverá ter a prioriadade mais baixa no próximo ciclo é chamado de árbitro *round-robin* (RoR). Árbitros *round-robin*, para uma demanda igualmente distribuida, apresentam justiça forte. Apesar de existirem árbitros mais eficientes e que promovem melhor justiça, os árbitros *round-robin* são muito utilizados em roteadores para NoCs.

3 METODOLOGIA

Este capítulo apresenta a metodologia de desenvolvimento do roteador e seus sub-circuitos, um breve histórico das primeiras arquiteturas desenvolvidas e sua evolução para o modelo atual, e, por fim, a metodologia de avaliação dos circuitos desenvolvidos.

3.1 PROCEDIMENTO METODOLÓGICO

Para atingir objetivo de desenvolver um roteador para NoC funcional utilizando a tecnologia SET, foi realizada uma investigação na literatura científica para avaliar arquiteturas de roteadores previamente implementadas em outras tecnologias que pudessem ser tomadas como base. A partir dessas arquiteturas, os módulos digitais que compõem o roteador foram identificados e estudados. Cada um dos módulos foi projetado utilizando uma metodologia hierárquica baseada na porta NAND nanoeletrônica. Para tanto uma pequena biblioteca de circuitos digitais básicos foi construída. Todos os circuitos básicos desenvolvidos para implementar o roteador nanoeletrônico estão apresentados no Apêndice A.

A arquitetura de um roteador pode se tornar bastante complexa dependendo da quantidade de funções que ele implementa. Por este motivo o roteador proposto neste trabalho foi desenvolvido em três etapas, partindo de uma arquitetura mais simples, cuja única função é o roteamento propriamente dito entre as entradas e saídas. Para uma melhor compreensão do desenvolvimento do roteador apresentado neste trabalho, as duas primeiras arquiteturas desenvolvidas nas etapas correspondentes serão apresentadas e brevemente explicadas na próxima seção. A arquitetura final será analisada no capítulo 4.

3.2 HISTÓRICO

A primeira arquitetura a ser desenvolvida (figura 3.1) constitui um roteador simples de 4 entradas e 4 saídas. Composto apenas de um DEMUX 1:4 e um PISO 4:1, este roteador recebe como pacotes na entrada bits individuais, e o endereçamento é determinado por uma *Look-Up Table* (LUT) inserida manualmente no sistema. Os bits recebidos das 4 entradas são serializados na saída. Este roteador demonstra a capacidade de roteamento em um sistema desenvolvido com SET [23].

A etapa seguinte preocupou-se em adequar a arquitetura anteriormente desenvolvida para o funcionamento em uma rede de topologia *Mesh*. Portanto a segunda arquitetura foi expandida para 5 entradas e 5 saídas. Devido à capacidade modular do sistema digital e o seu desenvolvimento de natureza hierarquica, essa expansão foi obtida através da adição de um DEMUX e um

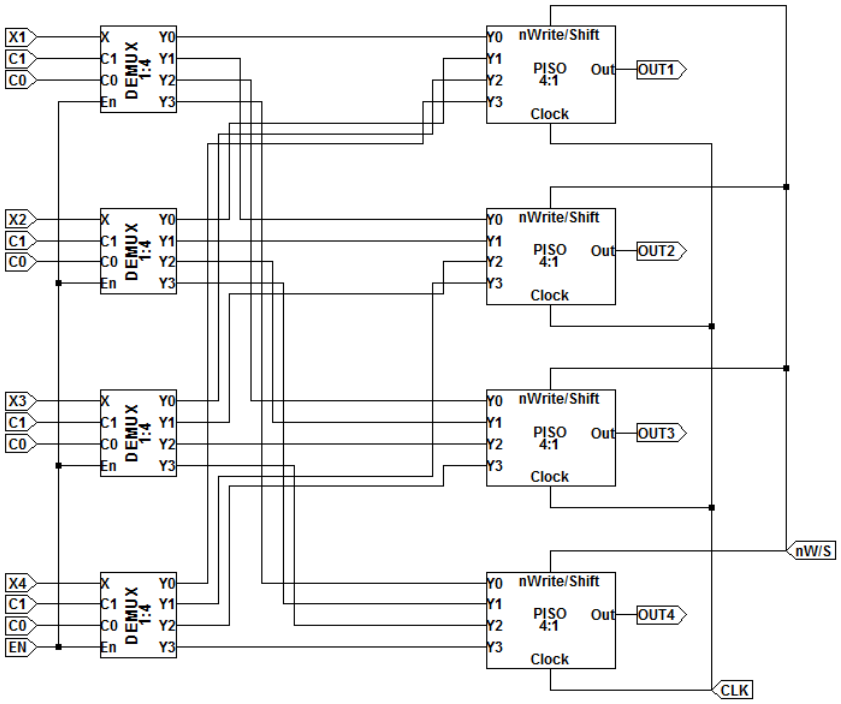

Figura 3.1: Primeiro roteador desenvolvido

PISO. Devido à nova quantidade de entradas e saídas os DEMUXes foram substituídos por sua versão 1:8 e os PISOs por sua versão 5:1.

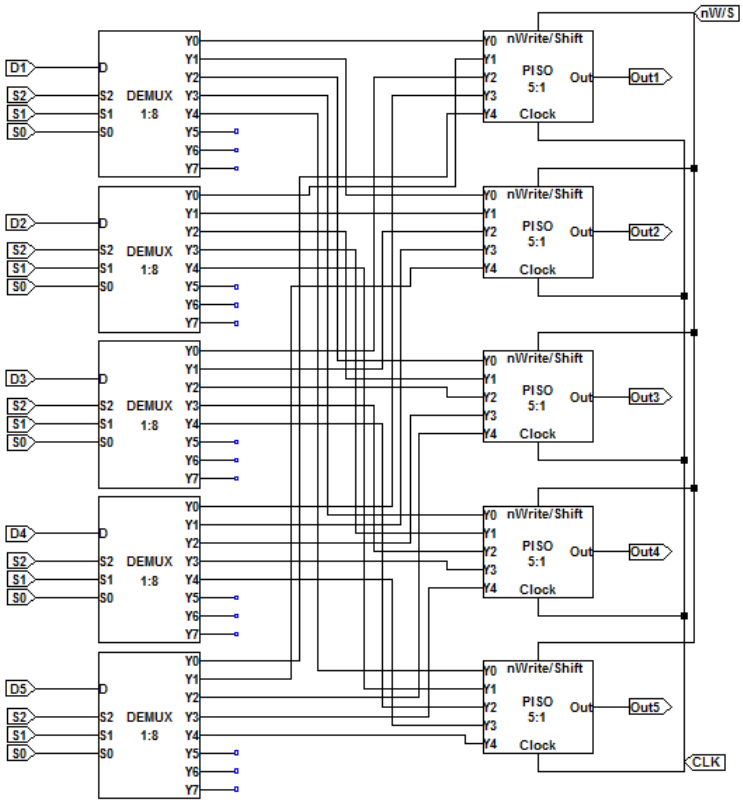

Figura 3.2: Segundo roteador desenvolvido

3.3 SIMULAÇÃO

As simulações realizadas neste trabalho foram executadas através do *software* LTspice IV [24], um simulador SPICE de alta performance. Muito utilizado no meio acadêmico, o LTspice é um dos melhores e mais robustos simuladores de circuitos *freeware*. O seu editor de esquemático possibilita a construção e modificação de circuitos de modo fácil e rápido. Além disso ele apresenta a capacidade de criação e adição de modelos macro, tornando mais simples a montagem de circuitos muito complexos.

Apesar de ser muito eficiente na simulação de circuitos baseados em CMOS, o mesmo não é verdadeiro para circuitos SET. O modelamento do dispositivo SET é feito através de equações estocásticas, e o algoritmo Newton-Rhapson utilizado pelo LTspice não é otimizado para o trabalho com este tipo de equações. No entanto o LTspice se mostrou uma opção mais adequada para a simulação deste trabalho do que seus concorrentes, que apresentam problemas como: quantidade limitada de elementos totais no circuito, impossibilidade de criação de macros e falta de simulação transiente do circuito.

Para as simulações foi utilizado o modelo SPICE para SET proposto por *Liesntsching et. al.* [**??**] (Apêndice B), que opera a temperatura ambiente. Os sinais utilizados em todas as simulações apresentam frequência na casa dos MHz. Em todas as simulações o período do sinal de *clock* é $0,5ms$. O modelo SPICE do SET utlizado foi testado exaustivamente em trabalhos anteriores do LDCI e esta faixa de frequência para os sinais, bem como o período específico para o *clock* foram determinados como sendo valores ótimos para a simulação de circuitos com esse dispositivo [25].

A máquina utilizada para a simulação possui um processador Intel com 4 núcleos com operação a 2,5 GHz, 8 GB de memória RAM e HD de 500 GB SSD. Apesar de ser um computador rápido e processar suavemente a maioria dos programas mais pesados, foram necessárias várias horas para a simulação do roteador completo.

3.4 METODOLOGIA DE AVALIAÇÃO

Entre módulos simples e complexos foram desenvolvidos 45 circuitos ao todo durante a concepção da arquitetura final do roteador. Devido ao modo como a arquitetura final evoluiu a partir de arquiteturas mais simples, o projeto não foi orientado à testabilidade. Para os módulos mais simples, cujos comportamentos são completamente descritos em tabelas verdades, a validação consistiu em testar todas as combinações possíveis de entradas e comparar as saídas obtidas com aquelas esperadas na tabela verdade. Para os circuitos mais complexos, nos quais não era possível testar todas as combinações possíveis, foram aplicados testes específicos de modo a comprovar o seu funcionamento de acordo com o seu *design* conceitual.

4 ANÁLISE E RESULTADOS

Em uma abordagem *top-down* este capítulo apresenta a análise do roteador nanoeletrônico, seguido dos circuitos que o compõe. Também serão apresentados os resultados de suas simulações para a sua validação. No final são abordadas as questões de área e potência dissipada do roteador e é feito uma comparação com um roteador implementado com a tecnologia CMOS.

4.1 ROTEADOR

Por ser voltado para a topologia *Mesh*, o roteador desenvolvido possui 5 entradas e 5 saídas. O seu circuito completo é muito grande e um *snapshot* através do *software* LTspice produz uma imagem não informativa. Assim sendo o circuito completo do roteador é apresentado em formato de diagrama de blocos, apresentado na figura 4.1. Em cada entrada existe um conjunto EB + PISO para cada uma das 5 saídas, o que significa que cada entrada possui um total de 5 EBs e 5 PISOs. O bloco "Banco de EBs e PISOs"na figura 4.1 engloba esses 5 EBs e seus 5 PISOs correspondentes, um para cada endereçamento de saída. Isso significa que cada entrada é capaz de armazenar 8 palavras para uma mesma saída, ou ainda, 40 palavras ao todo (40 bytes).

Um pacote de dados é constituido por uma palavra de 8 bits, dos quais os 3 MSB contém o endereçamento de destino do pacote. Um pacote que chega em qualquer uma das entradas é inicialmente processado pelo SRwOR, onde os seus bits são paralelizados e o endereçamento é extraído. Ao final deste processo, que dura 8 períodos de *clock*, o endereçamento é utilizado pelo DEMUX, que seleciona a saída correta. O pacote é então enviado para o EB correspondente. Os EBs são responsáveis por fazer o armazenamento temporário dos dados, e executam o protocolo *ready/valid handshake*. Assim que um pacote é armazenado no EB, este envia uma solicitação de uso do recurso de saída ao árbitro RoR da sua saída correspondente. Quando sua solicitação é atendida, o pacote de dados é novamente serializado através do PISO de 8 bits e é finalmente expedido pelo enlace de saída.

As próximas seções farão uma análise individual de cada componente do roteador. Também serão apresentadas duas simulações do roteador.

4.2 REGISTRADOR DE DESLOCAMENTO DE 8 BITS COM REGISTRADOR DE SAÍDA

Todos os quatro tipos de registradores de deslocamento e os registradores de armazenamento possuem arquiteturas muito bem descritas nos livros clássicos. O SRwOR porém não é normalmente descrito nos livros mais comuns. Sua arquitetura é bastante simples, e é apresentada na

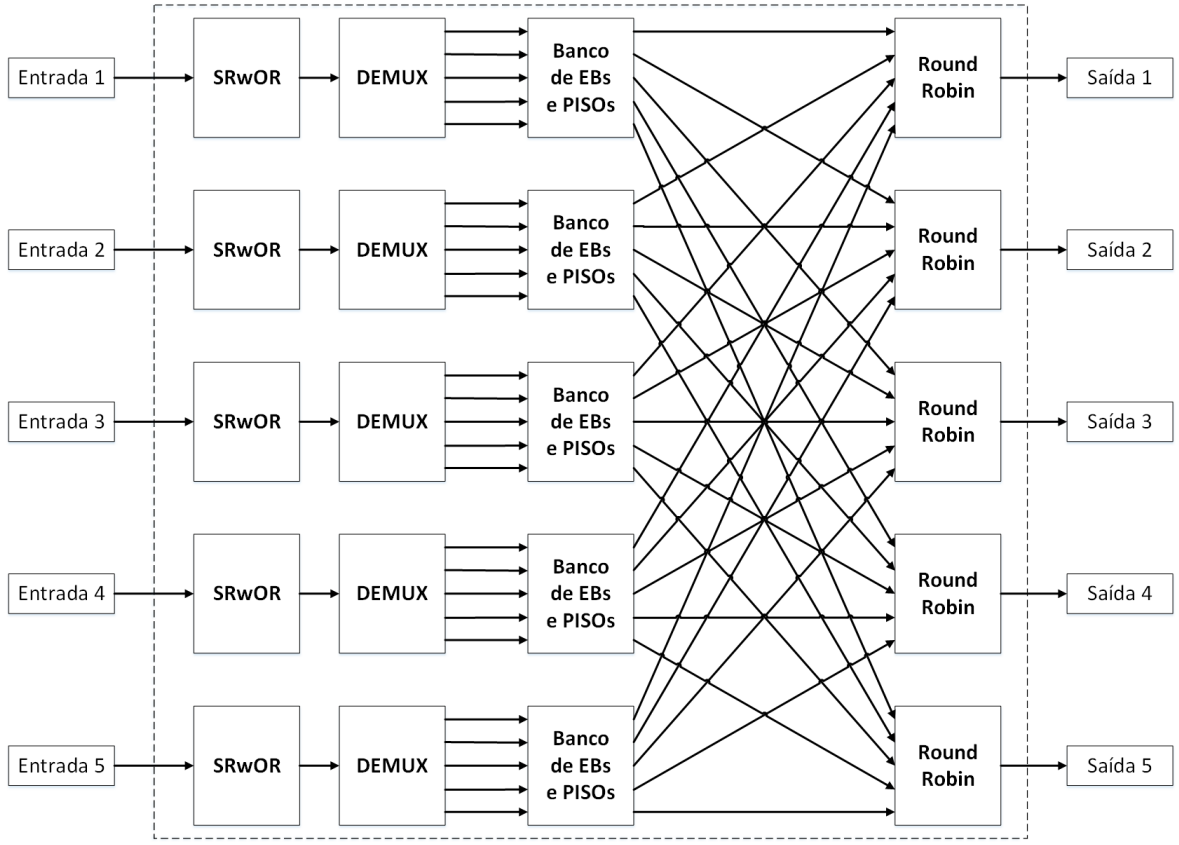

Figura 4.1: Esquemático do roteador completo

figura 4.2 [26]. Na coluna da esquerda encontra-se um SIPO de 8 bits com uma saída serial extra, e na coluna da direita encontra-se um registrador de armazenamento de 8 bits. A saída paralela do SIPO alimenta os bits do registrador de armazenamento, que por sua vez oferece uma saída paralela para cada um de seus bits. O bit POUT7 corresponde ao bit mais significativo (MSB) da palavra na entrada do SRwOR. Os três primeiros MSB da palavra (POUT7, POUT6, POUT5) são, por definição, o endereço de destino da palavra. Cada um dos registradores apresenta sinal de *clear* assíncrono, e seus sinais de *clock* são independentes um do outro.

Para que uma palavra de 8 bits na entrada de dados D_{in} seja completamente paralelizada sem perda de bits é necessário que transcorram 8 períodos no *clock* do SR (SRCLK). O registrador de armazenamento deverá armazenar a saída do SR apenas quando todos os bits da palavra de entrada estiverem na posição correta. Portanto, 1 período no *clock* do registrador de armazenamento (RCLK) deverá corresponder a 8 períodos do SRCLK e ambos deverão estar sincronizados. Este resultado pode ser obtido através do uso de um divisor de frequências, como um contador por exemplo.

Na simulação do SRwOR deve-se gerar uma palavra de 8 bits na entrada de dados D_{in}. O SRCLK é o *clock* principal do circuito. Ele recebe o mesmo sinal do *clock* geral do roteador. Após 8 períodos do SRCLK, cada bit da palavra encontra-se em seu respectivo registrador. Como o RCLK (o *clock* do registrador de armazenamento) possui 1 período igual à 8 vezes o período de SRCLK, em 4 ms de simulação é possível ver os bits da palavra nas saídas correspondentes do

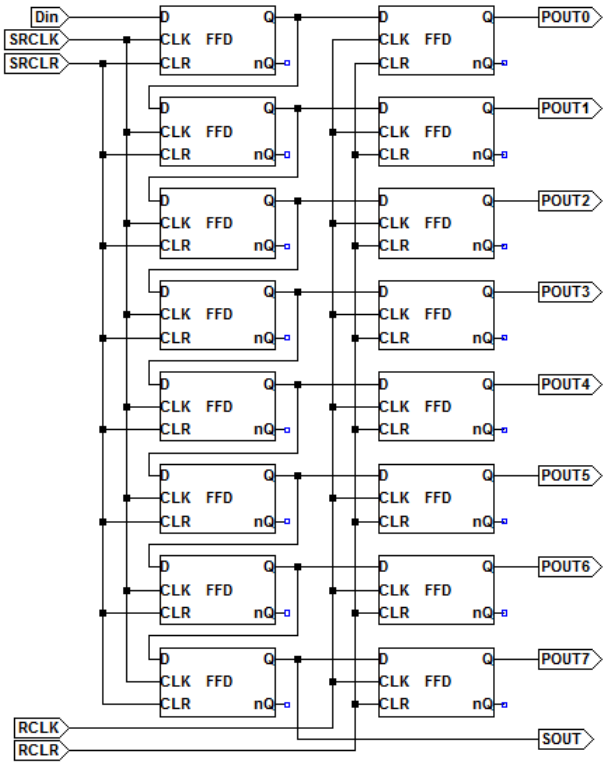

Figura 4.2: Esquemático do registrador de deslocamento de 8 bits com registrador de saída

SRwOR. A palara utilizada para a simulação encontra-se na tabela 4.1, e o resultado da simulação é apresentado na figura 4.3.

Tabela 4.1: Palavra de teste do SRwOR

b7	b6	b5	b4	b3	b2	b1	b0
1	0	0	1	0	1	1	1

4.3 MEMÓRIA SRAM

A memória SRAM desenvolvida neste trabalho pode ser considerada uma falsa *dual-port* SRAM, já que ela possui dois endereçamentos separados, porém apenas um controle de leitura, um de escrita e uma entrada de dados. Deste modo, cada um destes controles possui o seu próprio endereçamento separado, o que possibilita um controle individual e simultâneo dessas operações. Esta SRAM possui capacidade 8X8, i.e., 8 palavras de 8 bits cada. Seu bloco básico são *arrays* de 8 bits, como mostrado na figura 4.4.

Cada bit é determinado por um *flip-flop* D e é numerado como b_n, onde b7 é o bit mais significativo da palavra. Para que um novo dado seja escrito é necessário que o *array* seja selecionado no modo escrita com a entrada SW (*Select Write*) e que seja dado o comando de escrita através da entrada WEn (*Write Enable*). O *array* é regido por um sinal de *clock* e a escrita do novo dado ocorre apenas na subida do sinal de *clock* quando SW e WEn estiverem ativos. Para que o dado

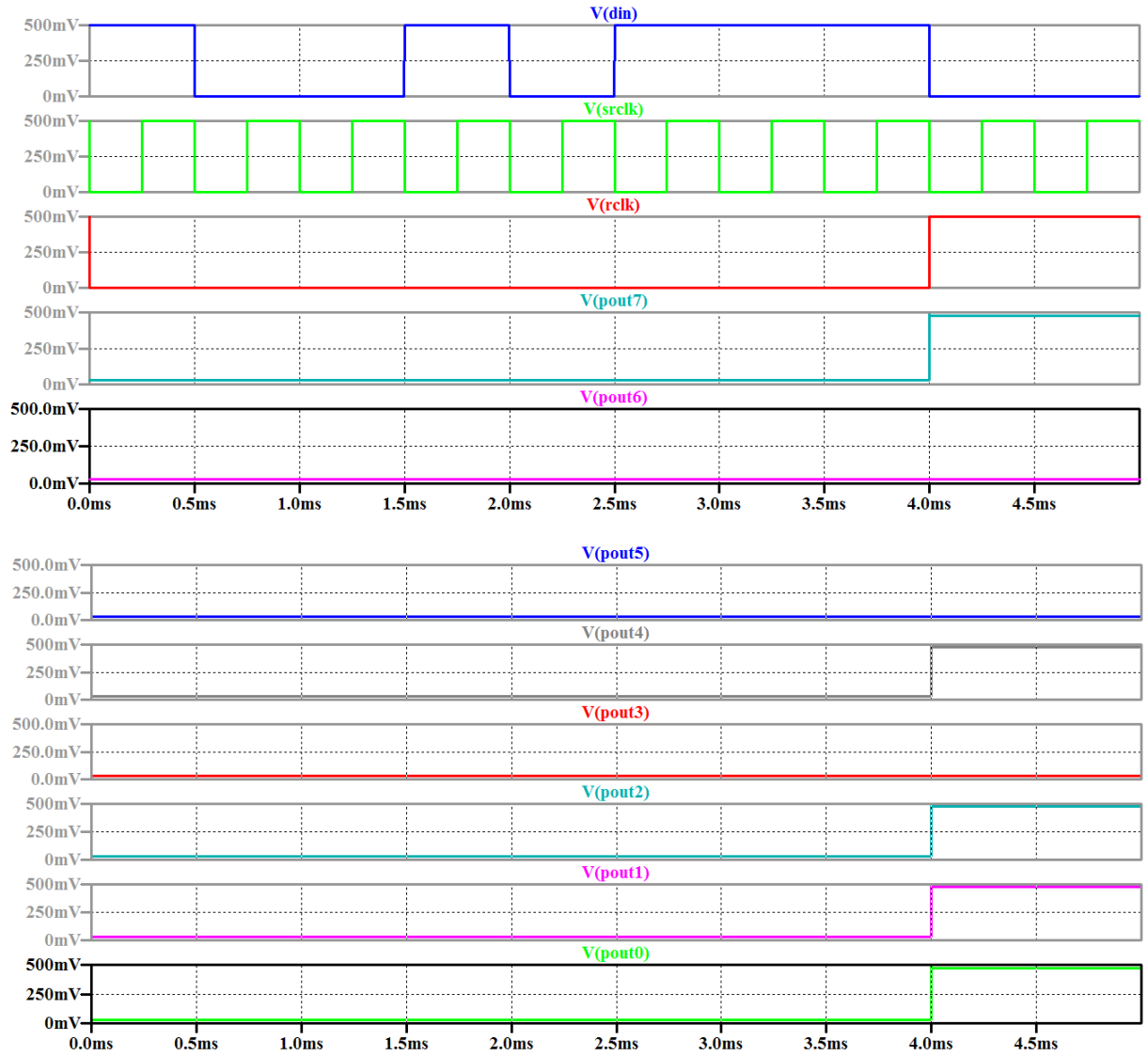

Figura 4.3: Simulação do SRwOR de 8 bits

armazenado no *array* seja lido é necessário que o *array* esteja selecionado no modo leitura através do sinal SR (*Select Read*) e que seja selecionado o comando de leitura na entrada REn (*Read Enable*).

Em um conjunto de 8 *arrays* a seleção de qual *array* deverá ser lido ou escrito é feita através de decodificadores 3:8. Os sinais de entrada S2, S1 e S0 do decodificador são os bits de endereçamento da SRAM e as suas saídas são conectadas ao sinal *select* de cada *arrray*. Devido às entradas de seleção separadas para escrita e leitura, são utilizados dois decodificadores, o que fornece um endereçamento para escrita e outro endereçamento para a leitura (figura 4.5).

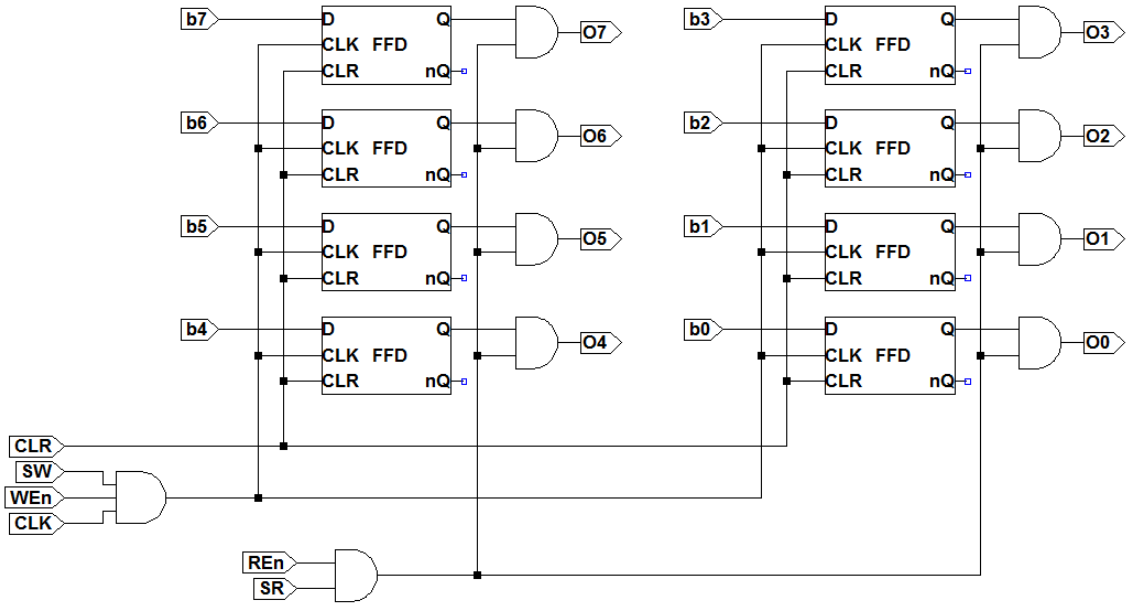

Figura 4.4: Esquemático de um *array* de 8 bits.

A simulação da SRAM apresenta o seu funcionamento quando os comandos de escrita e leitura ocorrem em períodos de *clock* diferentes e quando eles ocorrem no mesmo período. Neste teste duas palavras diferentes são gravadas nos endereços 3 e 7, sucessivamente (tabela 4.2). Os comandos de escrita e leitura são dados de acordo com a tabela 4.3. A figura 4.6 apresenta os resultados obtidos.

Tabela 4.2: Palavras de teste da memória SRAM

T (ms)	End.	b7	b6	b5	b4	b3	b2	b1	b0
0 - 1	3	1	0	0	1	0	1	1	1
1 - 1.5	7	1	1	0	1	1	0	0	1

Tabela 4.3: Comandos de escrita e leitura para o teste da SRAM

T (ms)	W	R
0 - .5	1	0
.5 - 1	0	1
1 - 1.5	1	1

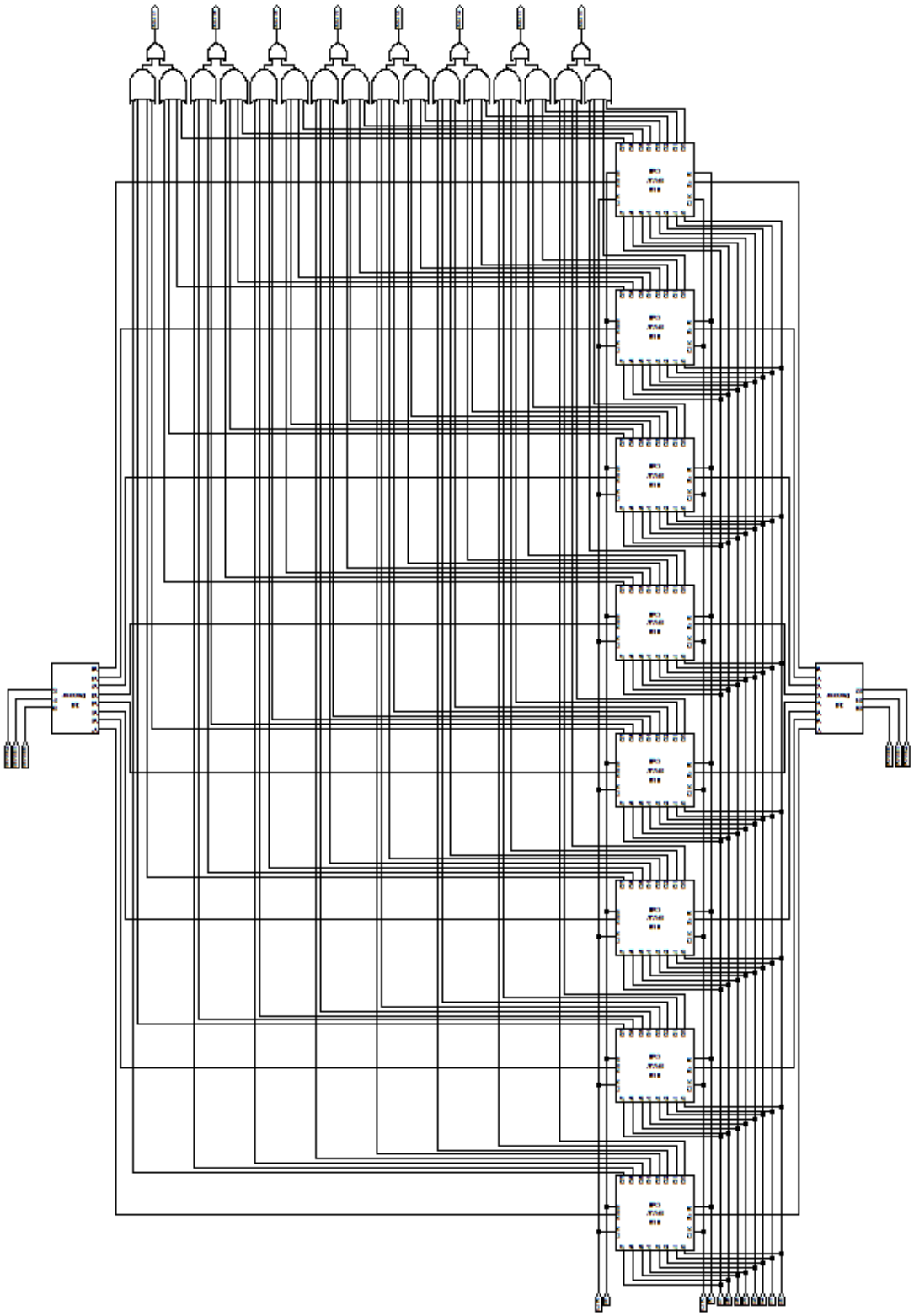

Figura 4.5: Esquemático da memória SRAM de 8 bytes

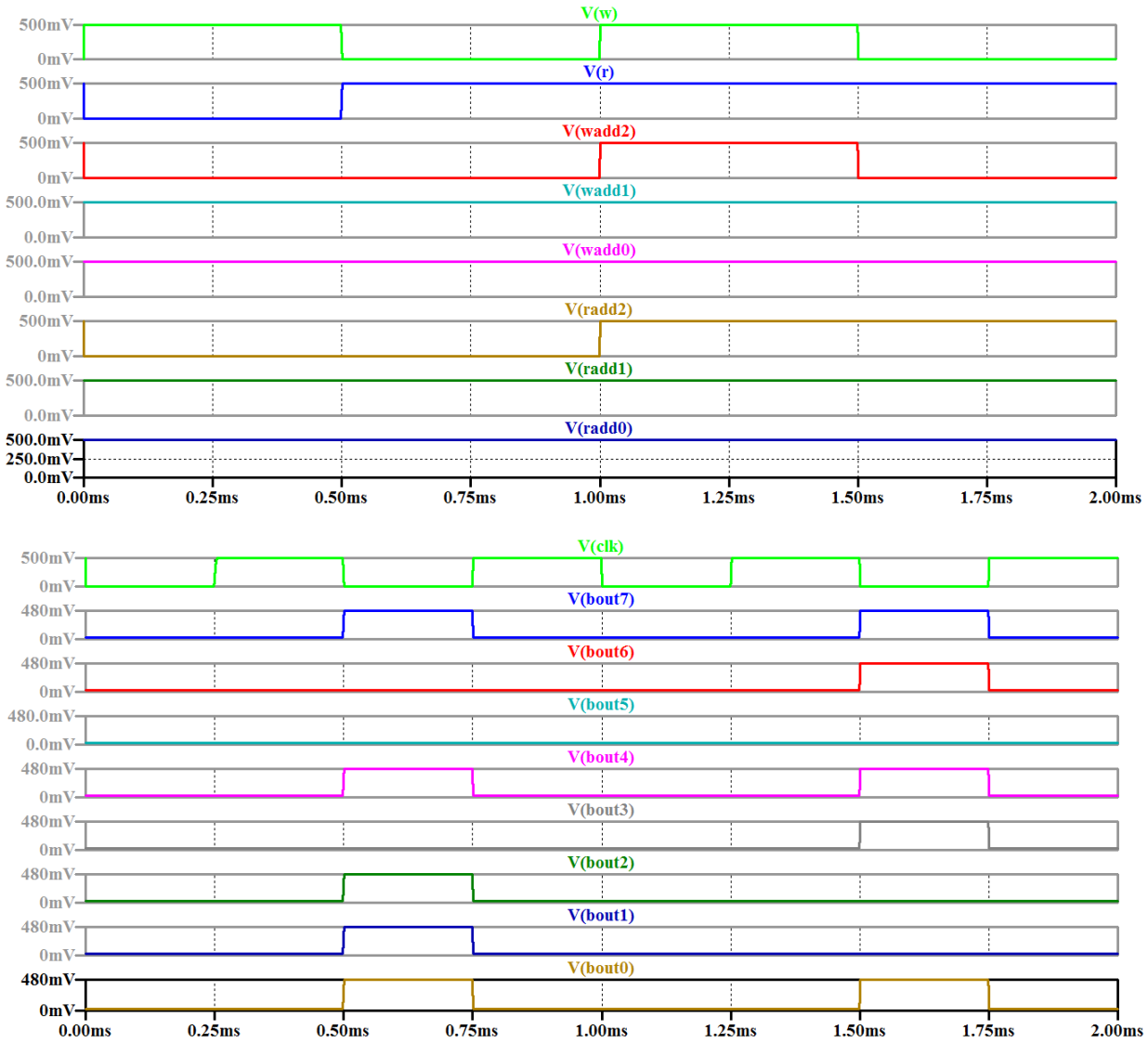

Figura 4.6: Simulação da memória SRAM de 8 bytes

4.4 FIFO

O FIFO é composto pelos seguintes elementos: memória SRAM, lógica de ponteiro e lógica de *flag*. A seguir serão mostrados a metodologia e desenvolvimento da lógica de ponteiro, da lógica de *flag* e do FIFO como uma unidade.

4.4.1 Lógica de ponteiro

Como se trata de um FIFO circular, o ponteiro deve pecorrer os endereços de 0 a 7 e retornar a 0 no final, como se fosse uma memória contínua. Além disso o ponteiro de escrita deve ser incrementado em uma posição cada vez que uma operação de escrita tiver sido executada. De modo similar o ponteiro de leitura deve ser incrementado ao final de cada operação de leitura.

Para atender essa finalidade, a lógica de ponteiro consiste em um contador binário de 3 bits

ativado pela borda de descida do *clock*. A entrada de *clock* do ponteiro de escrita é dada pelo sinal de *clock* geral do FIFO em conjunto com o sinal de *push*. A entrada de *clock* do ponteiro de leitura é dada pelo dobro do período do sinal de *clock* do sistema em conjunto com o sinal de *pop*. Ao dobrar o período do *clock* garante-se que os dados de saída do FIFO estarão disponíveis por um período completo do *clock* do sistema. Os ponteiros são apresentados na figura 4.7.

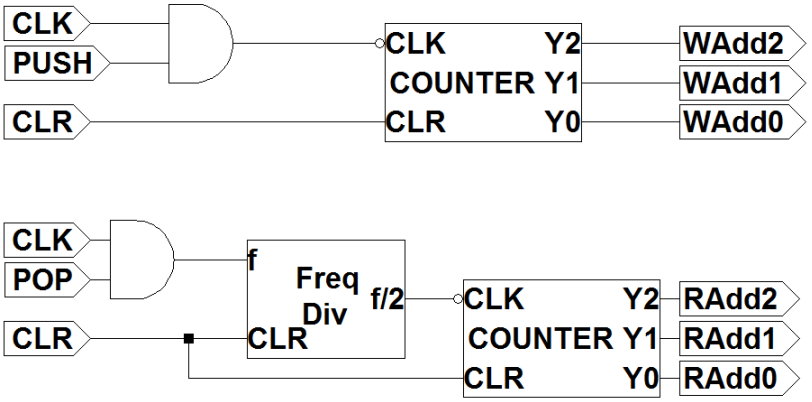

Figura 4.7: Lógica do ponteiro

4.4.2 Lógica de *flag*

Existem diversas maneiras de implementar a lógica de *flag*. O FIFO possui dois *flags* com funções opostas. A *flag full* indica que todas as posições da memória do FIFO estão preenchidas, enquanto a *flag empty* indica que não há nenhum dado (ou pelo menos nenhum dado novo) na memória do FIFO.

O *latch* SR pode ser utilizado como um buffer individual para cada *array* da memória, indicando se houve uma operação de escrita naquele *array* (função *SET*), ou se houve uma operação de leitura (função *RESET*), caso em que o dado já pode ser substituído por um novo. Como a memória SRAM possui 8 *arrays* são necessários 8 *latches* SR (figura 4.8). Os *latches* são endereçados através de dois DEMUX, um para escrita e um para leitura, assim como na memória SRAM. O DEMUX com o endereçamento de escrita (WAdd) é conectado ao *SET* de cada *latch*, enquanto o DEMUX de leitura (RAdd) é conectado ao *RESET*. Como operações de escrita e leitura podem ser realizadas ao mesmo tempo na memória SRAM, pode ocorrer o caso em que elas aconteçam simultaneamente no mesmo endereço. Nesse caso ocorreria uma operação proibida no *latch* SR, o que poderia gerar instabilidade. Para contornar esse problema é dado prioridade à operação de *push*, i.e., operação de escrita sobre a de leitura. A *flag full* será ativada quando todos os *arrays* tiverem sido escritos, mas não tiverem sido lidos, e a *flag empty* será ativada quando todos os *arrays* tiverem sido lidos ou ainda não tiverem sido escritos.

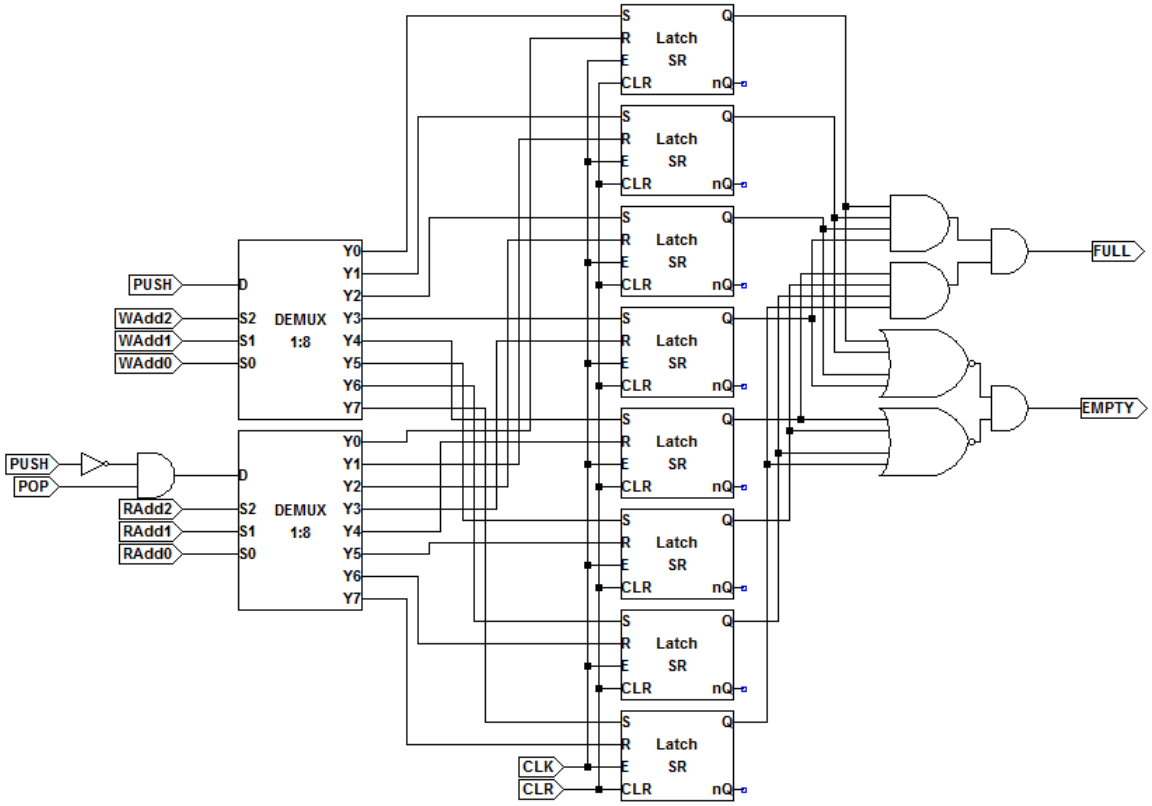

Figura 4.8: Lógica da *flag*

4.4.3 FIFO

O FIFO completo possui um sinal de *clock* e um sinal de *clear* universais. As lógicas de ponteiro de escrita e leitura são conectadas diretamente aos endereços de escrita (WAdd) e leitura (RAdd) respectivamente. Os mesmos ponteiros também são conectados aos endereços correspondentes na lógica de *flag*. Os bits de entrada são numerados de b0 à b7 e os bits de saída são numerados de bOUT0 à bOUT7. Por fim, os sinais de *push* e *pop*, que já estão conectados na lógica de ponteiro são diretamente conectados aos comandos de escrita (W) e leitura (R) na SRAM, respectivamente, e aos comandos homônimos na lógica de *flag*. A figura 4.9 apresenta o esquemático do FIFO.

Serão apresentados três testes do FIFO. O primeiro e o segundo testes mostram as *flags full* e *empty* em funcionamento. O terceiro teste mostra as operações de *push* e *pop* para duas sequências de dados armazenados.

Ao iniciar as operações do circuito do FIFO nenhum *push* ou *pop* foi realizado ainda, i.e., o FIFO está vazio. Desse modo, no primeiro teste, o FIFO é mantido vazio e pode-se observar que a *flag empty* permanece ativada, enquanto a *flag full* permanece desativada (figura 4.10).

Para que o FIFO esteja cheio são necessários no mínimo 8 comandos de *push* coincidentes com a subida do *clock* e que nenhum *push* ocorra. Isso é demonstrado no segundo teste, onde observa-se que, ao iniciar as operações, o FIFO encontra-se vazio. Mas ao final, quando todas as

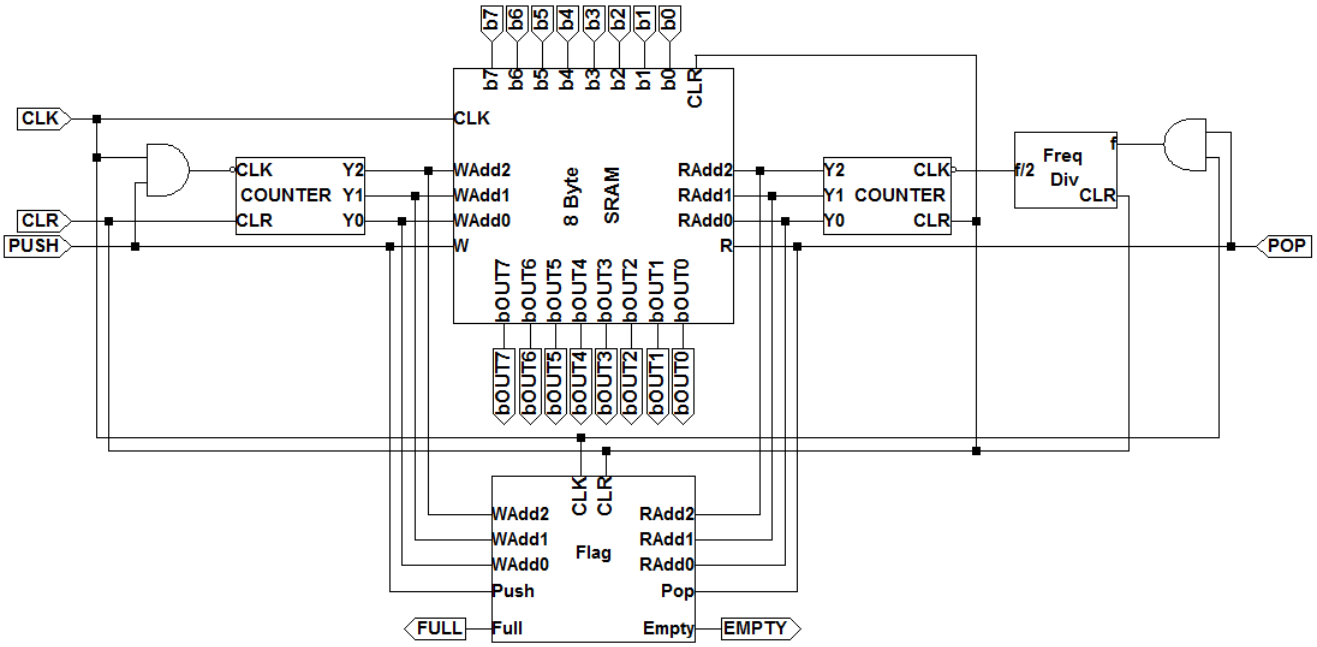

Figura 4.9: Esquemático do registrador FIFO

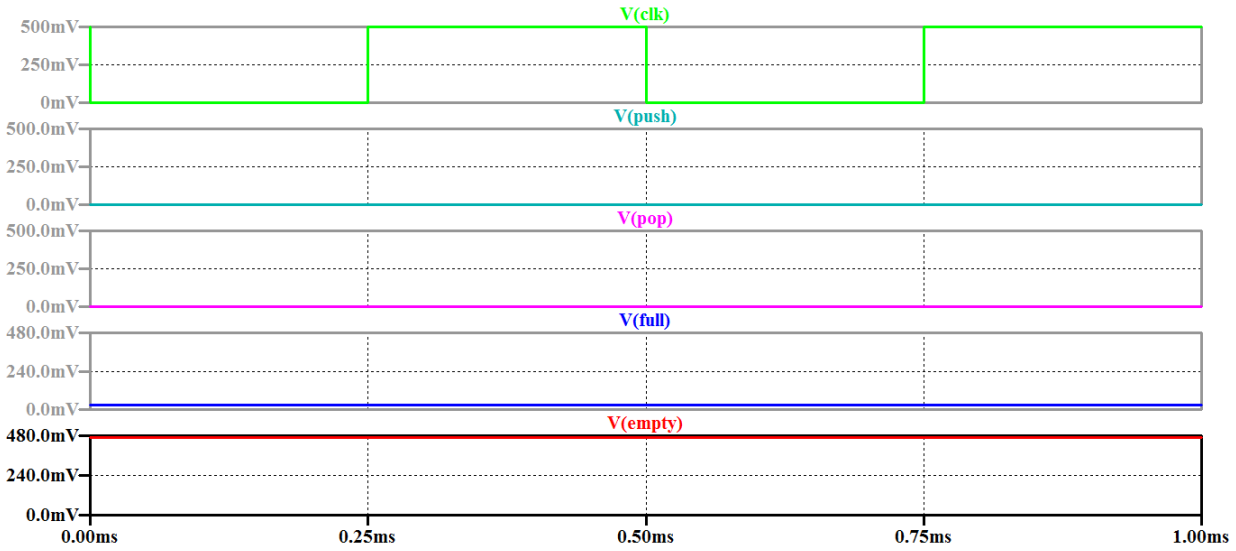

Figura 4.10: Simulação do FIFO com memória vazia e *flag empty* indicando esse *status*

posições na memória foram preenchidas, a *flag full* é ativada (figura 4.11).

No terceiro teste duas palavras são carregadas sequencialmente na memória do FIFO através da operação *push*, e em seguida elas são recuperadas através da operação *pop*. As palavras utiizadas são as mesmas da tabela 4.2, porém elas são carregadas nas posições 0 e 1 da SRAM respectivamente, dado que o FIFO é circular e sua sequência de endereçamento sempre inicia na posição 0. A tabela 4.4 mostra a sequência de carregamento dos sinais *push* e *pop* e dos bits das palavras na entrada do FIFO. A figura 4.12 apresenta a respectiva simulação do teste 3 do FIFO.

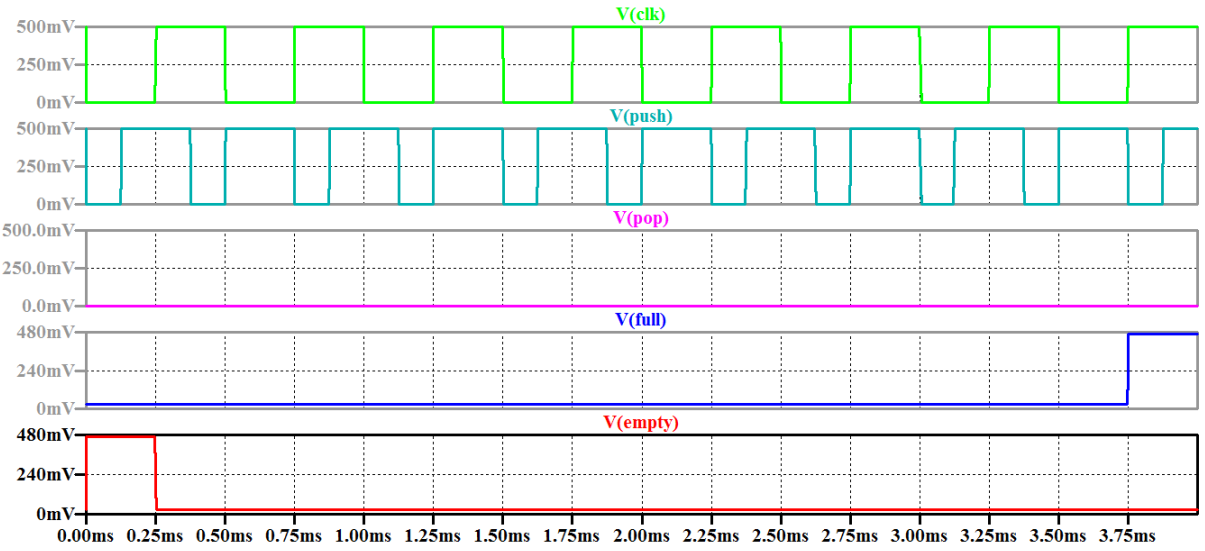

Figura 4.11: Simulação do FIFO: após preenchimento completo da memória a *flag full* é acionada

Tabela 4.4: Sinais do 3º teste do FIFO

T (ms)	*Push*	*Pop*	b7	b6	b5	b4	b3	b2	b1	b0	Palavra
0 - .75	1	0	1	0	0	1	0	1	1	1	1ª
.75 - 1.25	1	0	1	1	0	1	1	0	0	1	2ª
1.25 - 1.75	0	1	0	0	0	0	0	0	0	0	-
1.75 - 2.25	0	1	0	0	0	0	0	0	0	0	-

4.5 *BUFFER* ELÁSTICO

O EB pode ser facilmente obtido a partir de um FIFO (figura 4.13). Os sinais de *valid_in* e *valid_out* indicam que há dados disponíveis para serem recebidos ou enviados respectivamente. Já os sinais de *ready_out* e *ready_in* indicam se o FIFO está pronto para receber dados ou se o *receiver* está pronto para receber os dados, respectivamente. O FIFO não oferece nenhuma garantia sobre como serão gerenciados um *push* quando a fila estiver cheia ou um *pull* quando a fila estiver vazia. Este problema é resolvido no EB com a adição de portas AND no exterior do FIFO. Um *push* só é executado caso haja dados disponíveis para serem enviados por parte do *sender* e se houver espaço disponível na memória do FIFO. Um *pull* só é executado quando o FIFO tiver dados para enviar para o *receiver*, i.e., não estiver vazio, e o *receiver* indicar que está pronto para receber os dados.

As simulações do EB estão estruturadas de modo similiar às do FIFO. As duas primeiras simulações mostram como um FIFO cheio ou vazio afeta as respostas do protocolo *ready/valid handshake*. O terceiro teste ilustra o armazenamento e fornecimento de dados seguindo-se o protocolo.

Os dois primeiros testes foram realizados seguindo a mesma lógica do testes equivalentes para o FIFO. As figuras 4.14 e 4.15 apresentam os resultados. É possível verificar que quando o FIFO encontra-se vazio (figura 4.14) o sinal *valid_out* permanece desativado, o que indica para

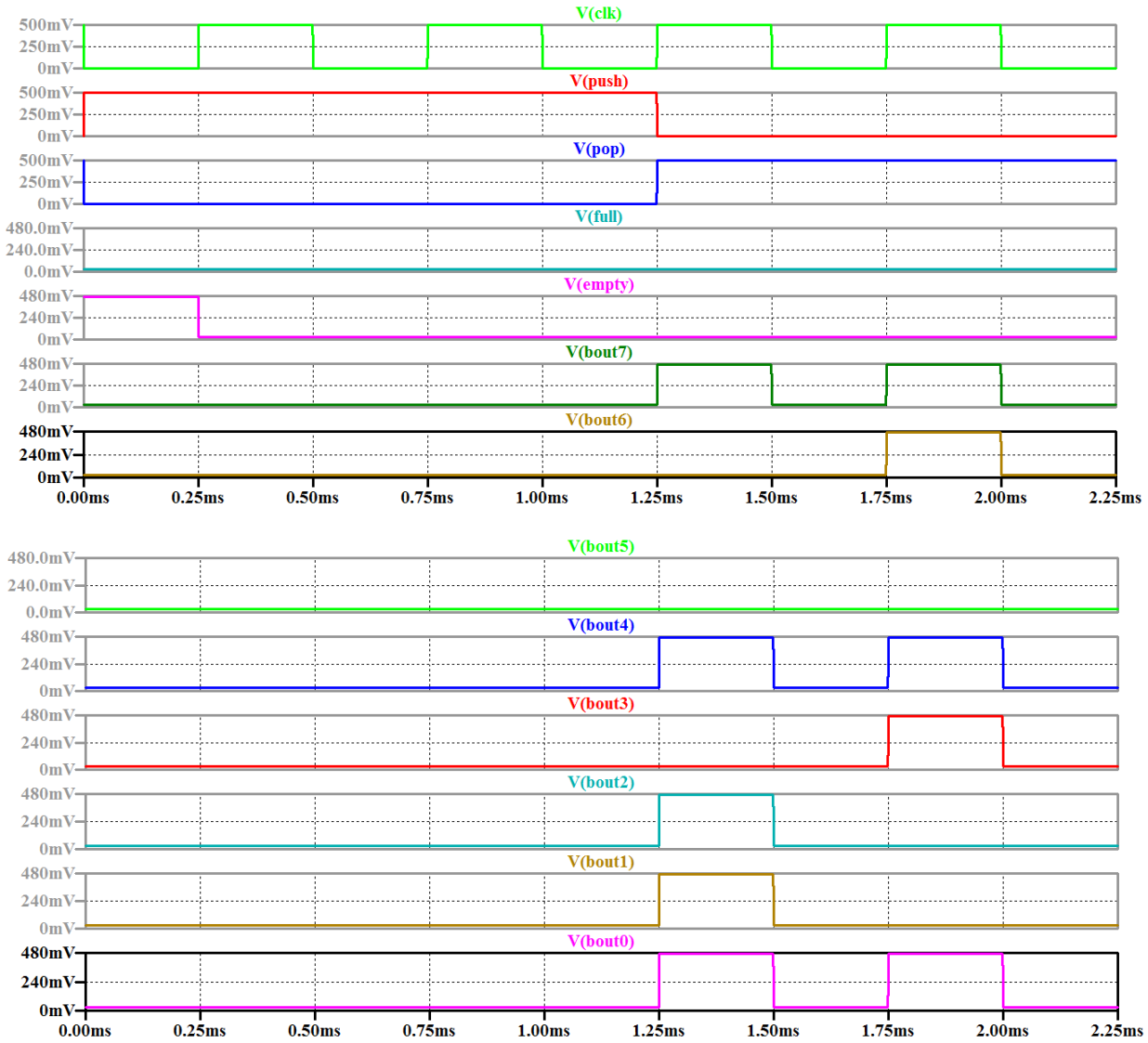

Figura 4.12: Simulação do 3º teste do FIFO

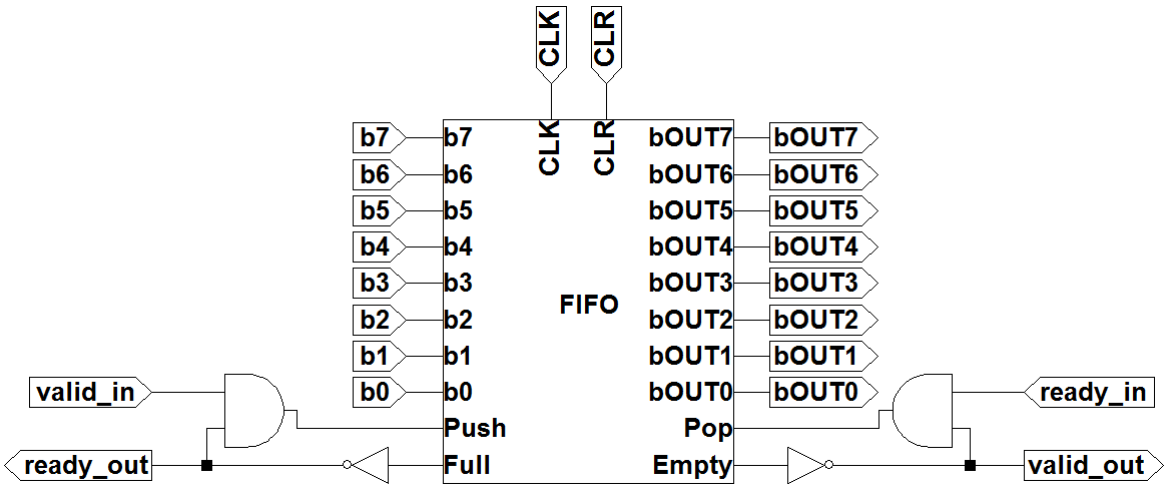

Figura 4.13: Esquemático do *Buffer* Elástico

o *receiver* que não há dados para serem enviados. Quando o FIFO torna-se cheio (figura 4.15) o sinal *ready_out* é desativado, o que indica para o *sender* que o *receiver* não pode receber novos dados no momento, e o *sender* deverá entar em estado de *hold*.

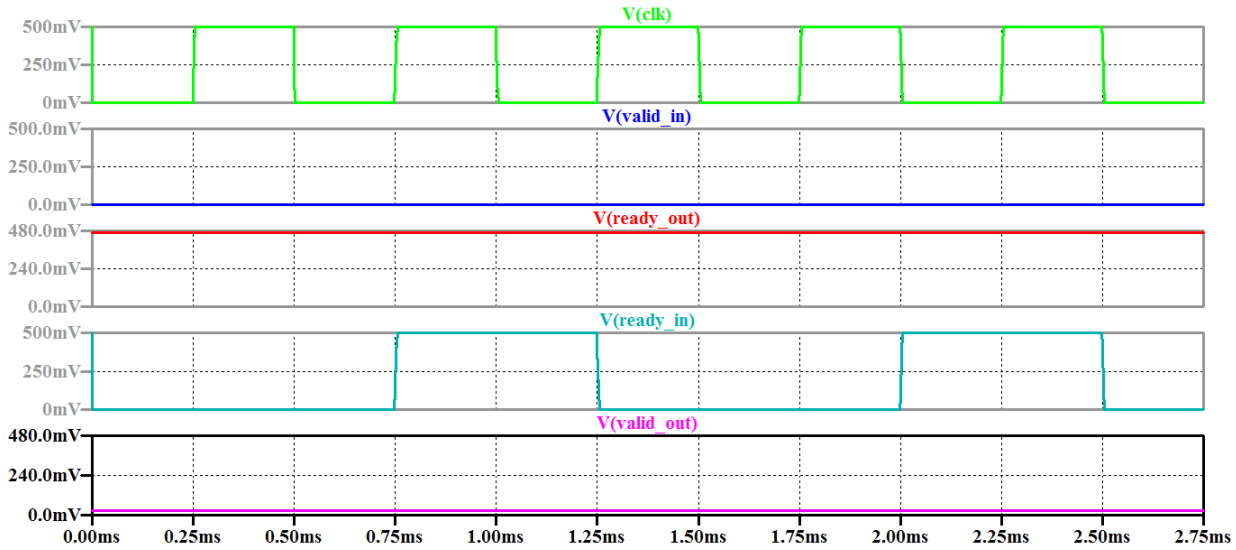

Figura 4.14: Simulação do EB com FIFO vazio

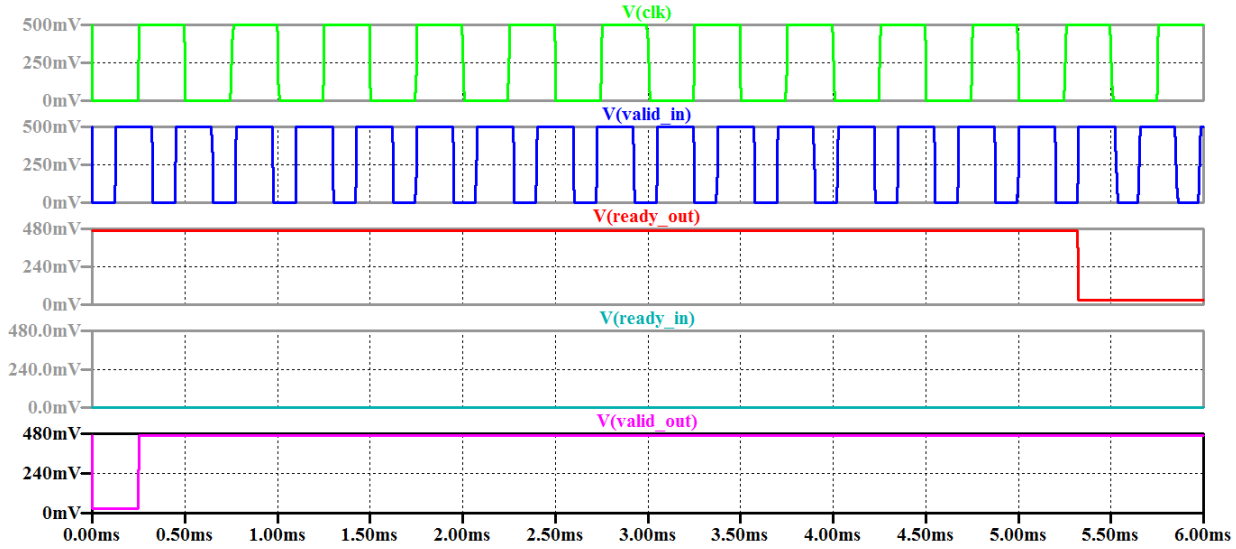

Figura 4.15: Simulação do EB com FIFO cheio

O terceiro teste é quase igual ao terceiro teste do FIFO, com a diferença de quando os pedidos de escrita e leitura são realizados através dos sinais *valid_in* e *ready_in*. Os sinais deste teste são apresentados na tabela 4.5 e os resultados encontram-se na figura 4.16

Tabela 4.5: Sinais do 3º teste de EB

T (ms)	*valid_in*	*ready_in*	b7	b6	b5	b4	b3	b2	b1	b0	Palavra
0 - .75	1	0	1	0	0	1	0	1	1	1	1ª
.75 - 1.25	0	1	0	0	0	0	0	0	0	0	-
1.25 - 2	1	0	1	1	0	1	1	0	0	1	2ª
2- 2.5	0	1	0	0	0	0	0	0	0	0	-

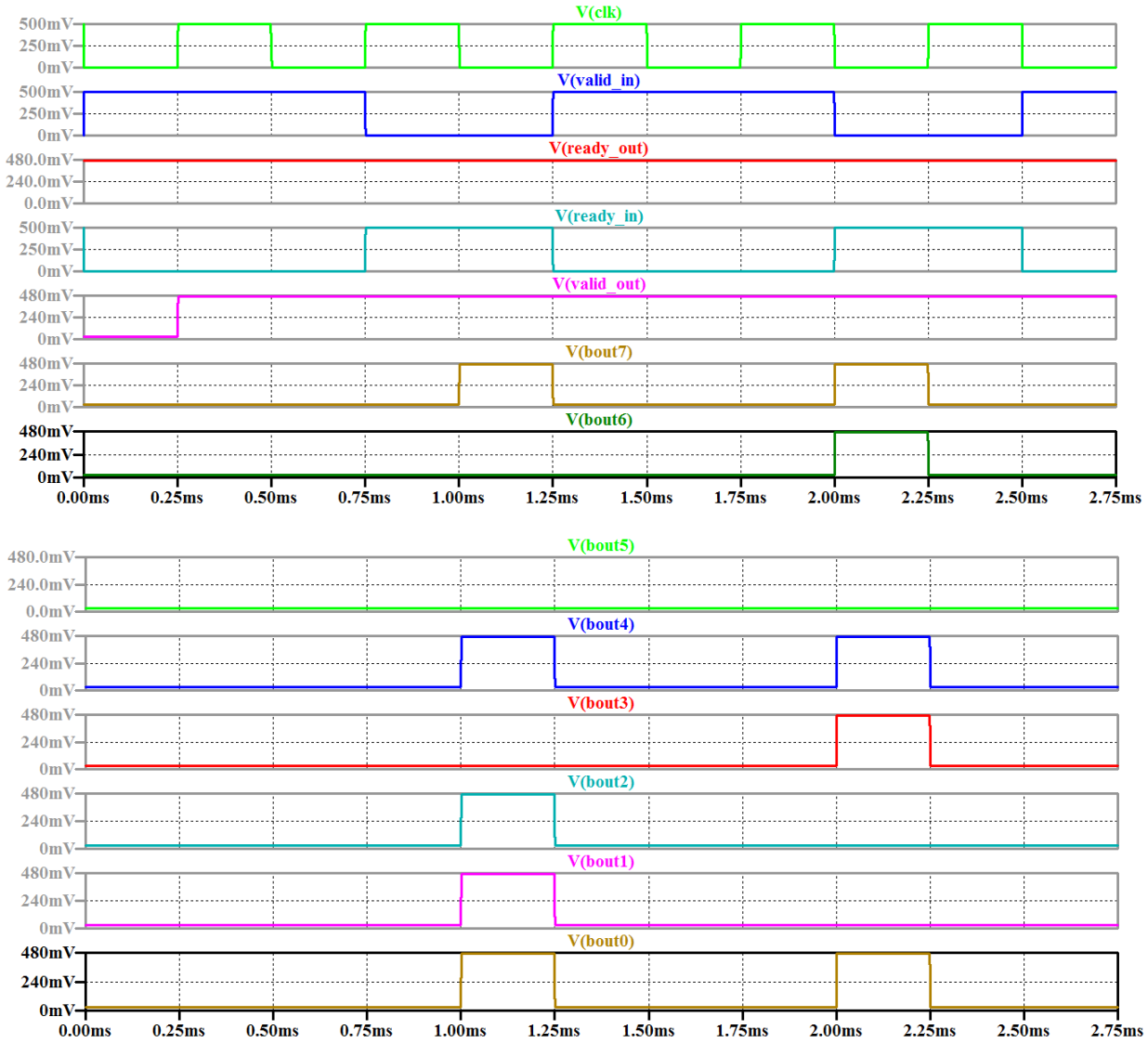

Figura 4.16: Simulação do 3º teste do EB

4.6 ÁRBITRO *ROUND-ROBIN*

A arquitetura do RoR é dividida em três partes: um árbitro inconsciente com prioriade variável iterativa (OA), que possui entradas para os bits de prioriadade, um gerador de prioridade *round-robin* e um circuito *grant-hold*, que mantém o *grant* concedido pelo período especificado pelo comando de *hold* e impede a concessão do recurso à outro agente durante este período. A figura 4.17 apresenta o esquemático de 1 bit do OA.

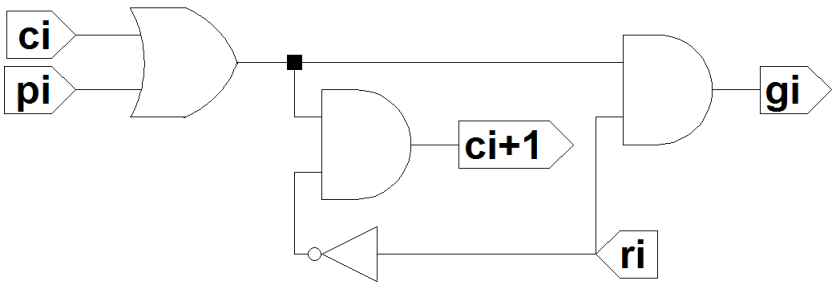

Figura 4.17: Esquemático de 1 bit do árbitro inconsciente com prioridade variável iterativa

O OA possui os bits *ri*, *pi* e *ci* como entrada e gera os bits *ci+1* e *gi* na saída. O bit *ri* é usado para indicar o pedido de uso do recurso. O bit *ci* indica que o recurso ainda não foi alocado para nenhum pedido de maior prioridade. Quando *pi* é ativado, o seu pedido recebe prioridade alta. Se *ri* e *ci* estiverem ativos e a respectiva célula possuir a maior prioridade, o pedido é concedido e o bit *ci+1* é desativado, indicando que o recurso foi concedido e não está mais disponível. Como o roteador possui 5 portas de entradas o árbitro completo deve possuir 5 bits. A figura 4.18 apresenta o OA completo.

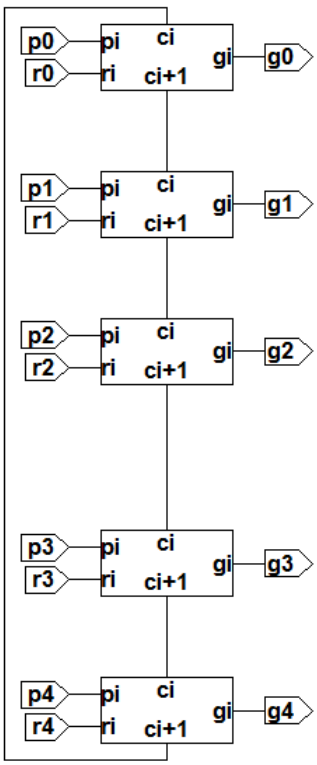

Figura 4.18: Esquemático do árbitro inconsciente com prioridade variável iterativa

A figura 4.19 apresenta o esquemático do gerador de prioridade *round-robin*. Quando um pedido é atendido (*gi*) o próximo bit de prioridade *pi+1* torna-se alto no próximo ciclo, o que faz com que o pedido mais próximo do pedido atendido receba prioridade máxima e o pedido atendido vá para o fim da fila.

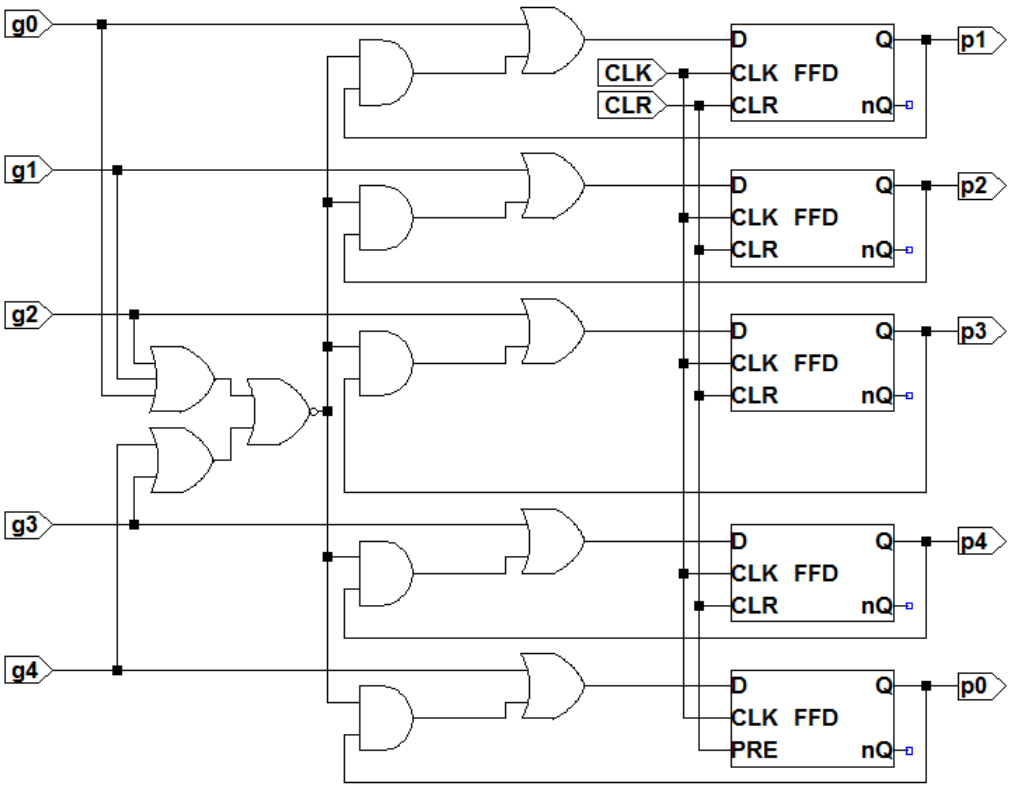

Figura 4.19: Esquemático do gerador de prioridade *Round-Robin*

Tabela 4.6: Pedidos e resultados do teste do RoR

T (ms)	*r2*	*r4*	*r1*	*g2*	*g4*	*g1*
0 - .5	1	0	0	1	0	0
.5 - 1	1	1	1	0	1	0
1 - 1.5	1	0	1	0	0	1
1.5 - 2	1	0	0	1	0	0

A figura 4.20 apresenta o *grant-hold* de 5 bits. Ele recebe como entrada os *grants* gerados pelo OA (*gci*) e o pedido de *hold* (*hi*), que especifica o total de períodos pelo qual o *grant* atual deve ser mantido, impedindo que novos pedidos de *grant* sejam atendidos durante esse intervalo. A figura 4.21 apresenta o árbitro de 5 bits completo.

Na simulação do RoR três agentes (*r1, r2* e *r4*) requisitam o uso do recurso. O agente *r2* é o primeiro a pedir o recurso e é prontamente atendido. Porém este agente faz um novo pedido no ciclo seguinte, juntamente com os outros agentes. Como seu pedido foi atendido no ciclo anterior, o agente *r2* passa agora para o final da final, e só é atendido novamente quando todos os outros agentes que requisitaram o recurso tiverem sido atendidos. A tabela 4.6 mostra a ordem dos pedidos e o resultado esperado, que é apresentado na figura 4.22. Durante esta simulação cada sinal de *hold* foi mantido apenas por um período de *clock*.

Se, ao invés de manter o *hold* por apenas um período de *clock* em todos os *requests*, for mantido o *hold* do agente 4 por dois períodos de *clock*, os próximos pedidos dos agentes 1 e 2 terão seu atendimento atrasado em 1 período relativo à simulação anterior. Esse resultado é

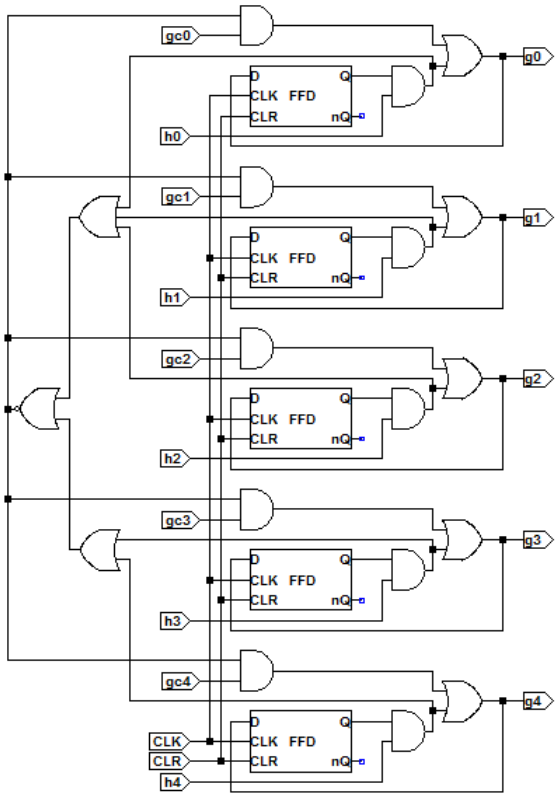

Figura 4.20: Esquemático do circuito de *Grant-Hold*

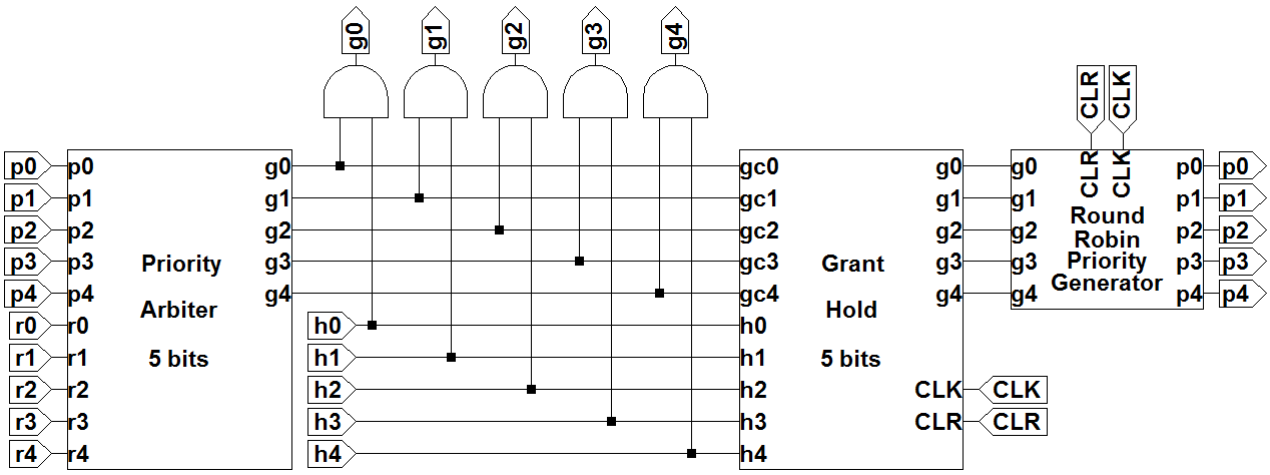

Figura 4.21: Esquemático do árbitro *Round-Robin*

apresentado na figura 4.23.

4.7 SIMULAÇÕES DO ROTEADOR

Para ilustrar o funcionamento do roteador completo serão apresentados dois testes. Em ambos os testes pacotes de dados são apresentados em uma determinada entrada, e é possível observar

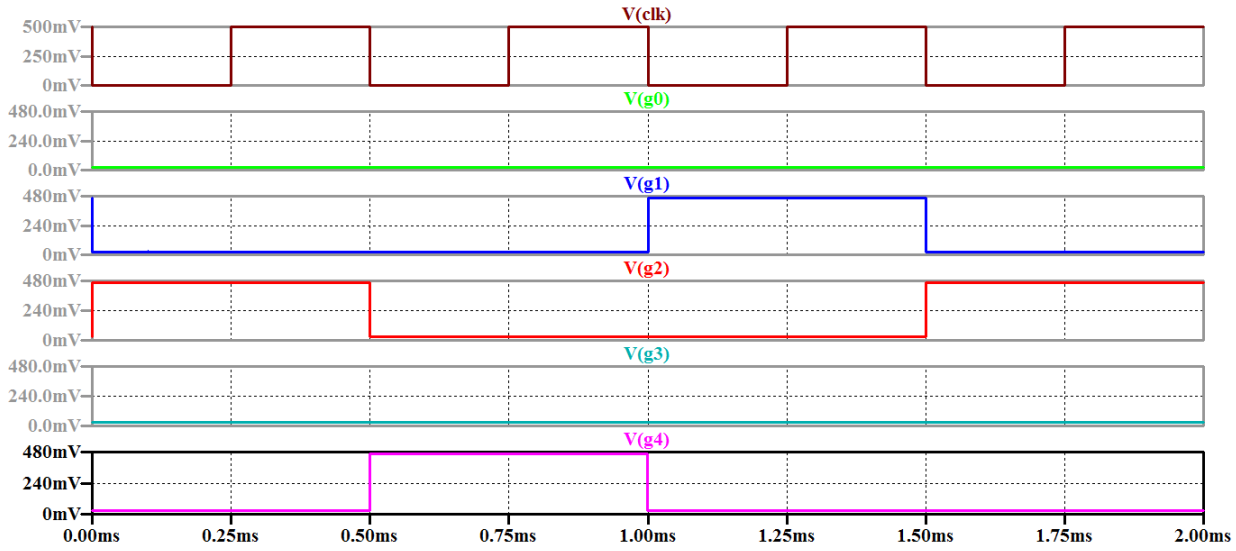

Figura 4.22: Simulação do árbitro *round-robin*

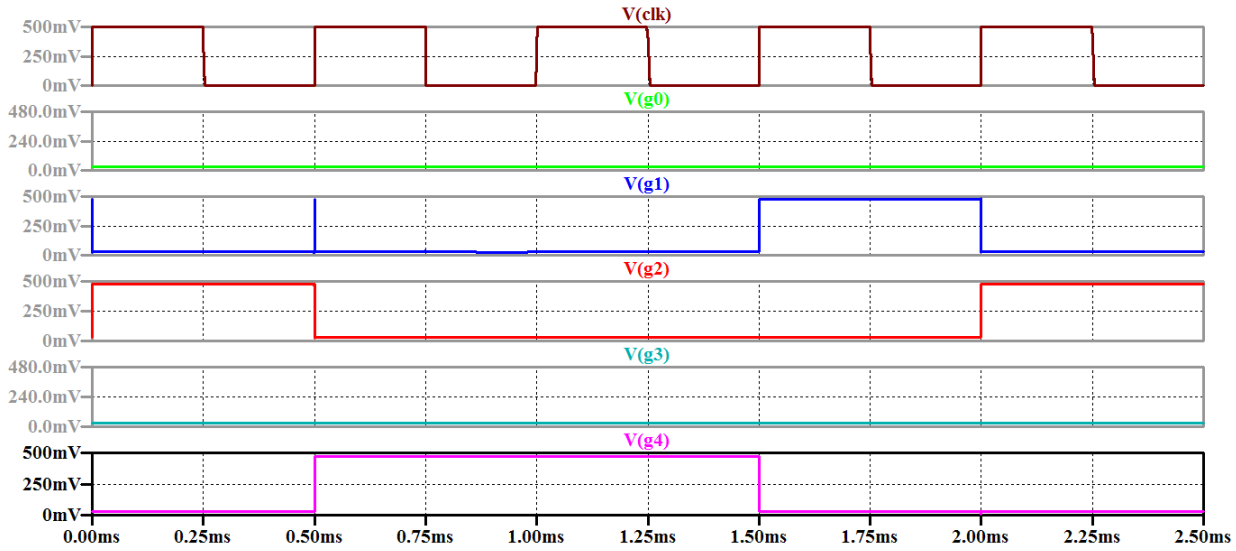

Figura 4.23: Simulação do árbitro *round-robin* com *hold* no agente 4

nos gráficos apresentados que, após o período de processamento necessário, o pacote de dados é apresentado corretamente na saída para a qual ele havia sido endereçado.

4.7.1 1º Teste do Roteador

Este primeiro teste apresenta o caso em que um único pacote de dados chega em uma entrada, e deve ser entregue a uma saída, sem concorrência de pacotes de outras entradas, de acordo com o endereçamento do pacote. A figura 4.24 apresenta o esquemático de 1 entrada conectada à 1 saída do roteador. Neste caso a entrada é a de número 0 e a saída também é de número 0.

Para validação da comunicação entre 1 entrada e 1 saída, um pacote de dados, ou palavra, (tabela 4.7) é enviado da entrada 0 para a saída 0. Como os três bits MSB da palavra são usados

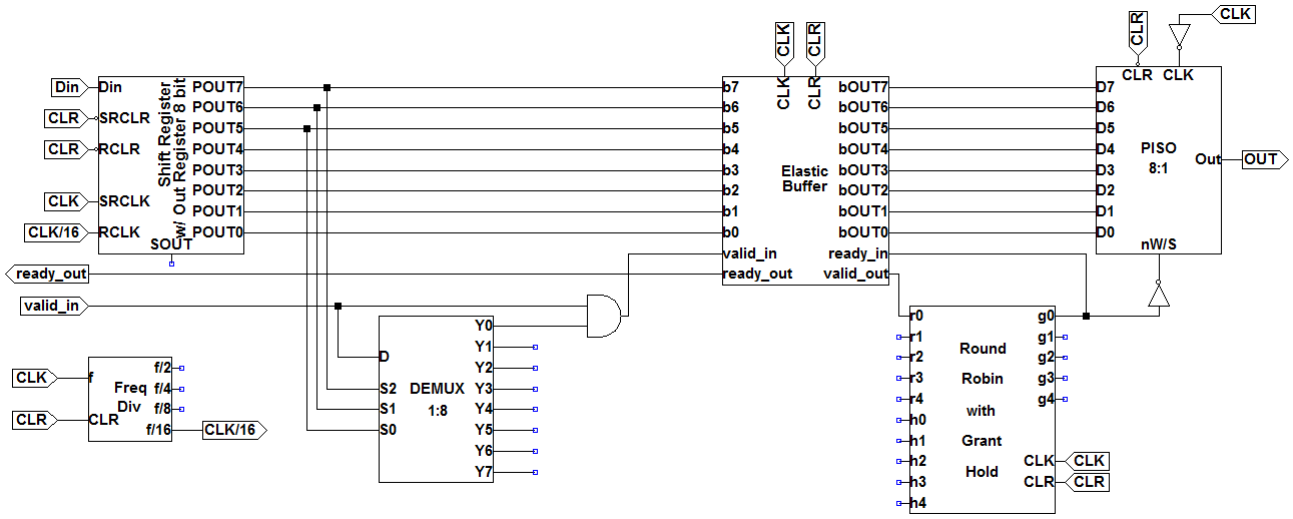

Figura 4.24: Esquemático de 1 entrada conectada à 1 saída do roteador

como endereçamento, os três primeiros bits da palavra de teste são 000 (tabela 4.7). A figura 4.25 apresenta o resultado obtido, no qual a mesma palavra de entrada (sinal Vdin) é apresentada na saída (sinal Vout) a partir de 4,50 ms, i.e., após o tempo necessário de processamento que inclui a paralelização e a posterior serialização do pacote, e execução do protocolo *ready/valid handshake* e da arbitragem.

Tabela 4.7: Palavra da 1ª simulação do roteador

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	1	1	0	1

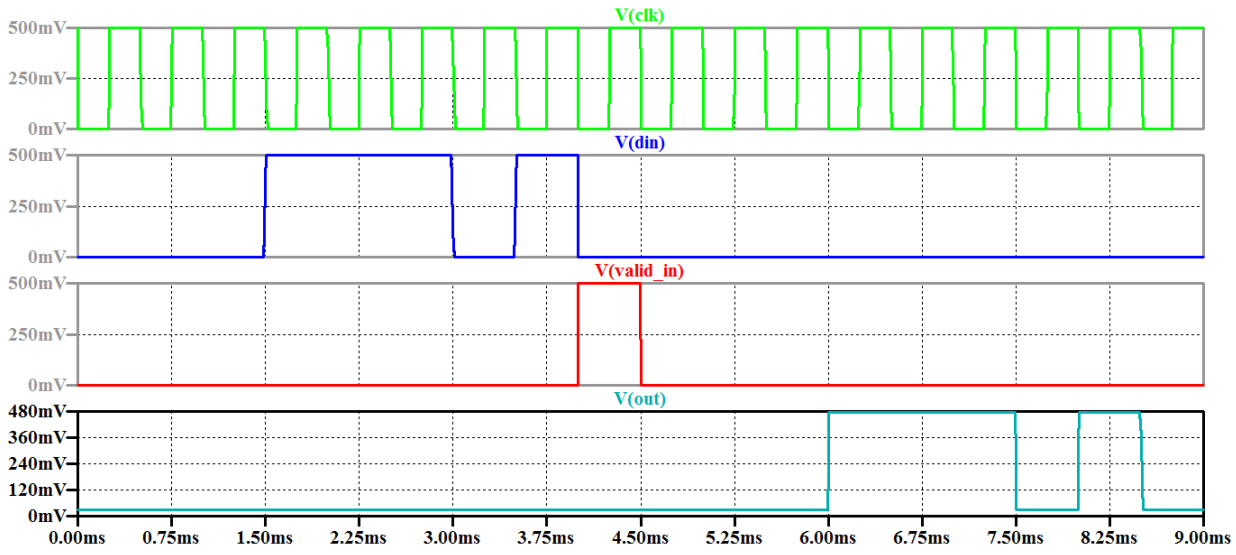

Figura 4.25: Simulação de 1 entrada conectada à saída 0 do roteador

4.7.2 2º Teste do Roteador

No segundo teste duas entradas (portas 1 e 4) requisitam ao mesmo tempo a porta de saída 1. A figura 4.26 apresenta o esquemático com as duas entradas e os EBs relativos à saída 1. Neste caso, como não havia sido feito nenhum pedido anteriormente e como os dois pedidos são executados ao mesmo tempo, o RoR atende os pedidos por ordem numérica. Desse modo a porta de entrada 1 será atendida antes da entrada 4.

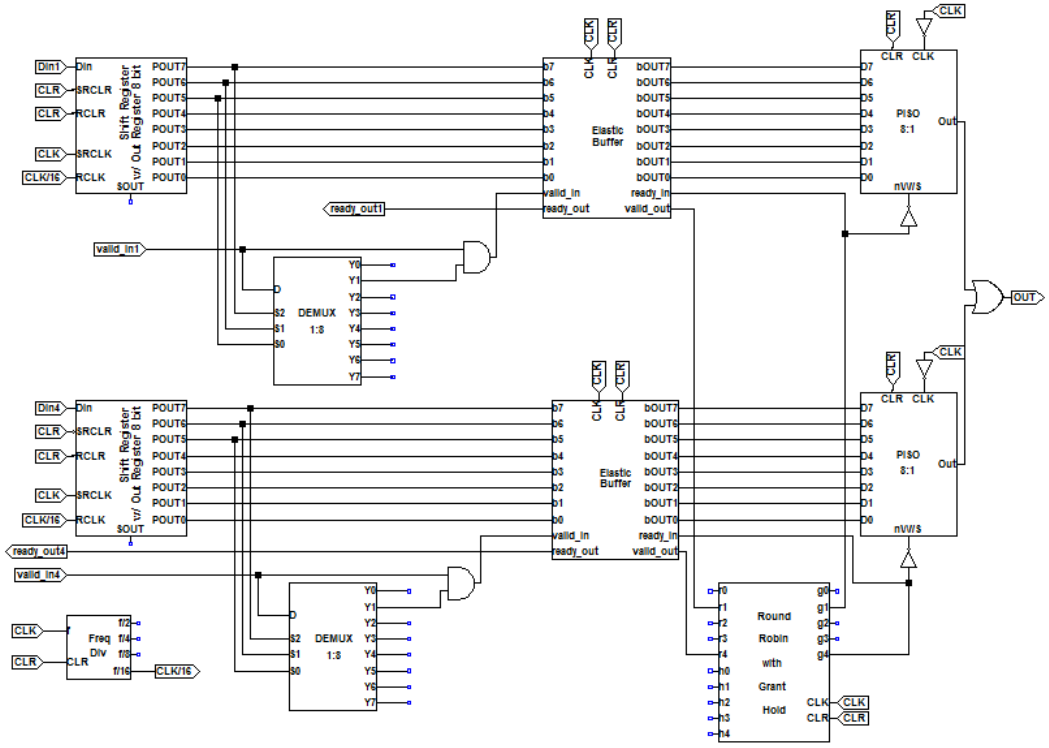

Figura 4.26: Esquemático de 2 entradas conectadas à 1 saída do roteador

A tabela 4.8 apresenta as palavras recebidas pelas portas de entrada 1 e 4 respectivamente. Como elas estão endereçadas para a saída 1 os três bits MSB das duas palavras são 001. Por fim a figura 4.27 apresenta o resultado da simulação, onde o pacote da entrada 1 (sinal Vdin1) é apresentado primeiro na saída (sinal Vout), a partir de 4,50 ms, e o pacote da entrada 4 (sinal Vdin4) é apresentado por último na saída,a partir de 8,50 ms.

Tabela 4.8: Palavras da 2ª simulação do roteador

	b7	b6	b5	b4	b3	b2	b1	b0
Din 1	0	0	1	1	0	1	0	1
Din 4	0	0	1	0	1	0	0	1

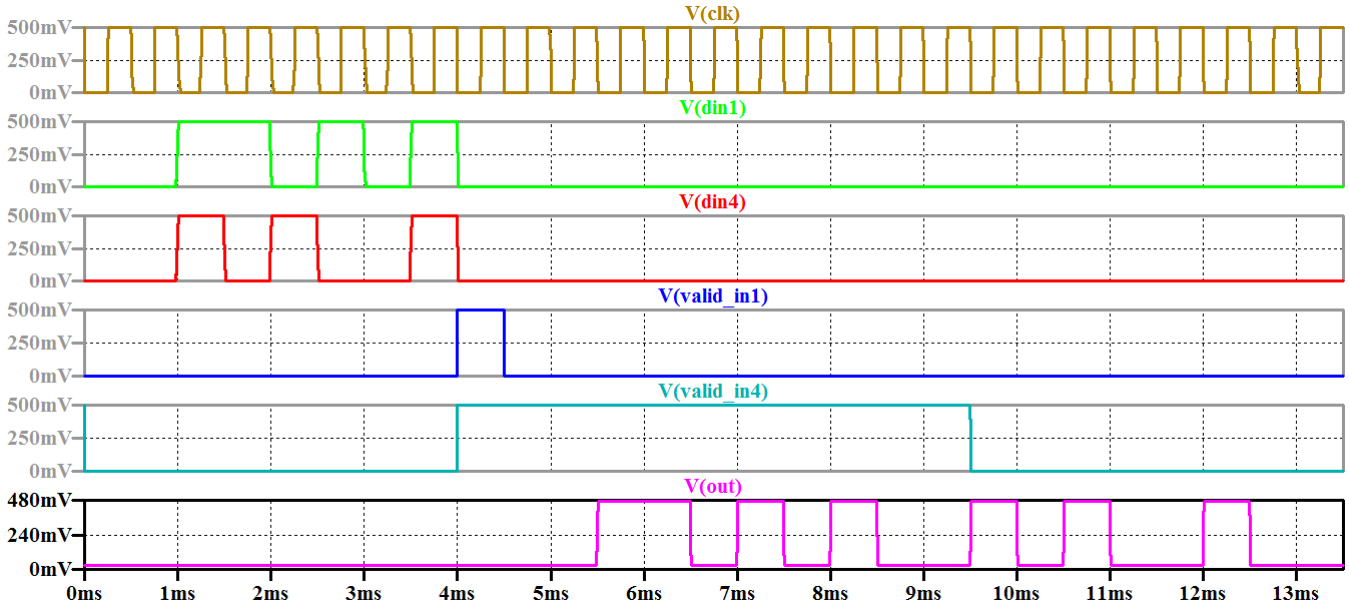

Figura 4.27: Simulação de 2 entradas conectadas à 1 saída do roteador

4.8 ANÁLISE DOS RESULTADOS

Dado que a arquitetura do roteador desenvolvido é constituída apenas de portas NAND nanoeletrônicas em seu elemento digital mais básico, é possível calcular a área e a potência dissipada pelo circuito a partir da área e potência de uma porta NAND nanoeletrônica (nanoNAND). Uma nanoNAND possui 172 nm^2 de área e dissipa 3.5 pW de potência considerando uma operação à temperatura de 300K, tensão máxima de 0.5 V e 1 GHz de frequência [27]. A tabela 4.9 apresenta a quantidade total de nanoNANDs em cada módulo, o total de módulos no circuito e a área e potência dissipada totais do circuito completo.

Tabela 4.9: Área e potência dos módulos do roteador

	Total de nanoNAND	Área	Potência Dissipada
nanoNAND	1	172 nm^2	3.5 pW
SRwOR	368	63296 nm^2	1288 pW
DEMUX	53	9116 nm^2	185.5 pW
EB	2318	398696 nm^2	8113 pW
PISO	209	35948 nm^2	731.5 pW
RoR	193	33196 nm^2	675.5 pW
Roteador 5x5	69619	$\sim$ 12 μm^2	$\sim$244 nW

A arquitetura digital é bastante flexível e torna fácil a modificação e/ou adição de módulos lógicos, como uma memória ou até mesmo portas de entrada ou saída. Apesar de esse roteador ter sido desenvolvido para topologia *Mesh* é possível adaptá-lo para outras topologias. Essa flexibilidade também permite que o próprio cerne dessa arquitetura seja modificado de SET para CMOS, bastando para isso apenas que os SETs da porta NAND sejam modificados para transis-

tores CMOS. A título de comparação a tabela 4.10 apresenta a área total e a potência dissipada caso a mesma arquitetura de roteador fosse construída a partir de transistores CMOS com canal de 22 nm [28]. Neste caso, a tensão V_{dd} de operação é de $0.9\,V$ a1 GHz de frequência.

Tabela 4.10: Área e potência do roteador CMOS

	Total de NAND	Área	Potência Dissipada
NAND CMOS	1	14800 nm^2	470 nW
SRwOR	368	5446400nm^2	172960 nW
DEMUX	53	784400 nm^2	24910 nW
EB	2318	34306400 nm^2	1089460 nW
PISO	209	3093200 nm^2	98230 nW
RoR	193	2856400nm^2	90710 nW
Roteador CMOS	69619	$\sim$ 1030 μm^2	$\sim$32.72 mW

Devido à tensão de operação, faz-se necessário o cálculo da nova potência dissipada pela nanoNAND com $V_{\text{dd}} = 0.9\,V$. A potência total dissipada é dada pela equação 4.1:

$$P_{total} = P_{estática} + P_{dinâmica} \tag{4.1}$$

A potência dinâmica é dada pela equação 4.2, onde C_L é a capacitância de carga do SET dada pela tabela 2.1.

$$P_{dinâmica} = f \times V_{dd}^2 \times C_L \tag{4.2}$$

$$P_{dinâmica} = 1\,G \times 0.9^2 \times 0.25\,a = 202.5\,pW \tag{4.3}$$

A potência estática é dada pela equação 4.4, onde $I_{\text{máx}}$ é a corrente máxima dissipada pelo circuito com V_{dd} = 0.9 V. O valor de I_{mx} é obtido através da simulação da porta NAND sob as condições já citadas.

$$P_{estática} = V_{dd} \times I_{máx} \tag{4.4}$$

$$P_{estática} = 0.9 \times 19.466\,p = 17.52\,pW \tag{4.5}$$

A potência total dissipada é então:

$$P_{total} = 202.5\,p + 17.52\,p = 220\,pW \tag{4.6}$$

Os novos valores de potência para o roteador nanoeletrônico são apresentados na tabela 4.11

A tabela 4.12 faz uma comparação de área e potência dissipada entre os roteadores de mesma

Tabela 4.11: Potência do roteador para V_{dd} = 0.9 V

	Total de NAND	Potência Dissipada
nanoNAND	1	220 pW
SRwOR	368	80960 pW
DEMUX	53	11660 pW
EB	2318	509960 pW
PISO	209	45980 pW
RoR	193	42460 pW
Roteador nano	69619	$\sim$15 μW

arquitetura com tecnologia SET e CMOS sob condições de operação iguais. Aqui fica evidenciado uma das maiores vantagens introduzidas pela tecnologia nano.

Tabela 4.12: Comparação entre os roteadores MOS e nanoeletrônico

	Área	Potência
Roteador nano	12 μm^2	15 μW
Roteador CMOS	1030 μm^2	32.72 mW

4.9 COMENTÁRIOS FINAIS

Um dos principais problemas encontrados durante o desenvolvimento deste trabalho foi a falta de ferramenta adequada para o desenvolvimento e simulação dos circuitos. Ferramentas voltadas para circuitos nanoeletrônicos, como o *SIMON* por exemplo, são rudimentares e dificultam a criação de grandes circuitos, não havendo opção de criação de macros. O *LTspice* permite a criação de macros e sua interface gráfica é simples de usar, facilitando a criação de circuitos com muitos elementos. Em contrapartida o *LTspice* não possui um modelo nativo para SET, o que é contornado em parte pelo uso de modelos criados pelos usuários. Esses modelos porém, mesmo quando otimizados, exigem muito processamento por parte *software* e tornam impossível a simulação de circuitos grandes, podendo até mesmo fazer com que o computador trave.

O outro problema encontrado deve-se ao modo como o projeto em si foi desenvolvido. A proposta inicial era de um roteador simples cujo comportamento era facilmente testado, evidenciando prontamente qualquer falha de projeto ou instabilidade. Ao final do projeto, porém, a proposta havia crescido imensamente e vários circuitos mais complexos haviam sido adicionados ao roteador. Como o projeto não foi orientado à testabilidade desde o início, muitas vezes falhas de projeto ou instabilidades eram encontradas apenas em simulações do conceito final, fazendo com que fosse necessário retornar aos módulos mais básicos e, por vezes, refazê-los completamente.

5 CONCLUSÃO

A lei de Moore tem incentivado, durante várias décadas, uma corrida pela miniaturização dos transistores por parte da indústria de semicondutores. Essa corrida trouxe vários benefícios ao desenvolvimento da microeletrônica, barateando os custos de produção, diminuindo o consumo de potência dos circuitos e aumentando a capacidade de integração de transistores em um *chip* de silício. A alta densidade de elementos processadores em aparelhos eletrônicos cada vez menores abriu espaço para a criação de sistemas-em-*chip*. A performance dos SoCs é limitado pelas suas interconexões. As NoCs tentam resolver esse problema aplicando conceitos de rede bem estabelecidos ao nível do *chip*. O elemento central da NoC é o roteador, sendo responsável pelo controle do fluxo de dados no SoC.

A contínua miniaturização dos transistores não poderá ser mantida indefinidamente, evidenciando a necessidade de novas tecnologias na área da eletrônica. A nanoeletrônica é uma tecnologia que propõe manter a miniaturização e resolver problemas de consumo de potência através da aplicação da física quântica em dispositivos eletrônicos. Dentre os dispositivos nanoeletrônicos que estão em desenvolvimento atualmente, o SET é muito promissor. Seu funcionamento baseado no controle do fluxo de um elétron faz com que o dispositivo tenha um baixo consumo de potência, enquanto sua configuração de três terminais (reminiscente do CMOS) torna o desenvolvimento de novas arquiteturas mais fácil de ser compreendido.

Este trabalho propôs e validou uma arquitetura de roteador para NoC com topologia *Mesh* completamente baseado na tecnologia SET. Ele implementa o protocolo básico *ready/valid handshake* para controle de fluxo e possui *buffers* e árbitros para lidar com possíveis contenções em seus canais de saída. Sua arquitetura digital o torna facilmente adaptável para o uso em outras topologias e a sua implementação de natureza modular possibilita a expansão do número de entradas e saídas e do total de *buffers* internos. Para a implementação do roteador foram propostas novas arquiteturas para uma memória SRAM e um registrador FIFO não encontradas na literatura. Além disso foi criada uma biblioteca de circuitos digitais para o LTspice, alguns dos quais propostos pela primeira vez.

Durante o desenvolvimento a crescente complexidade dos circuitos tornou a abordagem de teste inadequada para verificação de todas as possíveis instabilidades do projeto. Muitas vezes esses problemas eram evidenciados apenas quando o circuito em questão era inserido como módulo de um circuito maior, gerando atrasos na execução completa do projeto.

Alguns trabalhos futuros são:

- faz-se necessário adequar o projeto para a testabilidade e desenvolver ferramentas de software adequadas para desenvolver projetos de circuitos e sistemas digitais nanoeletrônicos
- o roteador desenvolvido está pronto para a inclusão na NoC atualmente em desenvolvimento

pelo grupo de nanoeletrônica do LDCI/UnB. Em um trabalho futuro será feita a integração dos dois projetos

- o roteador deve ser melhorado através da implementação de um algoritmo de roteamento como o XY, por exemplo.

REFERÊNCIAS BIBLIOGRÁFICAS

1 HANSON, G. W. *Fundamentals of nanoelectronics*. [S.l.]: Pearson/Prentice Hall, 2008. 385 p.

2 MOORE, G. E. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Newsletter*, v. 20, n. 3, p. 33–35, sep 2006.

3 *ITRS Reports - International Technology Roadmap for Semiconductors*. 2015. Disponível em: <http://www.itrs2.net/itrs-reports.html>.

4 DALLY, W. J.; TOWLES, B. P. *Principles and Practices of Interconnection Networks.* [S.l.]: Elsevier, 2003. 581 p.

5 DIMITRAKOPOULOS, G. *Microarchitecture of network-on-chip routers.* [S.l.]: SPRINGER-VERLAG NEW YORK, 2016.

6 PÊS, B. d. S.; GUIMARÃES, J. G.; BONFIM, M. J. d. C. A modified nanoelectronic spiking neuron model. *Journal of Computational Electronics*, Springer US, v. 16, n. 1, p. 98–105, mar 2017.

7 KHANNA, V. K. *Integrated nanoelectronics : nanoscale CMOS, post -CMOS and allied nanotechnologies*. [S.l.]: Springer Nature, 2016.

8 ISMAIL, R. B.; AHMADI, M. T.; ANWAR, S. *Advanced nanoelectronics*. [S.l.]: CRC Press, 2013. 424 p.

9 REZENDE, S. M. *Materiais e dispositivos eletrônicos.* [S.l.]: Editora Livraria da Física, 2004.

10 MITIN, V. V. V. V.; KOCHELAP, V. A. V. A.; STROSCIO, M. A. *Introduction to nanoelectronics : science, nanotechnology, engineering, and applications*. [S.l.]: Cambridge University Press, 2008. 329 p.

11 WALKER, J.; HALLIDAY, D.; RESNICK, R. *Fundamentals of physics*. 10. ed. [S.l.: s.n.], 2013. 1450 p.

12 JANTSCH, A.; TENHUNEN, H. *Networks on chip*. [S.l.]: Kluwer Academic Publishers, 2003. 303 p.

13 SWAPNA, S.; SWAIN, A. K.; MAHAPATRA, K. K. Design and analysis of five port router for network on chip. In: *2012 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics*. IEEE, 2012. p. 51–55.

14 HOLSMARK, R.; PALESI, M.; KUMAR, S. Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions. *Journal of Systems Architecture*, v. 54, n. 3, p. 427–440, 2008.

15 SERPANOS, D. N.; WOLF, T. *Architecture of network systems*. Morgan Kaufmann, 2011. 320 p.

16 WAKERLY, J. F. *Digital design : principles and practices*. [S.l.]: Prentice Hall, 2000. 949 p.

17 NOLT, J.; ROHATYN, D. A.; VARZI, A. C. *Schaum's outline of theory and problems of logic*. [S.l.]: McGraw Hill Professional, 2011. 322 p.

18 SHEFFER, H. M. A Set of Five Independent Postulates for Boolean Algebras, with Application to Logical Constants. *Transactions of the American Mathematical Society*, v. 14, n. 4, p. 481–488, oct 1913.

19 VENKATARATNAM, A.; GOEL, A. CMOS architectures for NOR & NAND logic gates using single electron transistors. *Proceedings of the 2005 NSTI Nanotechnology*, 2005.

20 LAGEWEG, C.; COTOFANA, S.; VASSILIADIS, S. Static buffered SET based logic gates. In: *Proceedings of the 2nd IEEE Conference on Nanotechnology*. IEEE, 2002. p. 491–494.

21 GEROUSIS, C. P.; GOODNICK, S. M.; POROD, W. Nanoelectronic single-electron transistor circuits and architectures. *International Journal of Circuit Theory and Applications*, v. 32, n. 5, p. 323–338, sep 2004.

22 KUPHALDT, T. R. *Vol. IV - Digital - Electronics Textbook*. 2007. 517 p. Disponível em: <https://www.allaboutcircuits.com/textbook/digital/>.

23 CÂMARA, B. O.; GUIMARÃES, J. G.; COSTA, J. C. Proposal of a router circuit based on nanoelectronic devices. In: *Advanced Manufacturing, Electronics and Microsystems TechConnect Briefs 2016*. [S.l.]: TechConnect, 2016. p. 183–187.

24 LINEAR. *Linear Technology - Design Simulation and Device Models*. 2017. Disponível em: <http://www.linear.com/designtools/software>.

25 TELLES, M. d. O.; GUIMARÃES, J. G. Single-electron shift-register circuit. *Microelectronics Journal*, v. 44, n. 4, p. 332–338, 2013.

26

27 SILVA, L.; GUIMARÃES, J. G. Performance Analysis of Single-electron NAND Gates. In: *ECS Transactions*. ECS, 2009. p. 311–318.

28 ZAIDI, A.; GARG, K.; VERMA, A.; RAHEJA, A. Design & Simulation of CMOS Inverter at Nanoscale beyond 22nm. *Int. J. Emerg. Sci. Eng*, 2013.

29 LIENTSCHNIG, G.; WEYMANN, I.; HADLEY, P. Simulating Hybrid Circuits of Single-Electron Transistors and Field-Effect Transistors. *Japanese Journal of Applied Physics*, IOP Publishing, v. 42, n. Part 1, No. 10, p. 6467–6472, oct 2003.

APÊNDICES

APÊNDICE A

Assim como em todo projeto digital, o roteador desenvolvido neste trabalho foi construído a partir dos elementos digitais mais básicos como portas lógicas, *latches* e *flip-flops*. Conforme mencionado na seção 2.3 do capítulo 2, o elemento mais básico deste projeto é a porta NAND nanoeletrônica de duas entradas. Todos os demais elementos foram desenvolvidos a partir desta porta usando um procedimento de projeto hierárquico. Embora a arquitetura digital destes elementos a partir de uma porta NAND seja bem estabelecida na literatura, deve-se ter em mente que a eletrônica envolvida não é a mesma utilizada na validação dos modelos clássicos. Torna-se necessário, portanto, validar esses modelos construídos com nanoeletrônica.

Este apêndice apresenta a arquitetura e os resultados simulacionais destes elementos para a sua respectiva validação. Todas as simulções foram realizadas no *software* LTspice IV com V_{dd} = 0.5 V e frequência de 4 kHz.

1 Porta NOT

A porta NOT é facilmente implementada a partir de uma única porta NAND de duas entradas. A função booleana NOT pode ser escrita como:

$$\overline{A + A} = \bar{A} \tag{1}$$

Portanto a arquitetura da porta NOT a partir da porta NAND é como apresentada na figura A.1. A simulação desta porta é apresentada na figura A.2.

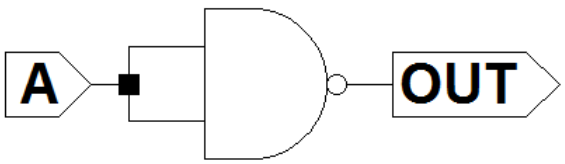

Figura A.1: Esquemático da porta NOT

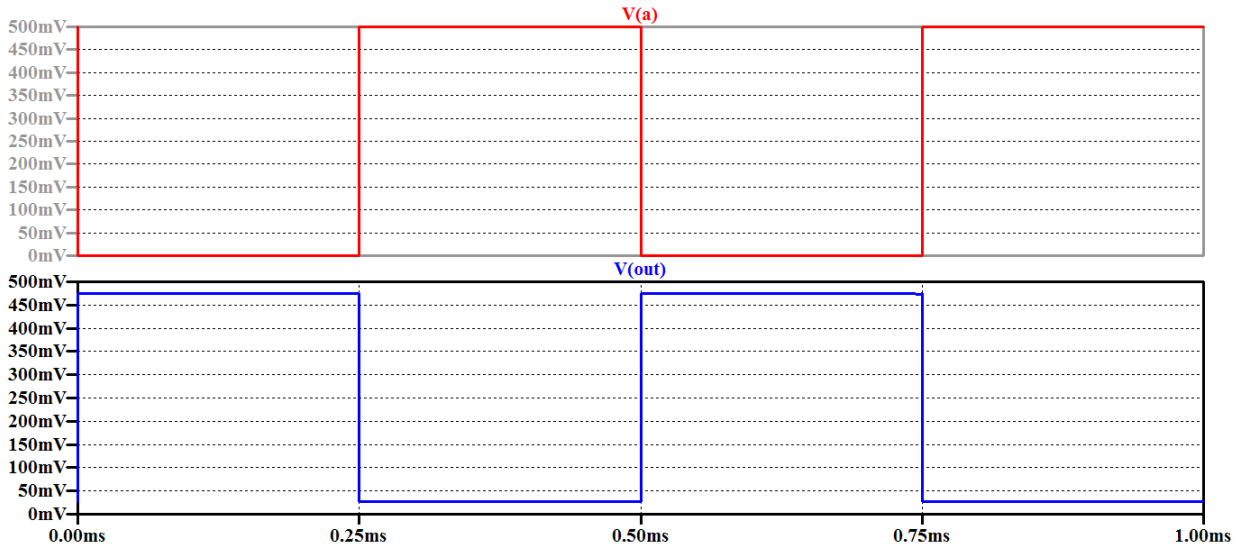

Figura A.2: Simulação da porta NOT

2 Porta OR

2.1 Porta OR de 2 entradas

A implementação da porta OR de duas entradas pode ser feita através de uma porta NAND de duas entradas e uma porta NOT. A função booleana da para essa porta pode então ser escrita como:

$$\overline{\overline{A}.\overline{B}} = A + B \tag{2}$$

As figuras A.3 e A.4 apresentam o esquemático da porta e os resultados da simulação respectivamente.

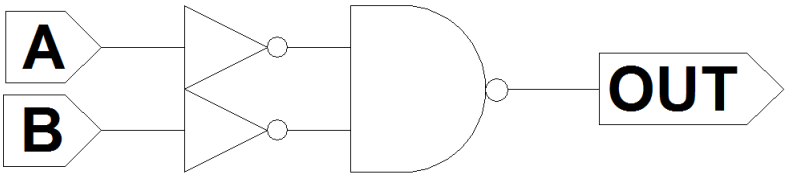

Figura A.3: Esquemático da porta OR de 2 entradas

2.2 Porta OR de 3 entradas

A porta OR de três entradas pode ser implementada a partir de duas portas OR de duas entradas com a seguinte função booleana:

$$(A + B) + C = A + B + C \tag{3}$$

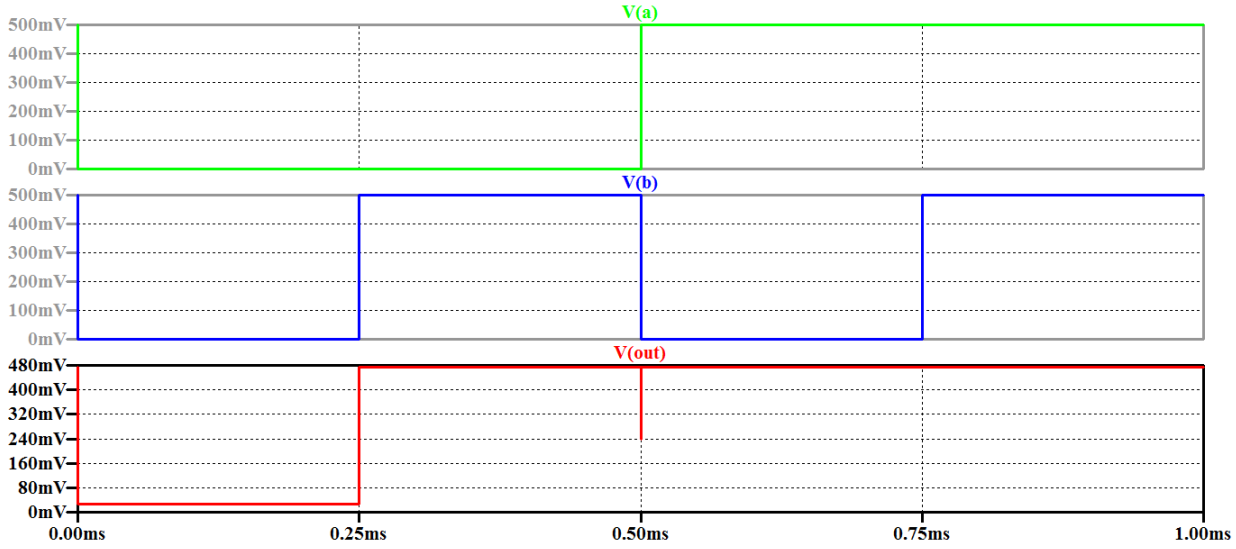

Figura A.4: Simulação da porta OR de 2 entradas

As figuras A.5 e A.6 apresentam o esquemático e os resultados.

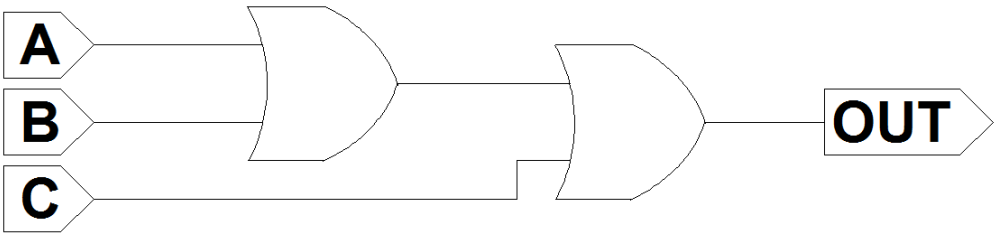

Figura A.5: Esquemático da porta OR de 3 entradas

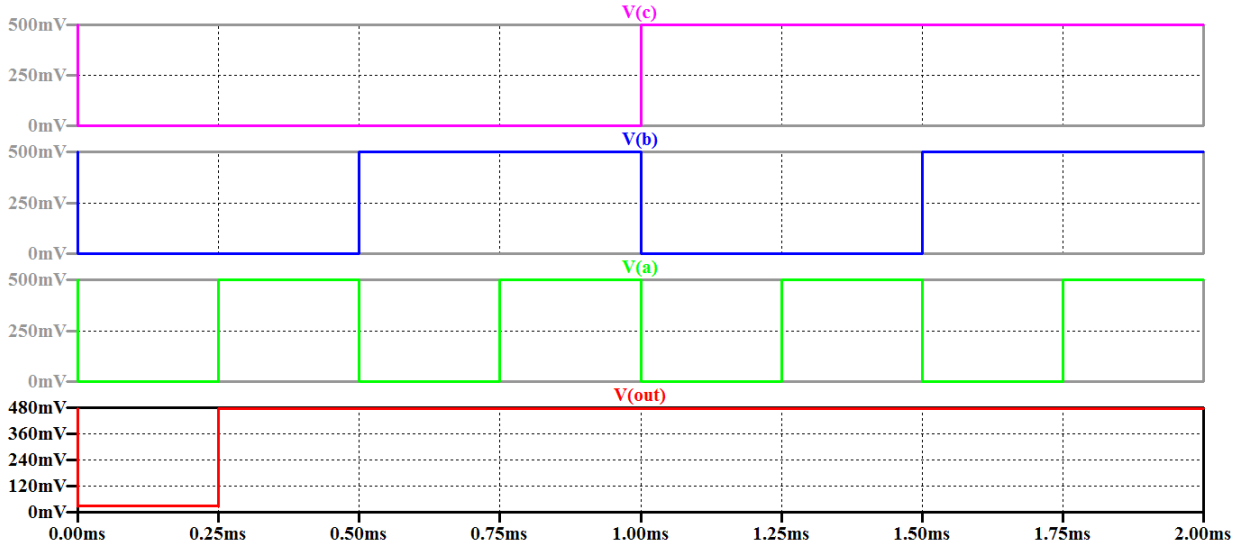

Figura A.6: Simulação da porta OR de 3 entradas

2.3 Porta OR de 4 entradas

Por fim a porta OR de 4 entradas pode ser implementada a partir de três portas OR de duas entradas cada. Sua função booleana fica definida como:

$$(A + B) + (C + D) = A + B + C + D \tag{4}$$

Seu esquemático e resultados encontram-se nas figuras A.7 e A.8 respectivamente.

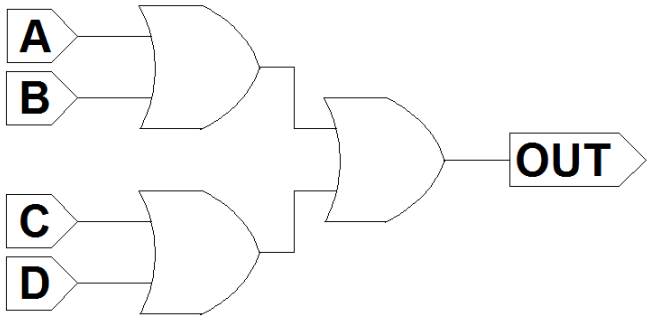

Figura A.7: Esquemático da porta OR de 4 entradas

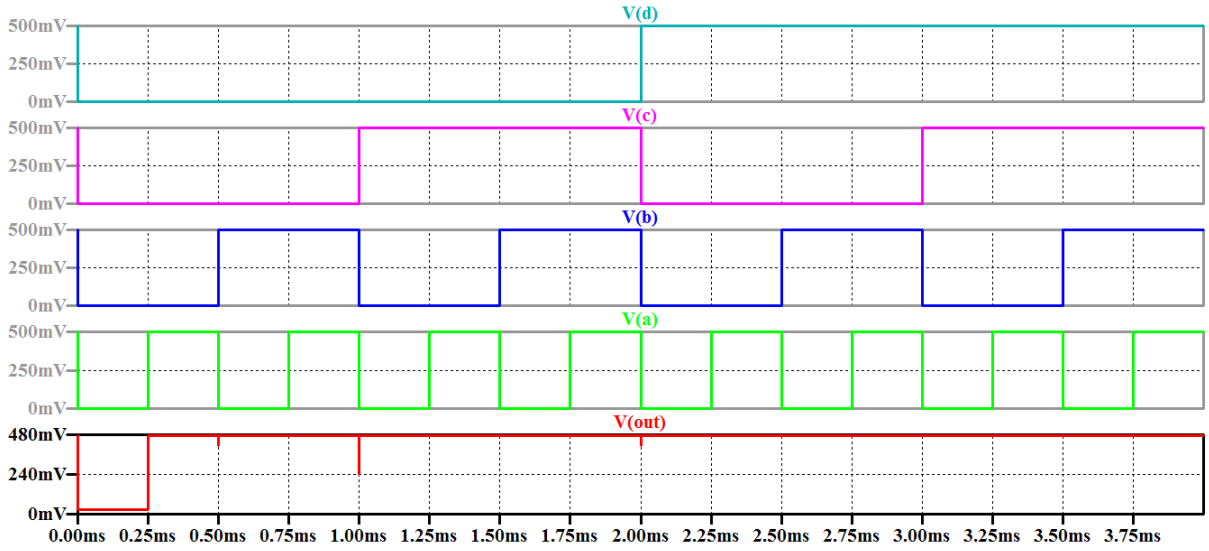

Figura A.8: Simulação da porta OR de 4 entradas

3 Porta AND

3.1 Porta AND de 2 entradas

A porta AND de duas entradas é imlementada a partir de uma porta NAND de duas entradas e uma porta NOT. Sua função booleana é dada por:

$$\overline{\overline{A.B}} = A.B \tag{5}$$

e sua arquitetura e resultado da simulação encontram-se nas figuras A.9 e A.10 respectivamente.

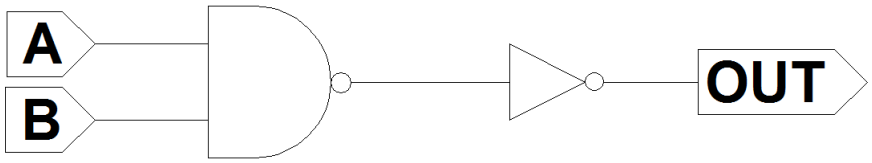

Figura A.9: Esquemático da porta AND de duas entradas

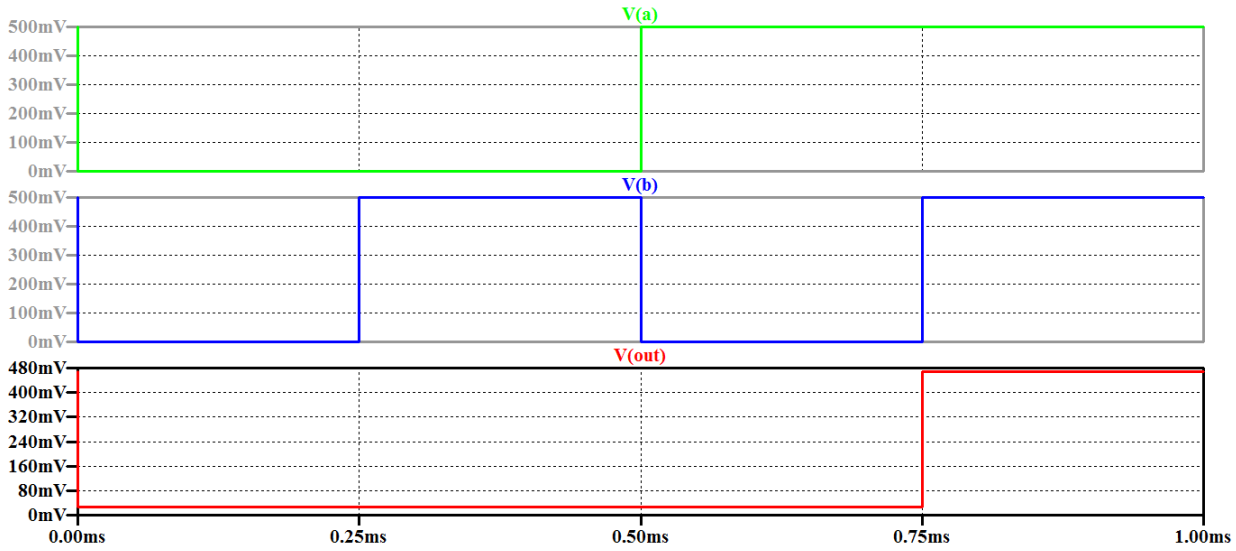

Figura A.10: Simulação da porta AND de duas entradas

3.2 Porta AND de 3 entradas

A porta AND de três entradas é realizada através de duas portas NAND de duas entradas e duas portas inversoras. Assim a função booleana AND para três entradas é:

$$\overline{[(\overline{A.B}).C]} = A.B.C \tag{6}$$

As figuras A.11 e A.12 apresentam o esquemático e os resultados respectivamente.

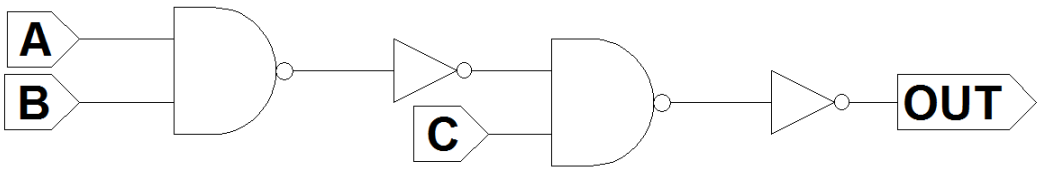

Figura A.11: Esquemático da porta AND de 3 entradas

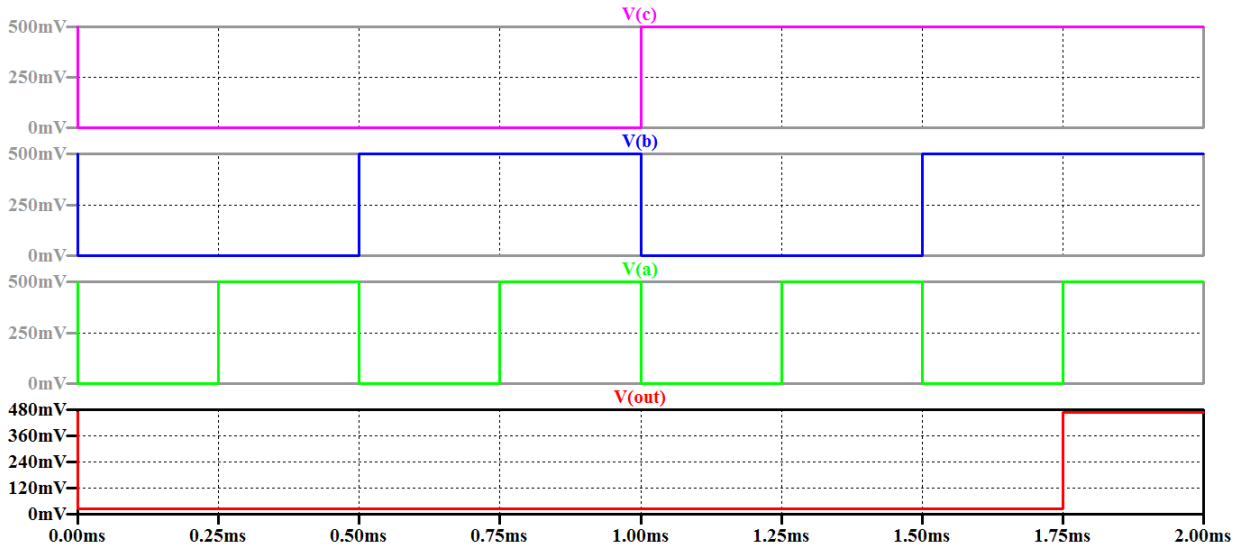

Figura A.12: Simulação da porta AND de 3 entradas

3.3 Porta AND de 4 entradas

Por fim a porta AND de quatro entradas pode ser construída a partir de duas portas NAND de duas entradas, uma porta OR de duas entradas e uma porta NOT. Sua função booleana é dada por:

$$\overline{(\overline{A.B}) + (\overline{C.D})} = A.B.C.D \tag{7}$$

O esquemático e os resultados são apresentados nas figuras A.13 e A.14.

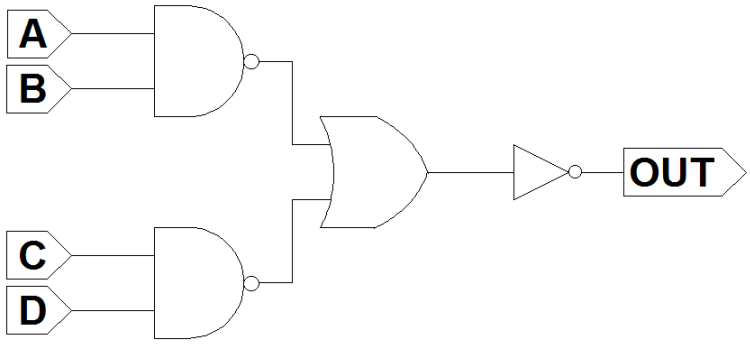

Figura A.13: Esquemático da porta AND de 4 entradas

4 Porta NAND de 3 entradas

A porta NAND de 3 entradas é facilmente implementada a partir de uma porta AND e uma porta NAND, ambas de 2 entradas, com a seguinte função booleana:

$$\overline{(A.B).C} = \overline{A.B.C} \tag{8}$$

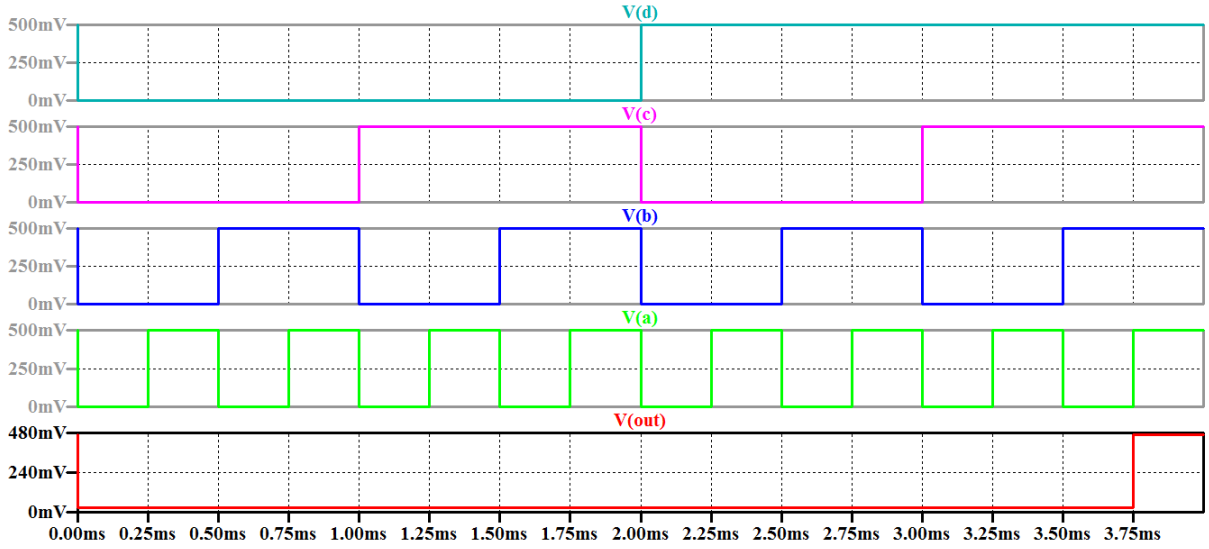

Figura A.14: Simulação da porta AND de 4 entradas

As figuras A.15 e A.16 apresentam o esquemático da porta e os resultados da simulação respectivamente.

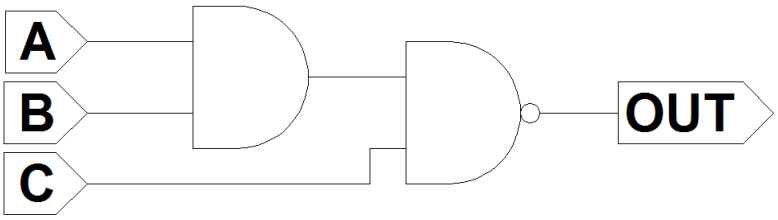

Figura A.15: Esquemático da porta NAND de 3 entradas

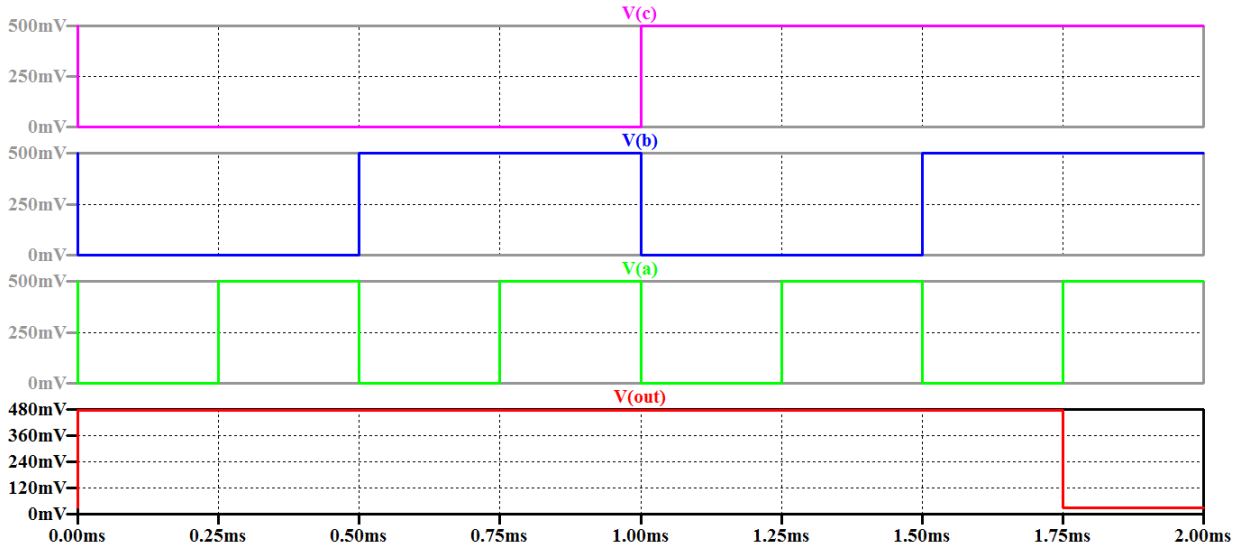

Figura A.16: Simulação da porta NAND de 3 entradas

5 Porta NOR de 4 entradas

A porta NOR de 4 entradas tem como função booleana:

$$\overline{(A+B+C+D)} = \overline{A+B+C+D} \tag{9}$$

Ou seja, a porta NOR pode ser implementada a partir de uma porta OR de 4 entradas, invertendo-se a sua saída. As figuras A.17 e A.18 apresentam o esquemático da porta e os resultados da sua simulação, respectivamente.

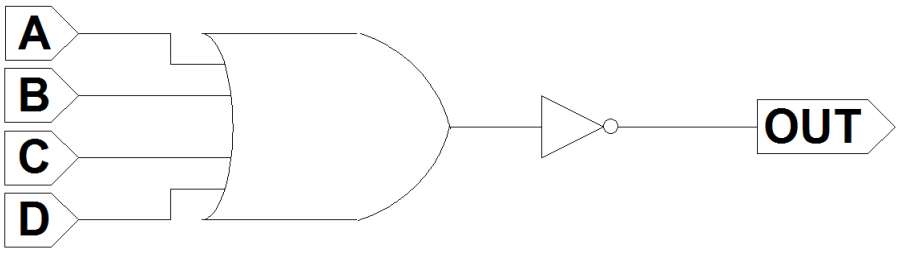

Figura A.17: Esquemático da porta NOR de 4 entradas

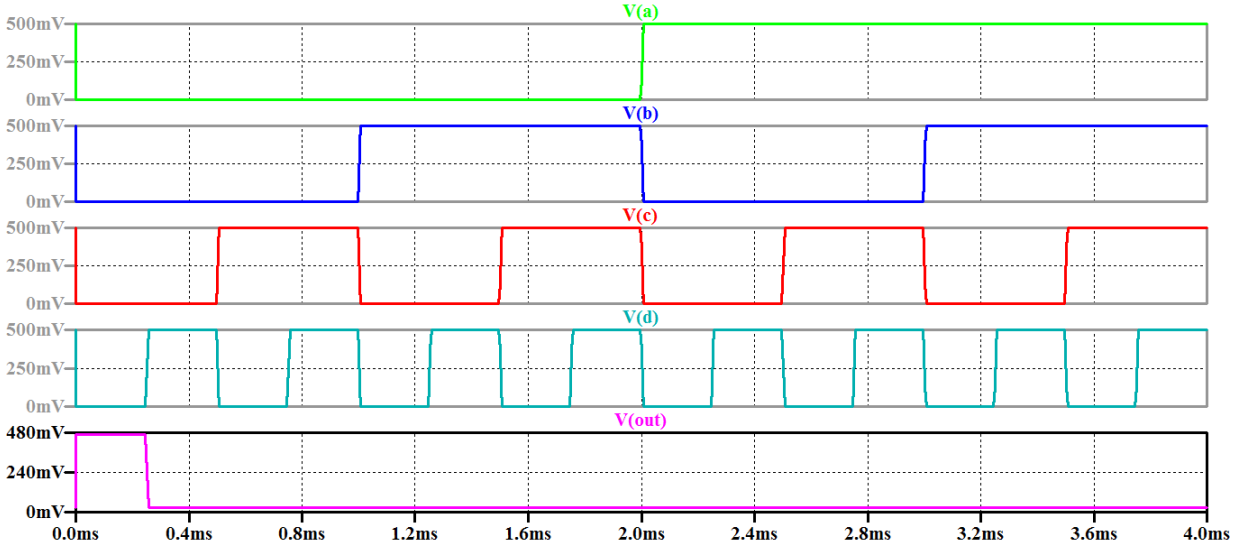

Figura A.18: Simulação da porta NOR de 4 entradas

6 Porta XOR

$$\overline{\overline{(\overline{A}.B)}.\overline{(A.\overline{B})}} = A \oplus B \tag{10}$$

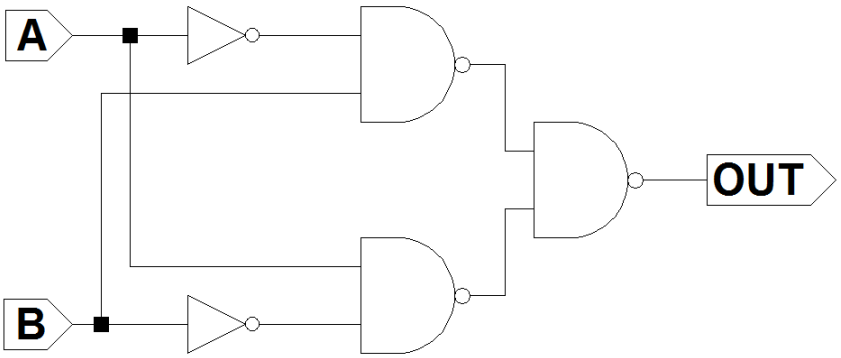

Figura A.19: Esquemático da porta XOR

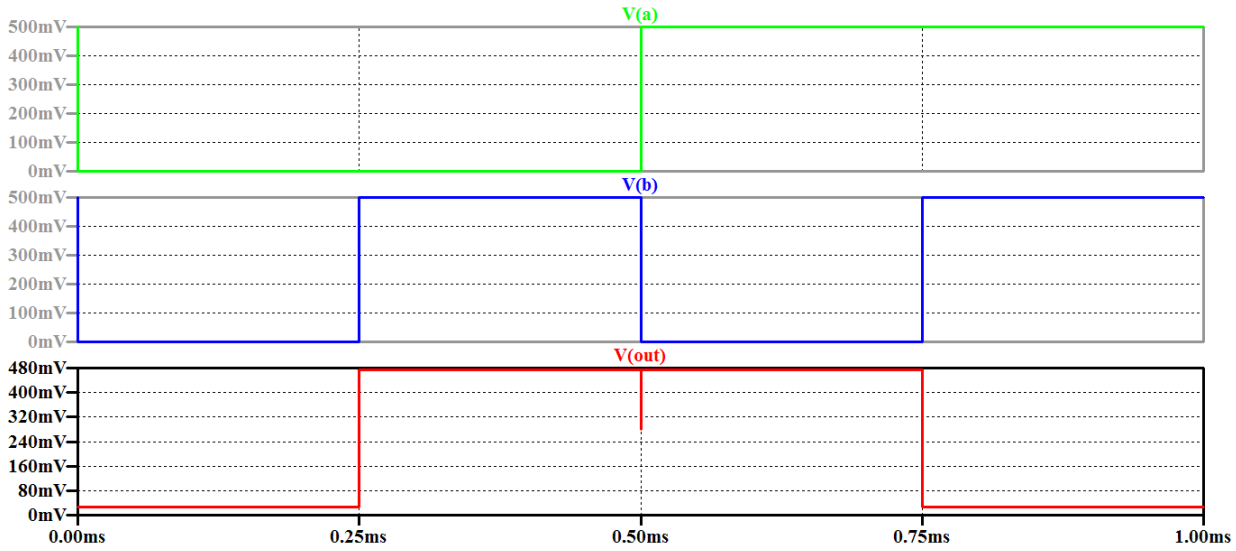

Figura A.20: Simulação da porta XOR

7 Decodificador Binário

O decodificador converte uma entrada codificada em uma saída codificada com um código diferente. Normalmente o código de saída possui mais bits que o código de entrada. Existe uma relação unívoca entre as palavras codificadas da entrada e as palavras codificadas da saída [16]. Dito isto o decoficador mais comum decodificador binário, ou decodificador n:2^n. Sua entrada é um código binário de n bits e sua saída é uma das n saídas correspondentes ao código binário. A figura A.21 apresenta o decodificador binário mais básico, de duas entradas para quatro saídas. Neste decodificador S1 é o MSB do código de entrada e a numeração da saída corresponde diretamente à conversão decimal do código binário de entrada. G corresponde ao sinal de *enable* do circuito.

Através do cascateamento do decodificador 2:4 é possível obter um decoficador de três entradas e oito saídas. O decodificador 3:8 é apresentado na figura A.22. S2 é o MSB do código de entrada deste decodificador. As figuras A.23 e A.24 apresentam o resultado das simulações dos decodificadores 2:4 e 3:8 respectivamente.

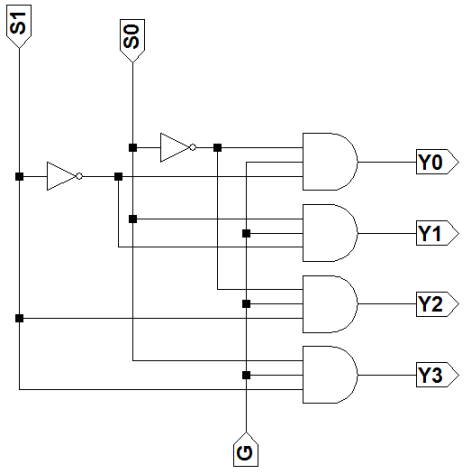

Figura A.21: Esquemático do decodificador 2:4

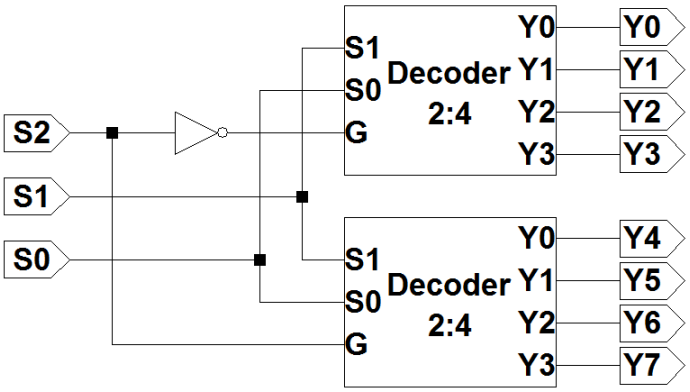

Figura A.22: Esquemático do decodificador 3:8

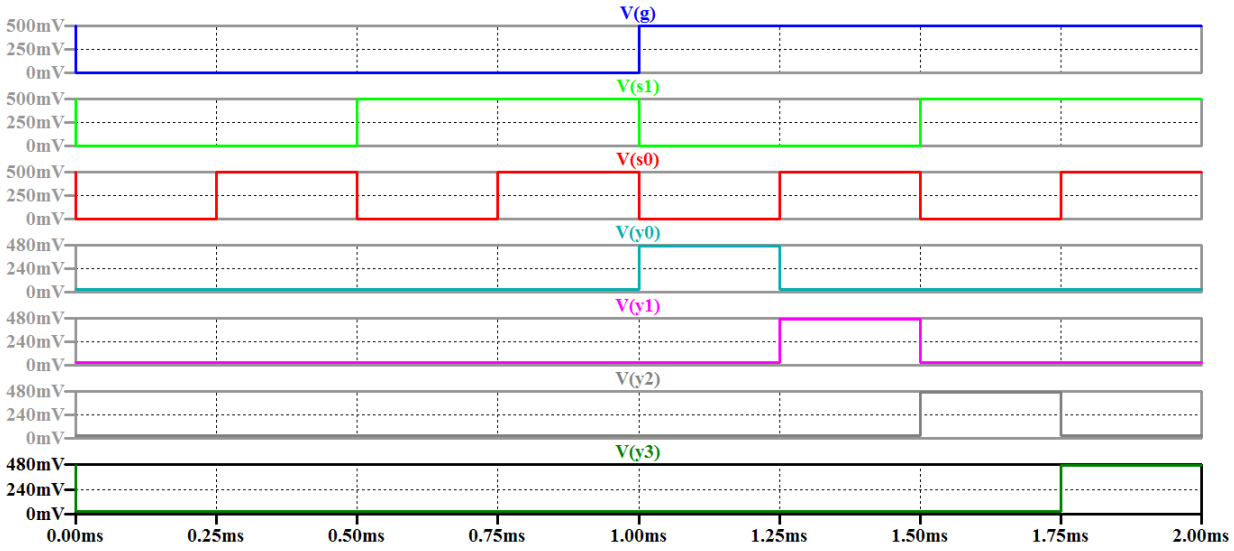

Figura A.23: Simulação do decodificador 2:4

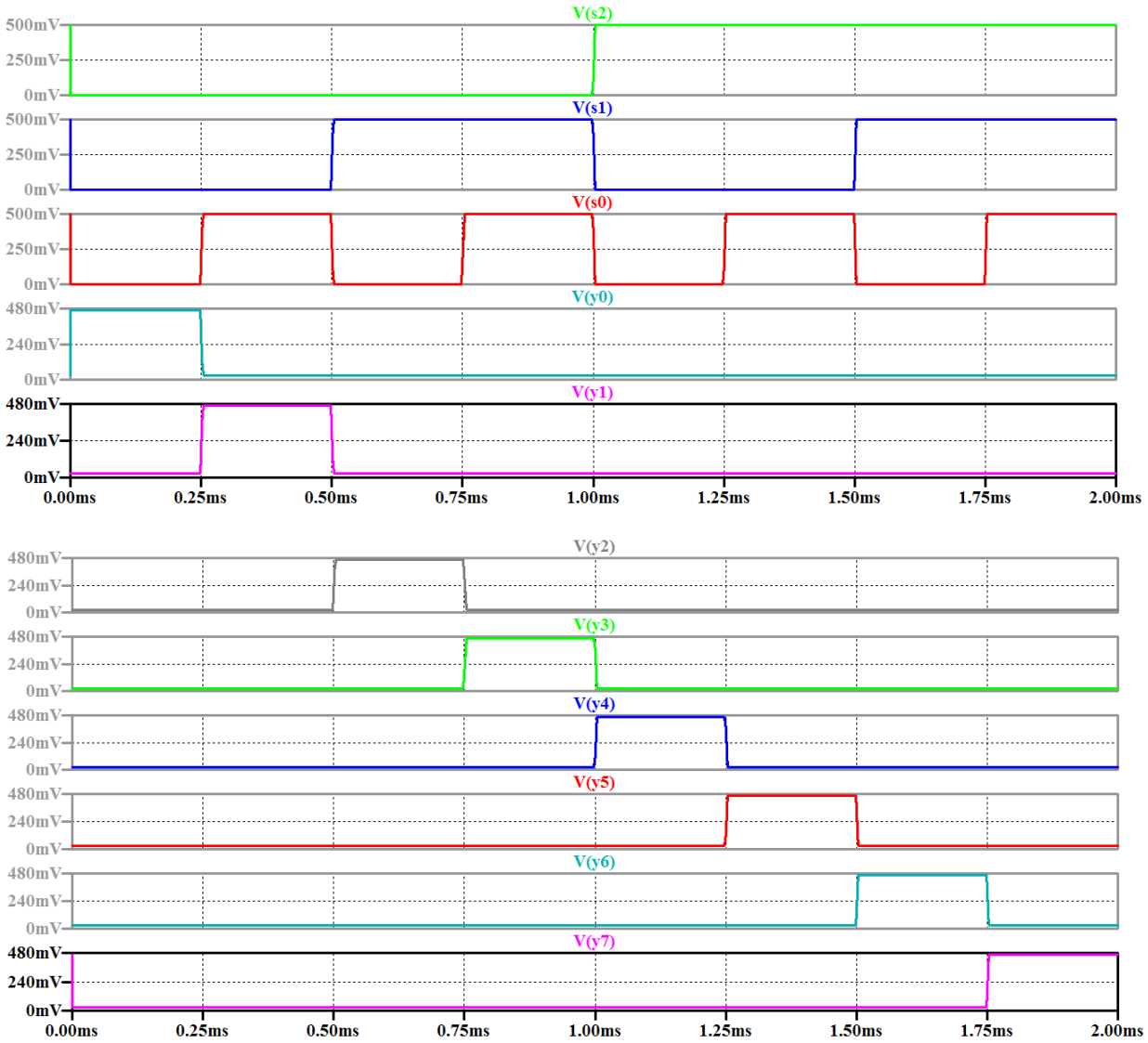

Figura A.24: Simulação do decodificador 3:8

8 Demultiplexador

Em um demultiplexador (DEMUX) os dados de sua única entrada são transferidos (ou roteados) para uma de suas várias saídas, de acordo com o endereço especificado no seu controle de seleção. Um DEMUX 1:4 (1 entrada para 4 saídas) é apresentado na figura A.25. Neste DEMUX X é a entrada de dados e C1 e C0 são o controle de seleção, onde C1 é o MSB. O DEMUX é ativado por um sinal *enable*.

O DEMUX 1:8 é facilmente obtido a partir do cascateamento do DEMUX 1:4 (figura A.26). Neste DEMUX a entrada é denominada D e o controle de seleção é dado por S2, S1 e S0, onde S2 é o MSB. As simulações dos DEMUX 1:4 e 1:8 são apresentadas nas figuras A.27 e A.28 respectivamente.

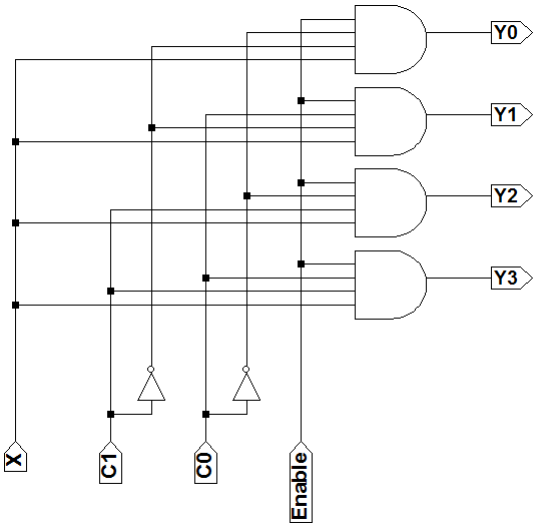

Figura A.25: Esquemático do DEMUX 1:4

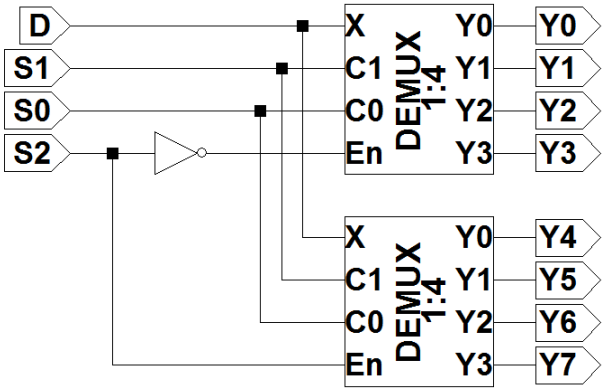

Figura A.26: Esquemático do DEMUX 1:8

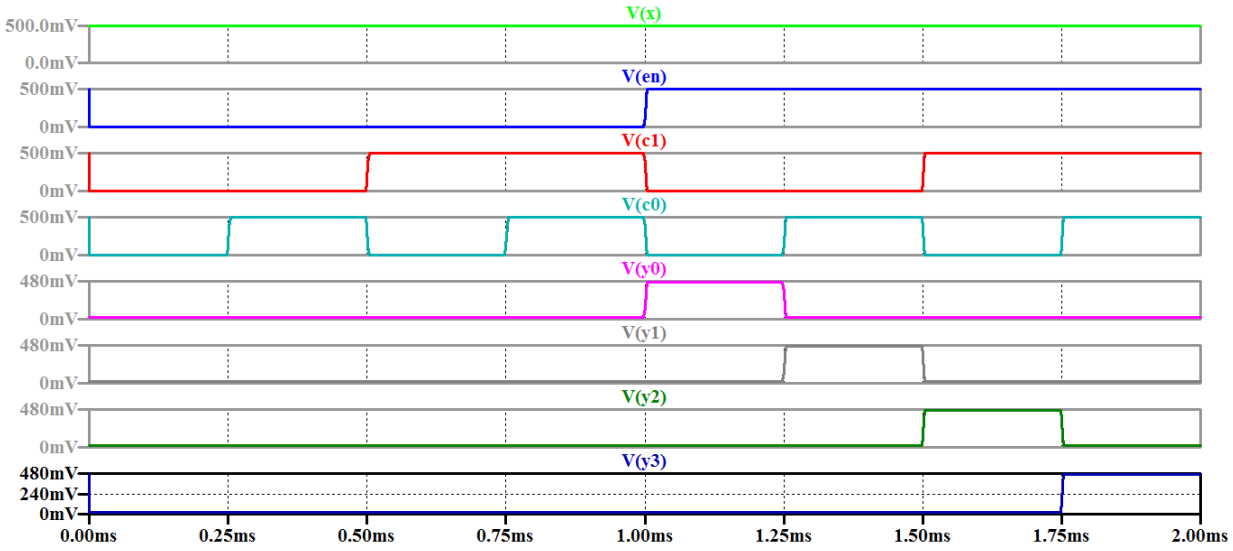

Figura A.27: Simulação do DEMUX 1:4

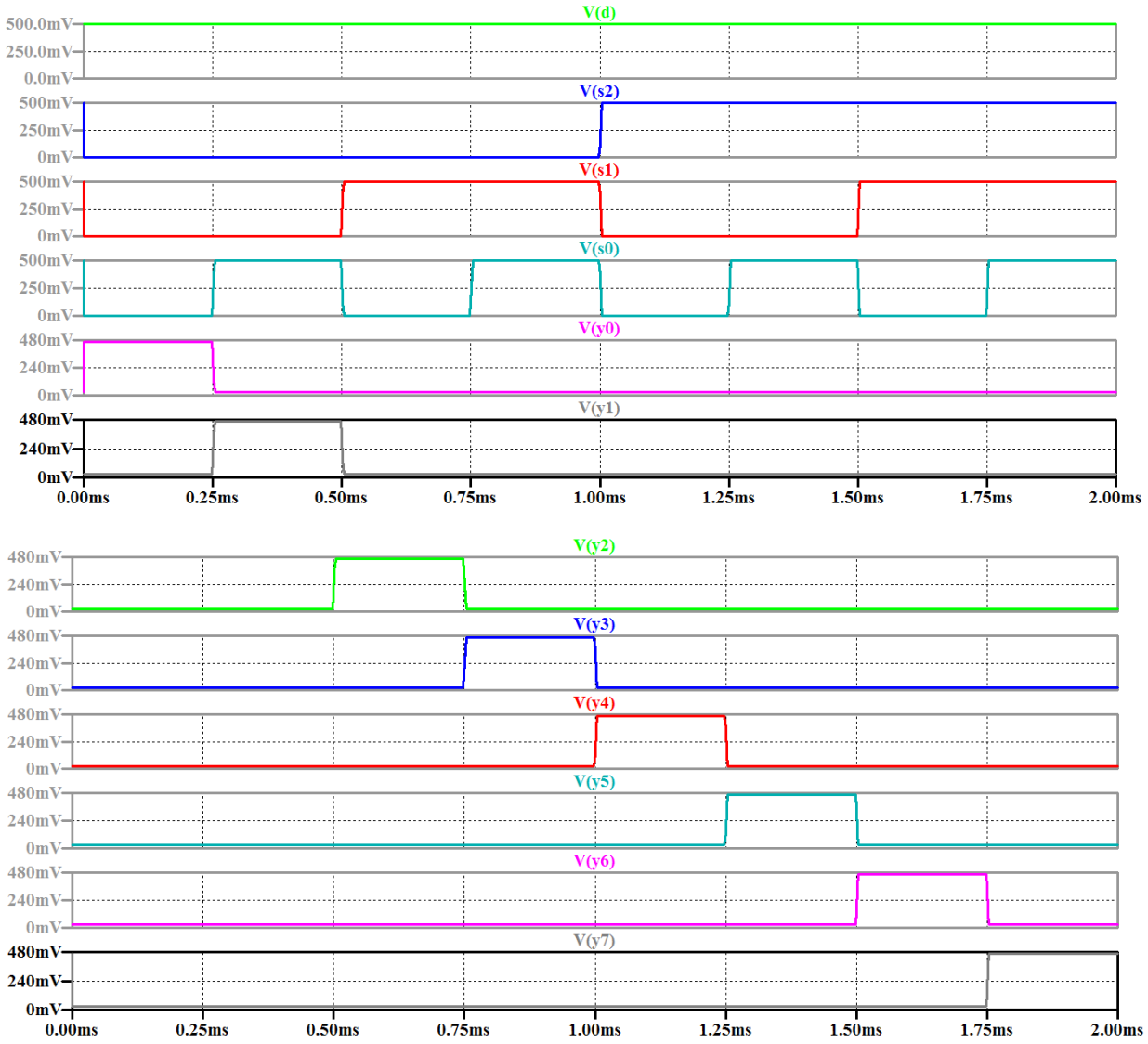

Figura A.28: Simulação do DEMUX 1:8

9 *Latch* SR com *Clear* Assíncrono

O *latch* SR é facilmente implementado através de duas portas NOR ou duas portas NAND. A entrada S significa *Set* e tem como função tornar o valor da saída Q = 1, enquanto a entrada R (*Reset*) torna a saída Q = 0. Quando ambos S e R forem 0 será mantido o estado anterior na saída Q, operação esta chamada de *hold*. Quando S e R forem 1 tanto Q quanto $\overline{Q}$ serão 1. Porém ambas as entradas não podem ser ativadas ao mesmo tempo, por gerar instabilidade no circuito, e portanto esta operação é considerada proibida. O *latch* SR pode possuir uma terceira entrada chamada *enable*. Sua função é manter o *hold* enquanto estiver desativada, e permitir o funcionamento normal do *latch* quado ativada. O *latch* SR com *enable* é constituído do *latch* SR mais simples acrescido de duas portas NAND. A figura A.29 apresenta a arquitetura de um *latch* SR com *enable* e *clear* assíncrono. Para a realização do *clear* foi feito uso de uma porta AND, uma porta OR e uma porta NOT.

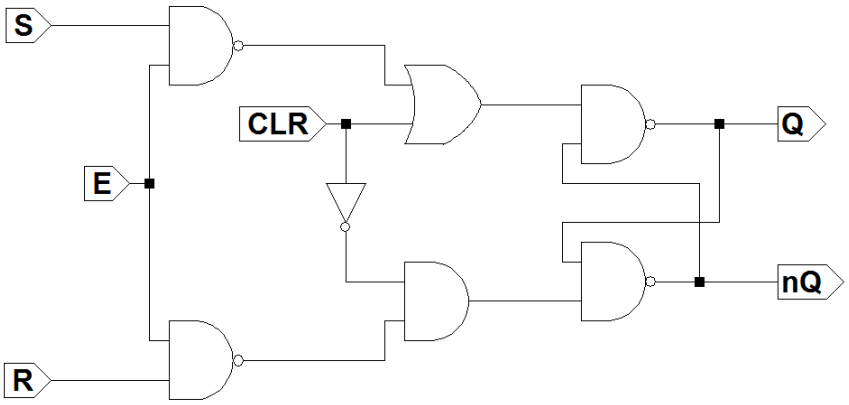

Figura A.29: Esquemático do *Latch* SR com *clear* assíncrono

A figura A.30 apresenta o funcionamento do *latch* SR.

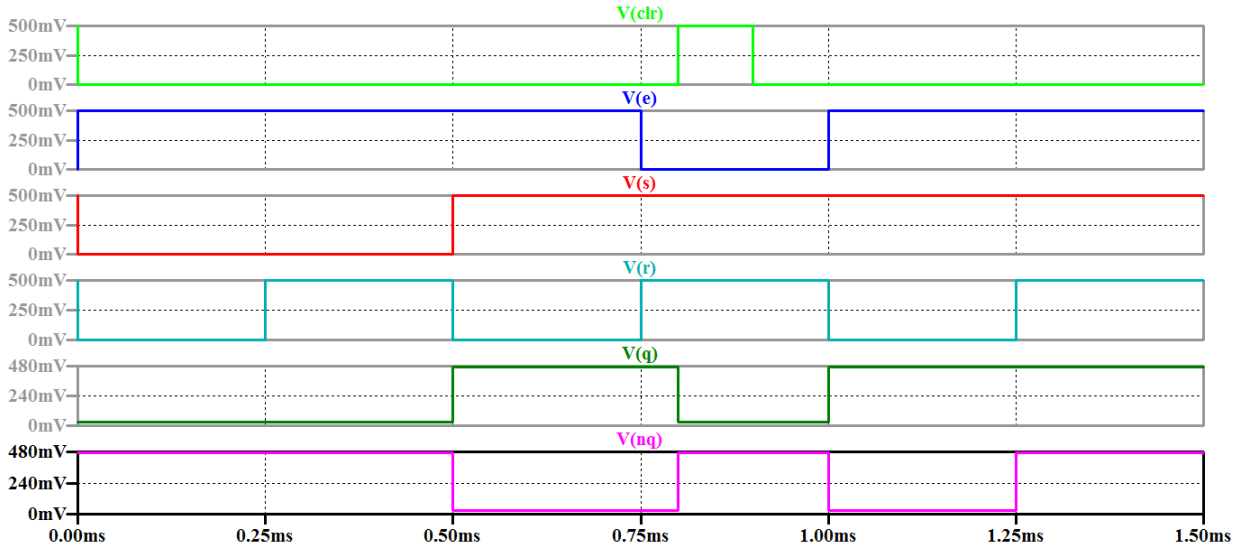

Figura A.30: Simulação do Latch SR

10 *Latch* D com *Clear* Assíncrono

O *latch* D é implementado a partir de um *latch* SR com *enable* conectando-se as entradas *set* e *reset* através de uma porta NOT. Enquanto o valor do *enable* for 0 será mantindo o último valor no *latch*. Quando *enable* for 1 a saída será a própria entrada D, o que faz com que o *latch* D seja chamado de *latch* transparente. A figura A.31 apresenta a arquitetura do *latch* D com *clear* assíncrono. A figura A.32 apresenta o funcionamento do *latch* D.

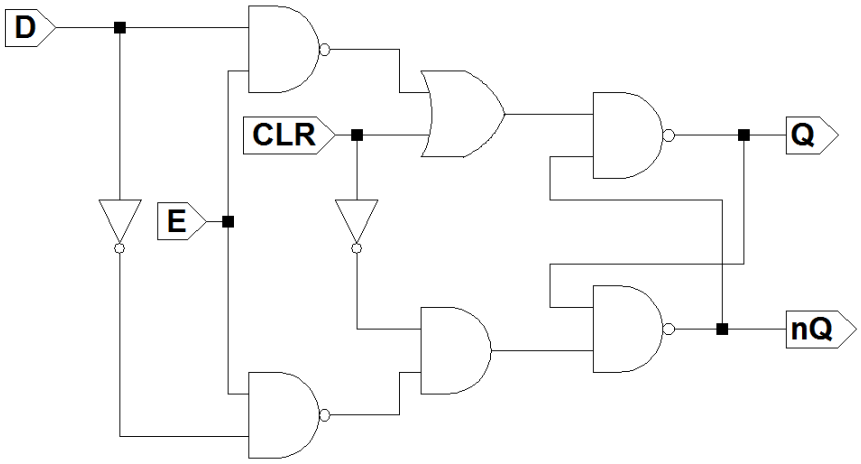

Figura A.31: Esquemático do *Latch* D com *clear* assíncrono

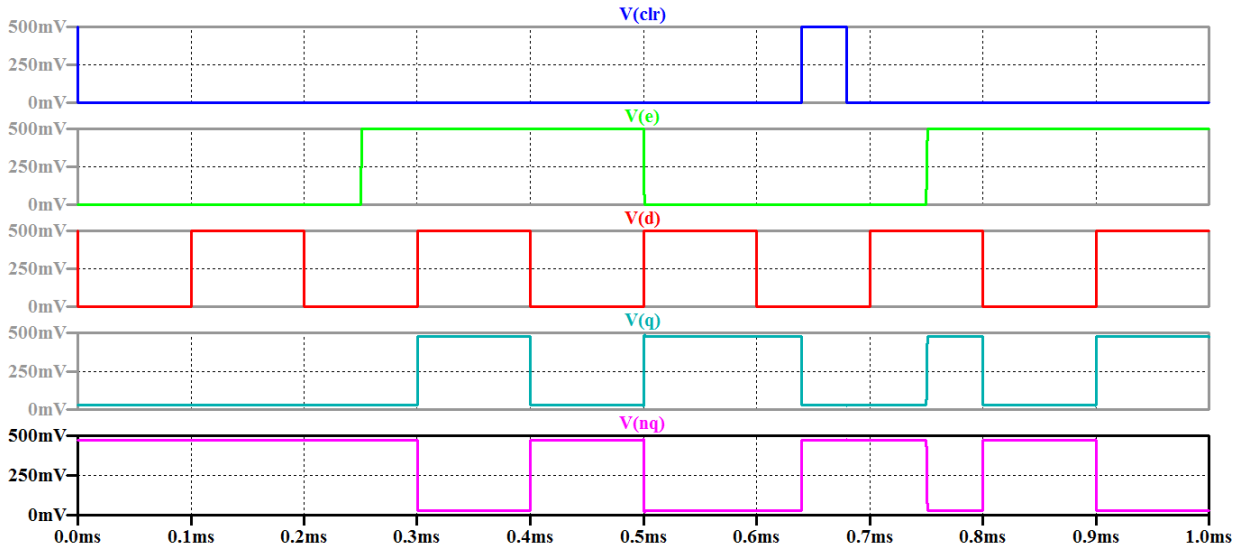

Figura A.32: Simulação do *Latch* D

11 *Latch* D com *Preset* Assíncrono

Seguindo a lógica do *latch* D com *clear* assíncrono, a entrada *preset* faz com que a entrada S seja 0 e a entrada R seja 1, de modo que a saída seja 1 a qualquer momento que PRE seja ativado. A figura A.33 apresenta o esquemático do *latch* D com *preset* assíncrono. Seu funcionamento é apresentado na figura A.34.

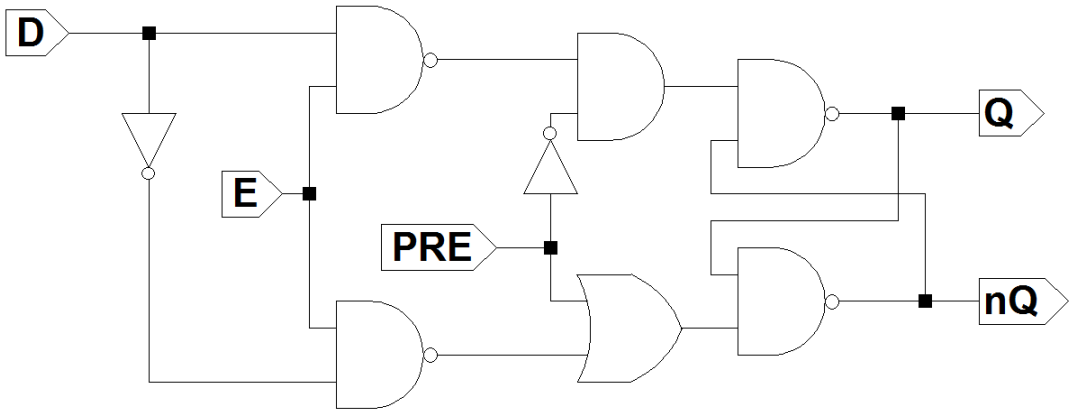

Figura A.33: Esquemático do *Latch* D com *preset* assíncrono

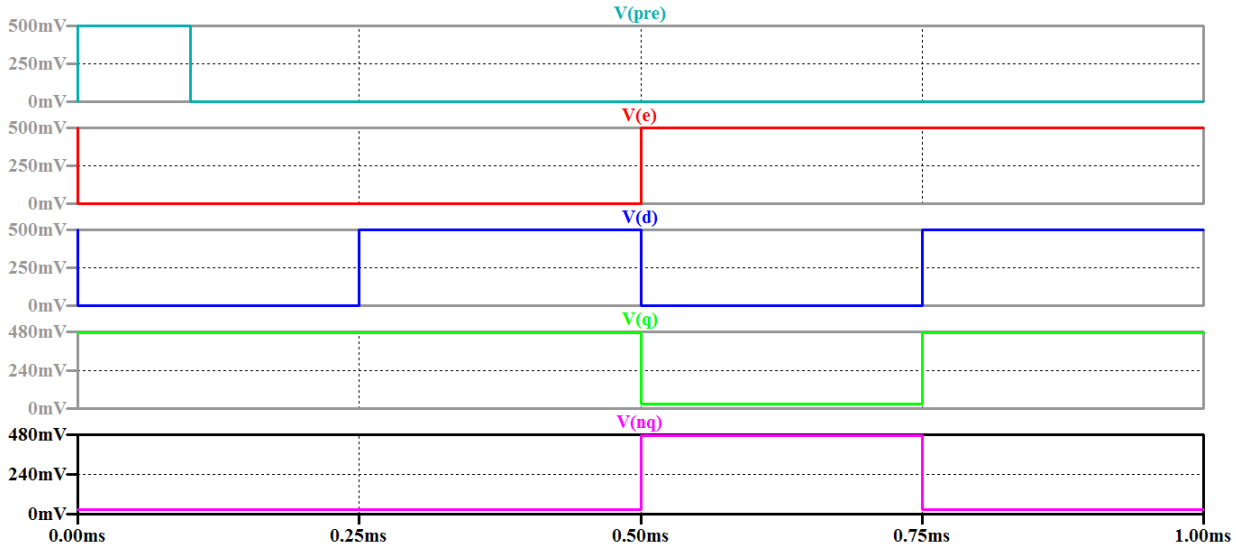

Figura A.34: Simulação do *Latch* D com *preset* assíncrono

12 *Flip-Flop* D com *Clear* Assíncrono

O *flip-flop* D é um dispositivo ativado pela borda de subida do sinal *clock*. Sua arquitetura é constituida por dois *latches* D, onde o primeiro *latch* é denominado mestre e o segundo é denominado escravo. O *latch* mestre segue a entrada quando o sinal de *clock* é 0. Quando o *clock* muda para 1 o mestre entra em *hold* e a sua saída é transferida para o escravo. Apesar de o escravo estar habilitado para mudanças na sua saída durante todo o período em o *clock* for 1, seu valor será alterado apenas na subida do *clock*, visto que durante este período o mestre encontra-se em *hold*.

A figura A.35 apresenta a arquitetura do *flip-flop* D. A saída seguirá a entrada D sempre que houver uma subida de *clock*. A figura A.36 apresenta os resultados da simulação do *flip-flop*.

É possível fazer um *flip-flop* D que seja ativado apenas na descida do *clock*. Para tanto basta que o sinal de *clock* seja invertido na entrada do *flip-flop* escravo e permaneça não invertido no *flip-flop* mestre. As figuras A.37 e A.38 apresentam o esquemático e a simulação do *flip-flop* D ativado na descida do *clock* respectivamente.

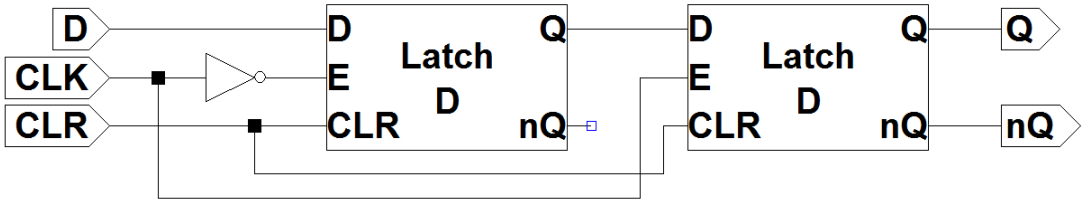

Figura A.35: Esquemático do *Flip-Flop* D

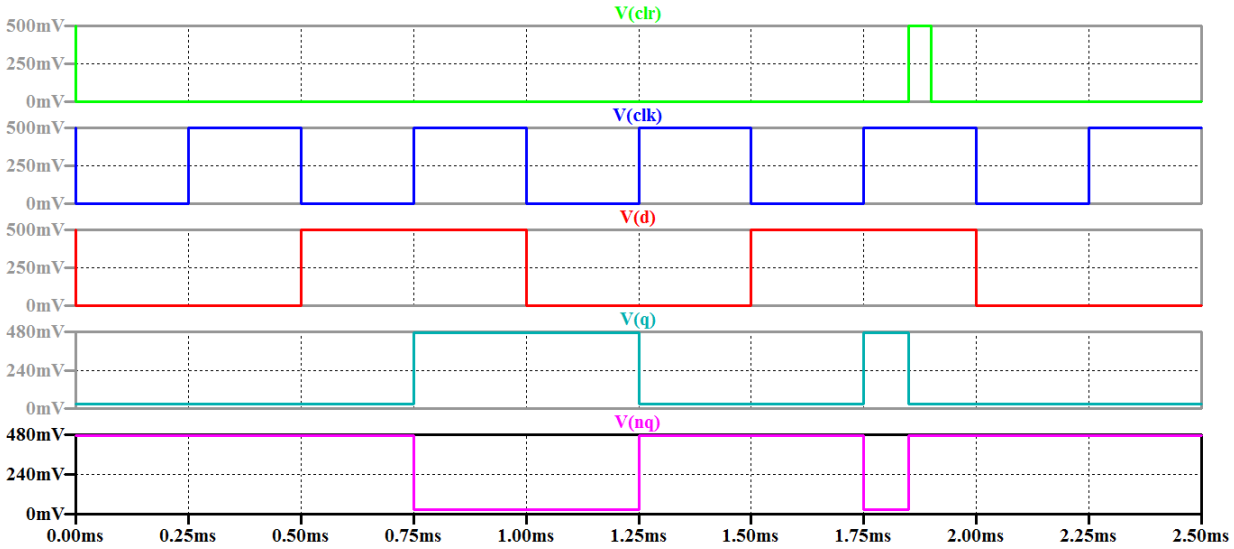

Figura A.36: Simulação do *flip-flop* D

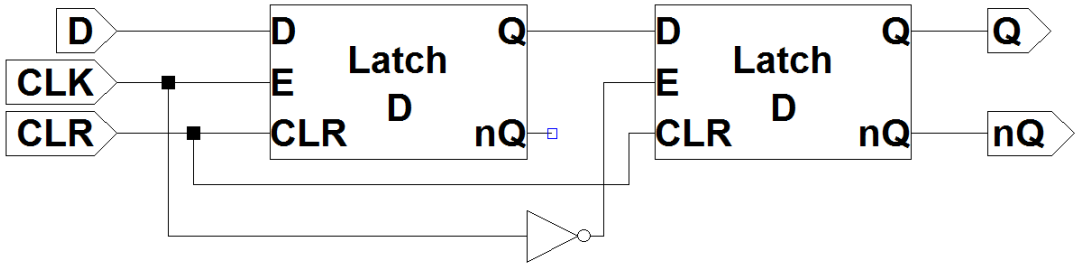

Figura A.37: Esquemático do *Flip-Flop* D ativado na descida do *clock*

13 *Flip-Flop* D com *Preset* Assíncrono

O *flip-flop* D com *preset* assíncrono também é composto por um *latch* mestre e um escravo. A diferença é que os *latches* utilizados são *latches* D com *preset*. As figuras A.39 e A.40 apresentam o esquemático e a simulção deste *flip-flop*, que é ativado na subida do *clock*.

14 Divisor de Frequência

O circuito divisor de frequência é facilmente realizável a partir de um *flip-flop* D. Ao realimentar a entrada D com a saída $\overline{Q}$ o sinal periódico adicionado na entrada CLK do *flip-flop* terá seu período multiplicado por 2. Para obter outras frequências basta cascatear novos *flip-flops*. Um

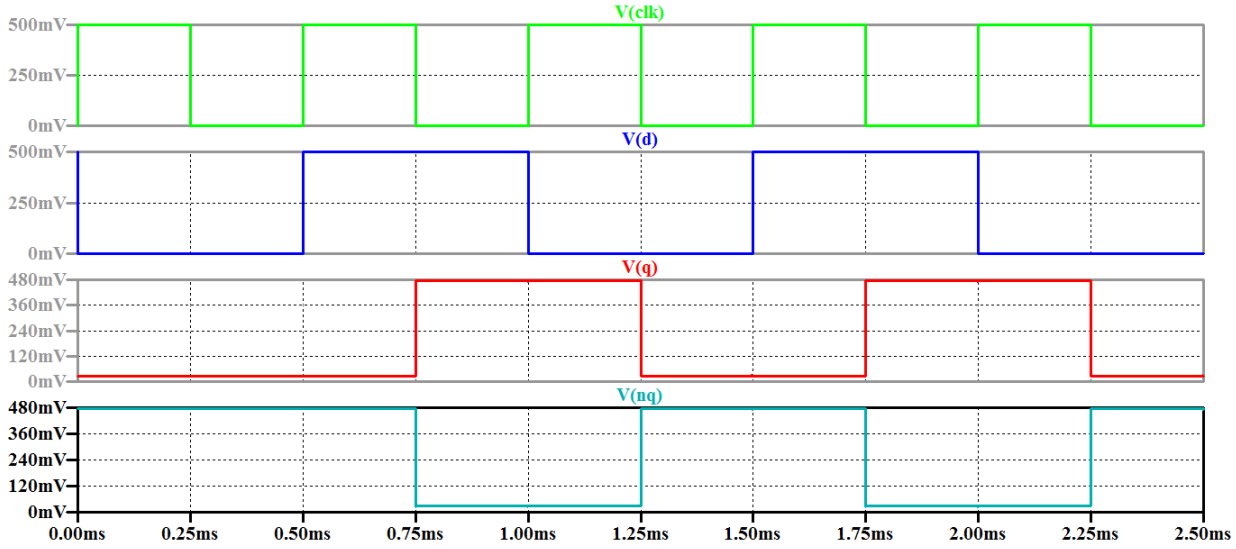

Figura A.38: Simulação do *flip-flop* D ativado na descida do *clock*

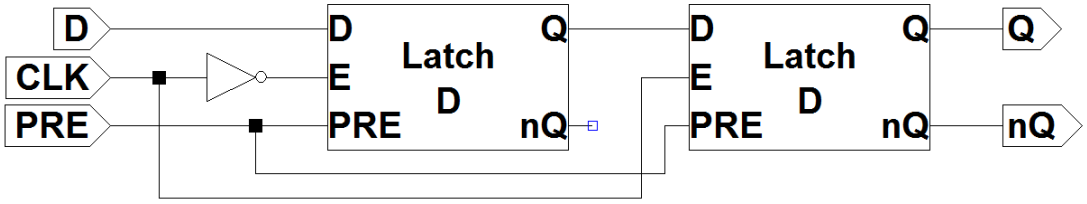

Figura A.39: Esquemático do *flip-flop* D com *preset* assíncrono

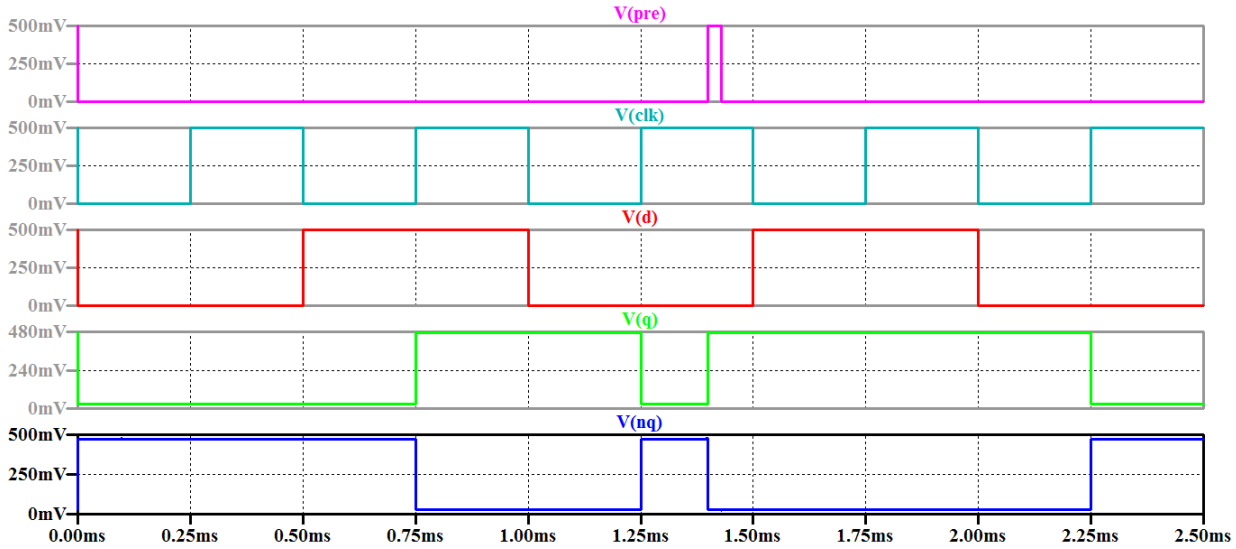

Figura A.40: Simulação do *flip-flop* D com *preset* assíncrono

circuito contendo *n flip-flops* produzirá todos os divisores de frequência de 2^1 à 2^n. A figura A.41 apresenta o esquemático de um divisor de frequência capaz de produzir até $f/16$, onde f é o sinal periódico a ser dividido. A figura A.42 apresenta o funcionamento do circuito.

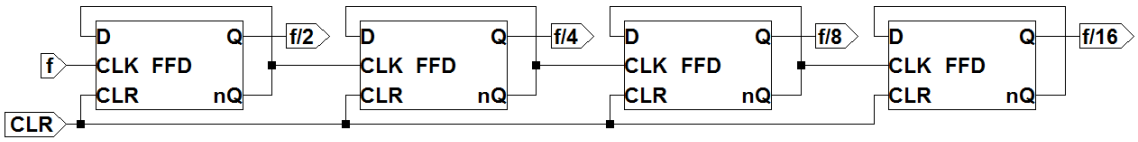

Figura A.41: Esquemático do divisor de frequências

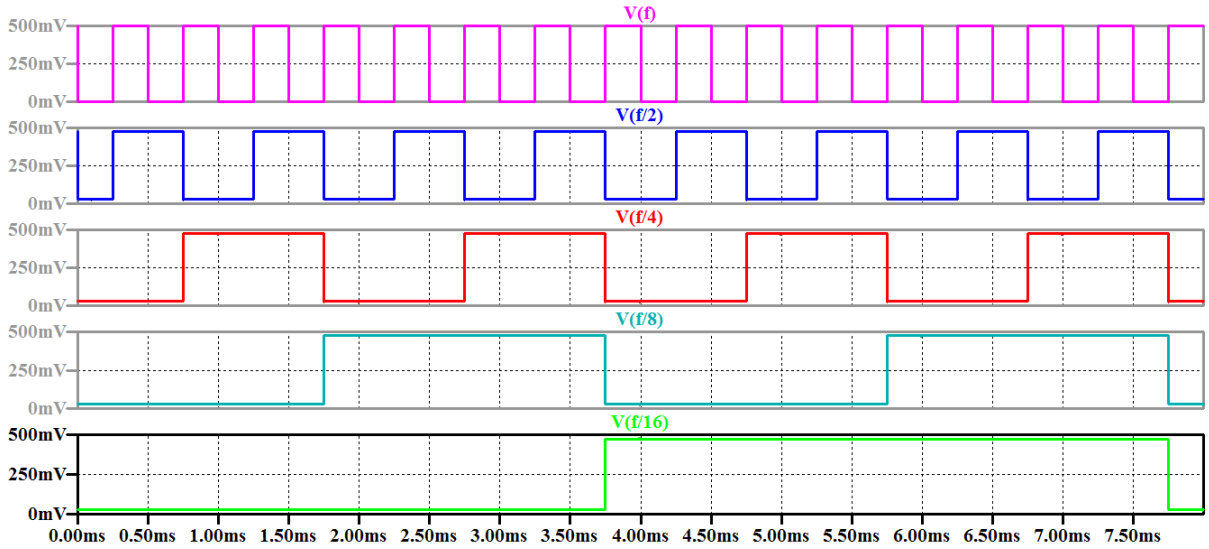

Figura A.42: Simulação do divisor de frequências

15 PISO

O PISO é um tipo de registrador de deslocamento capaz de serializar uma entrada de bits paralelos. Ele possui um sinal de entrada nW/S que controla o carregamento dos bits paralelos e a ativação do modo registrador de deslocamento para a serialização dos bits. Quando nW/S encontra-se baixo (0) os valores dos bits de entrada (D0 à D7) são armazenados nos *flip-flops* D durante a subida do *clock*. Quando o sinal nW/S é ativado (1) os bits armazenados são deslocados em uma posição à cada subida de *clock*. A figura A.43 apresenta o esquemático do PISO de 8 bits.

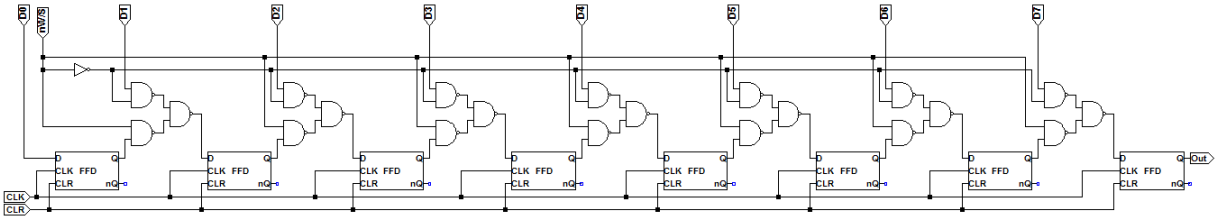

Figura A.43: Esquemático do PISO de 8 bits

A figura A.44 apresenta os resultados de uma simulação do PISO. Para este teste foi utilizada a palavra de 8 bits da tabela 1 como bits de entrada. É interessante notar que são necessários 1 ciclo de *clock* para que a palavra de entrada seja armazenada e mais 8 ciclos para que toda ela apareça serializada na saída, somando um total de 9 ciclos para que uma palavra de 8 bits seja

completamente serializada. Também deve-se ter cuidado para que a ativação e desativação do sinal nW/S ocorra antes de uma subida do *clock*.

Tabela 1: Palavra de entrada para o teste do PISO

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1	0	0	1

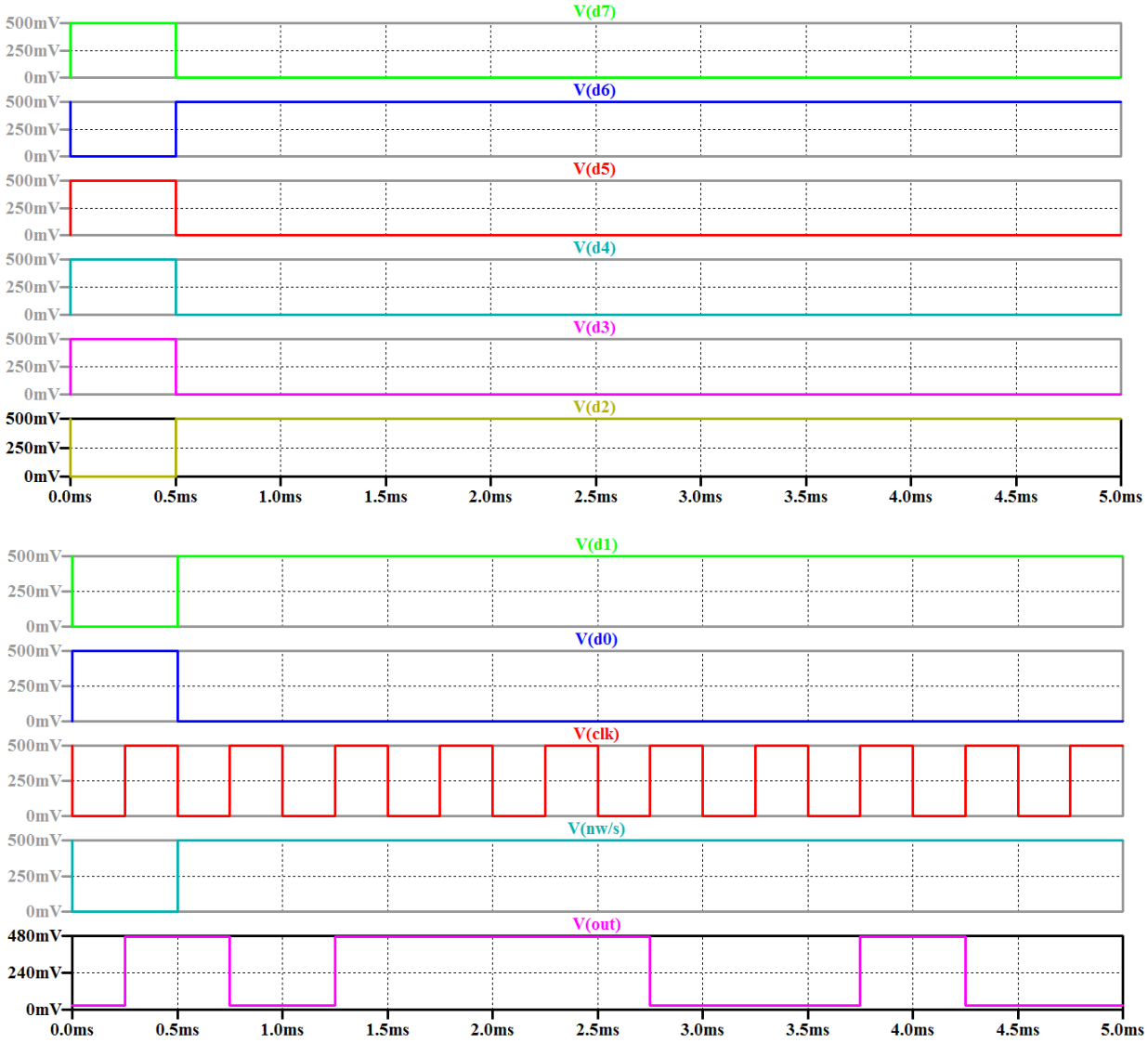

Figura A.44: Simulação do resgistrador PISO

16 Contador Binário

Um contador é qualquer circuito sequencial com *clock* cujo diagrama de estados possua apenas um único ciclo [16]. Um contador binário de n-bits é composto por n *flip-flops* e possui 2^n estados. A figura A.45 apresenta o esquemático de um contador binário de 3 bits, que é capaz de contar de 0 à 7. As saídas apresentam o atual valor do contador em código binário, onde Y2 é o MSB, sendo que o contador é ativado na descida do *clock*. A figura A.46 apresenta o resultado da

simulação do contador.

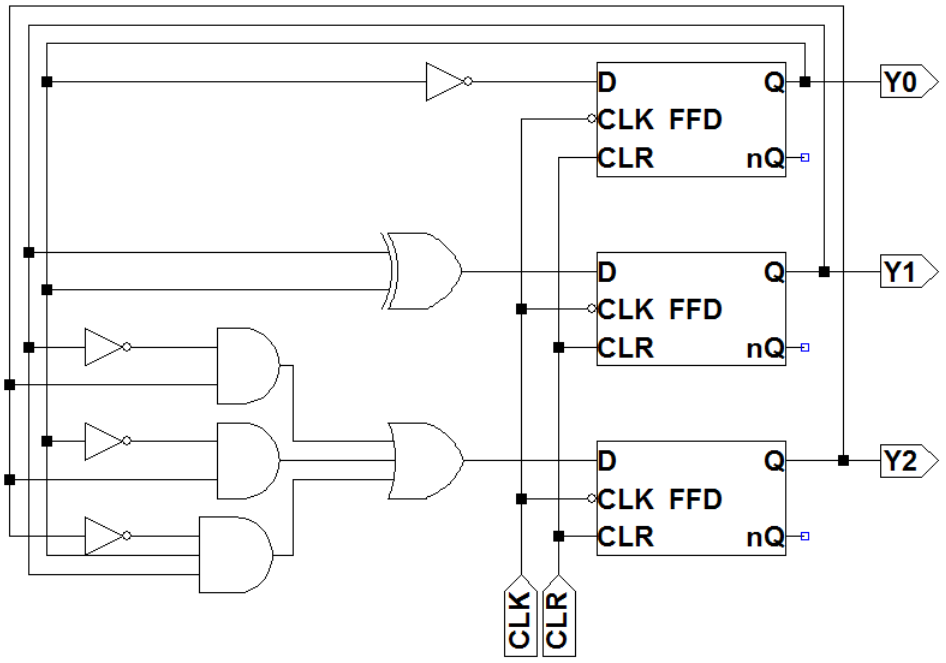

Figura A.45: Esquemático do contador binário de 3 bits

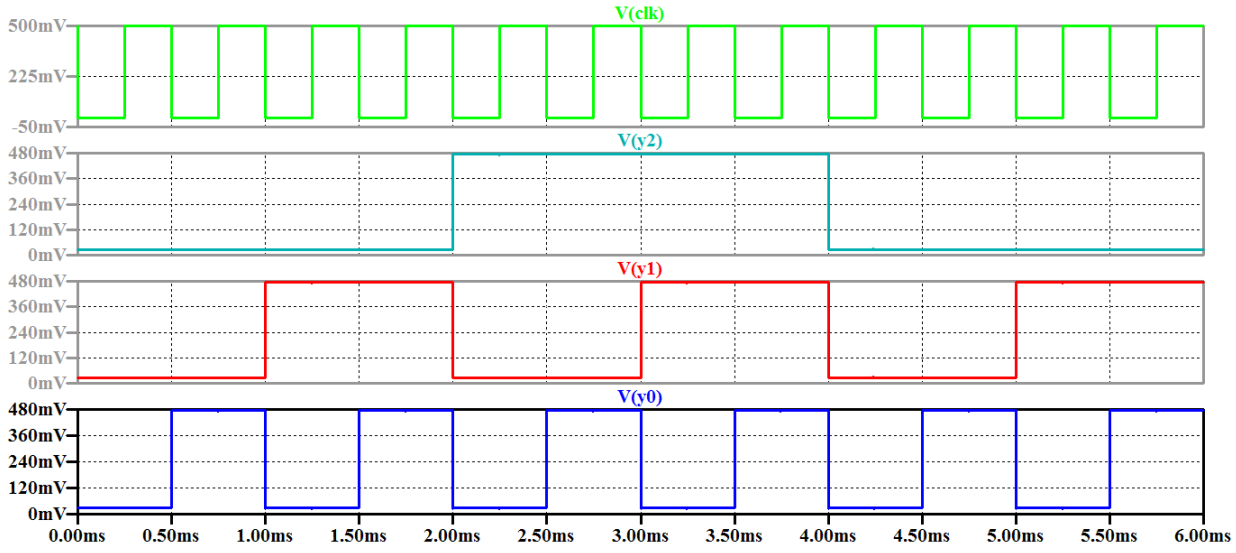

Figura A.46: Simulação do contador binário de 3 bits

APÊNDICE B

Este apêndice apresenta o código SPICE do SET proposto por *Lientsching et. al.* [29].

```
CE - A SPICE Model of a Single-Electron Transistor
*    Copyright (C) 2001  Peter Hadley and Guenther Lientschnig
*    Delft University of Technology, The Netherlands
*    Further Information about this program available in the article
*    "Simulating Hybrid Circuits of Single-Electron Transistors and Field-
*    Effect Transistors", G. Lientschnig, I. Weymanm, and P. Hadley,
*    Japanese Journal of Applied Physics, 42, 6467-6472 (2003).
*    and at http://lamp.tu-graz.ac.at/~hadley/set/spice/
*
*    This program is free software: you can redistribute it and/or modify
*    it under the terms of the GNU General Public License as published by
*    the Free Software Foundation, either version 3 of the License, or
*    any later version.
*
*    This program is distributed in the hope that it will be useful,
*    but WITHOUT ANY WARRANTY; without even the implied warranty of
*    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
*    GNU General Public License for more details.
*
*    The GNU General Public License can be found at:
*    http://www.gnu.org/licenses/
*
************************************************
*single electron transistor
*connections: source
*             | drain
*             | | gate1
*             | | | gate2
*             | | | |
.SUBCKT SETis   1 2 3 4 PARAMS:

     +C1=0.1E-19                      ;  Capacitance of junction 1
     +C2=0.1E-19                      ;  Capacitance of junction 2
     +R1=1E7                        ;  Resistance of junction 1
     +R2=1E7                         ;  Resistance of junction 2
     +Cg1=1.0E-19                     ;  Capacitance of gate 1
     +Cg2=1.0E-19



*    +C1=1E-18                       ;  Capacitance of junction 1
*    +C2=1E-18                       ;  Capacitance of junction 2
*     +R1=1E5                         ;  Resistance of junction 1
*     +R2=1E5                         ;  Resistance of junction 2
*     +Cg1=1E-18                      ;  Capacitance of gate 1
```

```
*     +Cg2=0                          ;  Capacitance of gate 2
      +C0=0                           ;  Self Capacitance of the island
      +Q0=0.0                         ;  Offset charge in units of e
      +TEMP=300                       ;  Temperature


.PARAM CSUM={C1+C2+Cg1+Cg2+C0}      ;  The total capacitance of the SET
.PARAM T={TEMP*CSUM*5.3785467E14}   ;  Normalized temperature, 5.3785467E14 = kB/e^2
.PARAM RN1={R1/(R1+R2)}             ;  Normalized resistance of junction 1
.PARAM RN2={R2/(R1+R2)}             ;  Normalized resistance of junction 2

.FUNC Q(a,b,c,d) { (Cg1*c+Cg2*d+C1*a+C2*b)/echarge+Q0 }
            ;  Definition of a charge term in units of e
.FUNC VN(v) { CSUM*v/echarge }
                    ;  The normalized voltage
.FUNC GAMMA(u) { IF(T==0,IF(u<0,-u,0),IF(u==0,T,u/(EXP(u/T)-1)))}
   ;  The rate function
*.FUNC ROUND(x) { x-IF(cos(pi*x)>0,arcsin(sin(pi*x))/pi,-arcsin(sin(pi*x))/pi) }
 ;  The round() function
.FUNC NOPT(a,b,c,d) { ROUND((-1*Q(a,b,c,d))+(CSUM/echarge)*(a*RN2+b*RN1)) }
    ;  The most probable charge on the island in units of e

*************************** the rates for the four tunnel events******************

.FUNC R1L(n,a,b,c,d) {GAMMA(0.5 - n - Q(a,b,c,d) + VN(a))/RN1}
.FUNC R1R(n,a,b,c,d) {GAMMA(0.5 + n + Q(a,b,c,d) - VN(a))/RN1}
.FUNC R2L(n,a,b,c,d) {GAMMA(0.5 + n + Q(a,b,c,d) - VN(b))/RN2}
.FUNC R2R(n,a,b,c,d) {GAMMA(0.5 - n - Q(a,b,c,d) + VN(b))/RN2}

*.FUNC NO(v1,v2,v3,v4) { ROUND(-Q(v1,v2,v3,v4)+(CSUM/E)*(v1*RN2+v2*RN1)) }

* determine the relative probabilities
;charge state N_OPT is initially assumed to have a relative
;probability equal to one
.FUNC PN_1(n,a,b,c,d) {(R1L(n,a,b,c,d)+R2R(n,a,b,c,d))/(R1R(n-1,a,b,c,d)
      +R2L(n-1,a,b,c,d))}
.FUNC PN_2(n,a,b,c,d) { PN_1(n,a,b,c,d)*
      +(R1L(n-1,a,b,c,d)+R2R(n-1,a,b,c,d))/(R1R(n-2,a,b,c,d)+R2L(n-2,a,b,c,d))}
.FUNC PN_3(n,a,b,c,d) { PN_2(n,a,b,c,d)*
      +(R1L(n-2,a,b,c,d)+R2R(n-2,a,b,c,d))/(R1R(n-3,a,b,c,d)+R2L(n-3,a,b,c,d))}
.FUNC PN_4(n,a,b,c,d) { PN_3(n,a,b,c,d)*
      +(R1L(n-3,a,b,c,d)+R2R(n-3,a,b,c,d))/(R1R(n-4,a,b,c,d)+R2L(n-4,a,b,c,d))}
.FUNC PN_5(n,a,b,c,d) { PN_4(n,a,b,c,d)*
      +(R1L(n-4,a,b,c,d)+R2R(n-4,a,b,c,d))/(R1R(n-5,a,b,c,d)+R2L(n-5,a,b,c,d))}
.FUNC PN1(n,a,b,c,d) {(R2L(n,a,b,c,d)+R1R(n,a,b,c,d))/(R2R(n+1,a,b,c,d)
      +R1L(n+1,a,b,c,d))}
.FUNC PN2(n,a,b,c,d) { PN1(n,a,b,c,d)*
      +(R2L(n+1,a,b,c,d)+R1R(n+1,a,b,c,d))/(R2R(n+2,a,b,c,d)+R1L(n+2,a,b,c,d))}
.FUNC PN3(n,a,b,c,d) { PN2(n,a,b,c,d)*
      +(R2L(n+2,a,b,c,d)+R1R(n+2,a,b,c,d))/(R2R(n+3,a,b,c,d)+R1L(n+3,a,b,c,d))}
.FUNC PN4(n,a,b,c,d) { PN3(n,a,b,c,d)*
      +(R2L(n+3,a,b,c,d)+R1R(n+3,a,b,c,d))/(R2R(n+4,a,b,c,d)+R1L(n+4,a,b,c,d))}
```

```
.FUNC PN5(n,a,b,c,d) { PN4(n,a,b,c,d)*
      +(R2L(n+4,a,b,c,d)+R1R(n+4,a,b,c,d))/(R2R(n+5,a,b,c,d)+R1L(n+5,a,b,c,d))}

.FUNC PSUM(n,a,b,c,d) { PN_5(n,a,b,c,d)+PN_4(n,a,b,c,d)+PN_3(n,a,b,c,d)
      +PN_2(n,a,b,c,d)
      ++PN_1(n,a,b,c,d)+1+PN1(n,a,b,c,d)+PN2(n,a,b,c,d)+PN3(n,a,b,c,d)
      ++PN4(n,a,b,c,d)+PN5(n,a,b,c,d) }

*************** calculate the current from source to drain ***************

.FUNC CUR(n,a,b,c,d) { PN_5(n,a,b,c,d)*(R1R(n-5,a,b,c,d)-R1L(n-5,a,b,c,d))
                           ++PN_4(n,a,b,c,d)*(R1R(n-4,a,b,c,d)-R1L(n-4,a,b,c,d))
                           ++PN_3(n,a,b,c,d)*(R1R(n-3,a,b,c,d)-R1L(n-3,a,b,c,d))
                           ++PN_2(n,a,b,c,d)*(R1R(n-2,a,b,c,d)-R1L(n-2,a,b,c,d))
                           ++PN_1(n,a,b,c,d)*(R1R(n-1,a,b,c,d)-R1L(n-1,a,b,c,d))
                           ++(R1R(n,a,b,c,d)-R1L(n,a,b,c,d))
                           ++PN1(n,a,b,c,d)*(R1R(n+1,a,b,c,d)-R1L(n+1,a,b,c,d))
                           ++PN2(n,a,b,c,d)*(R1R(n+2,a,b,c,d)-R1L(n+2,a,b,c,d))
                           ++PN3(n,a,b,c,d)*(R1R(n+3,a,b,c,d)-R1L(n+3,a,b,c,d))
                           ++PN4(n,a,b,c,d)*(R1R(n+4,a,b,c,d)-R1L(n+4,a,b,c,d))
                           ++PN5(n,a,b,c,d)*(R1R(n+5,a,b,c,d)-R1L(n+5,a,b,c,d)) }

.FUNC CURRENT(n,a,b,c,d) { echarge*CUR(n,a,b,c,d)/(CSUM*PSUM(n,a,b,c,d)*(R1+R2)) }

********************** calculate the island voltage ************************

.FUNC VOLT(n,a,b,c,d) { PN_5(n,a,b,c,d)*(n-5+Q(a,b,c,d))
                            ++PN_4(n,a,b,c,d)*(n-4+Q(a,b,c,d))
                            ++PN_3(n,a,b,c,d)*(n-3+Q(a,b,c,d))
                            ++PN_2(n,a,b,c,d)*(n-2+Q(a,b,c,d))
                            ++PN_1(n,a,b,c,d)*(n-1+Q(a,b,c,d))
                            ++n+Q(a,b,c,d)
                            ++PN1(n,a,b,c,d)*(n+1+Q(a,b,c,d))
                            ++PN2(n,a,b,c,d)*(n+2+Q(a,b,c,d))
                            ++PN3(n,a,b,c,d)*(n+3+Q(a,b,c,d))
                            ++PN4(n,a,b,c,d)*(n+4+Q(a,b,c,d))
                            ++PN5(n,a,b,c,d)*(n+5+Q(a,b,c,d)) }

.FUNC VOLTAGE(n,a,b,c,d) { (echarge/CSUM)*VOLT(n,a,b,c,d)/PSUM(n,a,b,c,d) }

*.PARAM nn = {NOPT(v(1),v(2),v(3),v(4))}


*G1 1 2 VALUE={CURRENT(NOPT(v(1,0),v(2,0),v(3,0),v(4,0)),v(1,0),v(2,0),
*v(3,0),v(4,0))}
E1 5 0 VALUE = { VOLTAGE(NOPT(v(1),v(2),v(3),v(4)),v(1),v(2),v(3),v(4)) }
   ;  Voltage of the island
G1 1 2 VALUE = { CURRENT(NOPT(v(1),v(2),v(3),v(4)),v(1),v(2),v(3),v(4)) }
  ;  Current from source to drain
CT1 1 5 {C1}
CT2 2 5 {C2}
CGATE1 3 5 {CG1}
```

```
151 CGATE2 4 5 {CG2}
152 .ENDS SETis
```