UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

KIM ARAGON ESCOBAR

Parallel Prefix Adder de 4 bits utilizando a tecnologia de autômatos celulares de ponto quântico.

Trabalho de Graduação.

Prof. Dr. Renato Perez Ribas
Orientador

Porto Alegre, junho de 2013.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CI	Circuito Integrado
QCA	Quantum-dot Celullar Automata
PPA	Parallel Prefix Adder
UFRGS	Universidade Federal do Rio Grande do Sul
CAD	Computer Aided Design
RTL	Register transfer level
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
ULA	Unidade Lógica e Aritmética
CA	Cellular Automata
AOI	And-Or-Inverter
API	Application Programming Interface
CSA	Carry Select Adder
RCA	Ripple Carry Adder
CLA	Carry Look Ahead Adder
CMOS	Complementary Metal Oxide Semiconductor

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Nos dias atuais tecnologia não é mais algo que só grandes empresas ou pesquisadores utilizam, esta está no dia a dia de muitas pessoas ao redor do globo. Tal tecnologia comumente é constituída de componentes eletrônicos, sejam eles digitais ou analógicos. Esses componentes integrados estão chegando no seu limite de fabricação, encontrando então um limite física para melhorias. Pesquisadores têm estudado formas alternativas de fabricação, incluindo novas tecnologias para construir esses circuitos eletrônicos.

Em 1993 pesquisadores propuseram uma tecnologia chamada Autômato Celular de Ponto Quântico, ao contrario do que seu nome indica não é para ser utilizada em computação quântica, e sim na computação baseada em números binários como a atual tecnologia, **CMOS**. Desde então muitos propuseram melhorias para essa tecnologia. Um dos focos na academia é a construção de blocos presentes nos circuitos integrados de hoje em dia, como memorias, somadores e registradores. Entre esses somadores e registradores são muito importantes para a unidade lógica e aritmética presente em todos os microprocessadores e microcontroladores da atualidade.

O atual trabalho se propõe a construção de um Parallel Prefix Adder utilizando a tecnologia QCA. A metodologia é explicada e o leiaute para cada um dos blocos que constituem o somador é mostrado. O leiaute para um PPA de 4 bits será mostrado e simulado utilizando **QCADesigner**. Serão realizadas comparações com um artigo publicado, o qual propõem outros somadores utilizando a mesma tecnologia.

Palavras-Chave: Automatos celulare de ponto quantico (QCA), Parallel Prefix Adder (PPA), Nanotecnologia, Somadores.

A 4 bits Parallel Prefix Adder using Quantum-dot cellular automata technology

ABSTRACT

Nowadays technology like computers and cellphones is not unreachable to ordinary people. These devices are mainly assembled with electronic compounds. These compounds are divided in to main families, digital compounds and analog compounds. The man fabrication method is to integrate these blocks in one circuit, but the foundries are reaching the physical fabrication limit and soon it will be impossible to build better circuits only reducing the technological node and raising the scalability. Researchers have been studied technologies with potential to substitute the CMOS technology as the main digital integrated circuits implementation technology.

In 1993 was proposed a new technology called Quantum-dot Cellular Automata, unlike its name could indicates it is not a quantum computation technology, the QCA is based on the same binary computation schema as **CMOS** technology. Since its proposal many researchers have proposed improvements and circuits implementation to these technology. These circuits normally are the main blocks found in microprocessors and microcontrollers like, memories, adders and registers. Adders and registers are very important to build a Aritmetic and Logic Unit.

This works propose an implementation to a 4 bits Parallel Prefix Adder using the QCA technology. The methodology will be explained and the layout to each block will be validated. The layout to 4 bits PPA will be shown as well as the simulation using the **QCADesigner**. After the simulation the actual result is compared with adders proposed by one paper published which proposes three other adders using QCA.

Keywords: Quantum-Dot Cellular Automata (QCA), Parallel Prefix Adder (PPA), Nanotechnology, Adder.

1 INTRODUÇÃO

A tecnologia nos rodeia, em todo o tipo de dispositivos utilizados no dia a dia, como computadores, televisores, *smartphones*, *tablets*, entre outros, portáteis ou não. Toda essa tecnologia está munida de circuitos eletrônicos para fazê-la funcionar. Esses circuitos têm as mais variadas funções, tais como processar e armazenar dados, regular tensão, converter o tipo de energia e de tensão, amplificar o som e o sinal, entre outras. Cada um desses circuitos eletrônicos consome espaço em um chip, o que se torna uma das maiores batalhas da indústria de eletrônica e escalabilidade dos circuitos, pois a cada dia que passa o consumidor exige que seu dispositivo seja mais potente e menor. Por esse motivo, a cada ano que passa a capacidade de integração e o desempenho desses circuitos eletrônicos vêm crescendo e atravessando barreiras para que possamos ter internet, jogos e telefone em um pequeno dispositivo que caiba no bolso.

Os circuitos eletrônicos podem ser divididos em dois grupos que se diferem pela excursão do sinal. Os circuitos analógicos têm a excursão de sinal variando continuamente ao longo do tempo. Já os circuitos digitais têm a excursão do sinal variando discretamente entre dois níveis. Dois circuitos digitais muito importantes que sempre estão presentes nos dispositivos que nos rodeiam são o processador e a memória. De acordo com John von Neumann um computador digital é composto por uma unidade de processamento, uma unidade de controle (processador) e uma memória.

1.1 Projeto de CI Digital

Na época de **John von Neumann** os circuitos eram desenhados à mão, pois além de serem menores e mais simples, não existia ferramentas de CAD para auxiliar no desenho. Após começar a utilizar os transistores de efeito de campo, a capacidade de integração dos circuitos começou a crescer rapidamente a cada ano, e, devido a essa alta capacidade de integração, uma célebre predição na microeletrônica foi dita por **Gordon Moore**. De acordo com ela, o numero de transistores dobra a cada 18 meses. A indústria vem seguindo essa predição como uma meta a ser atingida.

Esse aumento desenfreado na capacidade de integração dos chips fez com que, em pouco tempo, fosse cada vez mais complicado fazê-los à mão, e sem um padrão predefinido de projeto. Contudo, hoje em dia existe um fluxo para projeto digital muito bem consolidado. Esse fluxo é, basicamente, dividido em duas etapas, síntese lógica e síntese física. A primeira se trata de especificação, RTL e mapeamento tecnológico. A segunda consiste em planejamento de planta baixa, colocação das células e o roteamento. Esse é apenas um fluxo simples, pois existem testes e outras verificações que são feitas entre a maioria das etapas.

Como o projeto se trata de um passo a passo que é seguido repetidamente, é conveniente que se tenha várias peças no projeto que sejam bem conhecidas, como, por

exemplo, os módulos que compõem uma unidade lógica e aritmética. O somador é um módulo muito estudado e muito importante, que possui diferentes implementações, desde as mais simples como o *ripple carry adder* até as mais complexas como o *carry look ahead*. Cada uma das implementações foca em um dos critérios bem familiares na microeletrônica digital: área e desempenho.

Quando falamos em melhorar um projeto, normalmente, falamos em melhorar o desempenho ou diminuir a área. Muitas vezes é feito um compromisso entre os dois focos de optimização. Para tais optimizações, pode-se trabalhar em alguns níveis do projeto, analisando-o no sentido *top-down*. Quando a análise se dá no nível comportamental, pode-se melhorar os algoritmos e pensar em otmizar a sua complexidade. No nível RTL pode-se melhorar o desempenho ou a área, eliminado alguma porta lógica que possa estar excedente. Na síntese lógica podem ser criadas restrições para que a própria ferramenta de síntese tente fazer optimizações possíveis e adequadas às restrições. Nas próximas etapas, as optimizações são focadas no planejamento da planta baixa, colocação das células e noroteamento das portas lógicas. Outra alternativa factível é mudar o nodo tecnológico em que se vai implementar o circuito, ou ainda, mudar a tecnologia.

1.2 Motivação e desafios

Toda e qualquer optimização em um projeto tem um custo envolvido, seja ele um custo computacional, de área, de desempenho ou de potência. Usando como exemplo a fase de roteamento, o algoritmo que roda para que a optimização do roteamento seja feita não é determinístico, então, o nível de optimização dependerá das restrições, regras e condições de convergência estipulados para o algoritmo de roteamento. Ainda, caso o projetista não tenha gostado do resultado, pode-se retornar à etapa de colocação das células para reorganizar o circuito de tal forma que o roteamento possa ser aprimorado. Em qualquer fase, pode-se voltar às etapas anteriores para focar em algum dos critérios de optimização. Por exemplo, pode-se tentar mudar o algoritmo utilizado em um somador contido em um processador, focando em maior desempenho, e em vez de utilizar um *ripple carry adder*, pode-se utilizar um *carry look-ahead adder*. A tarefa do projetista é decidir o que ele quer reprimir para melhorar o circuito para um determinado uso.

Outro foco de optimização pode ser a mudança da tecnologia em que será implementado o projeto. Ao mudar o nodo tecnológico utilizado, modifica-se a área utilizada para um mesmo projeto e a velocidade de chaveamento das portas lógicas. Quanto menor o nodo tecnológico, menor será a área para o mesmo projeto e maior será a velocidade de chaveamento. Logo, diminuindo o nodo tecnológico teremos uma melhora em dois critérios de optimização. A diminuição do nodo também acarreta alguns efeitos parasitas (seja a inserção ou o aumento) que podem modificar o comportamento do circuito também. Essa deveria ser a solução sempre que se precisasse aumentar o nível de integração do circuito, mas essa técnica não funciona *ad infinitum* por um motivo: a diminuição do nodo envolve o aprimoramento da tecnologia de fabricação, o que seria possível se não estivéssemos chegando ao limite físico da tecnologia utilizada hoje em dia (CHO *et. al.*, 2007). O tamanho do nodo tecnológico está chegando muito próximo ao tamanho das moléculas utilizadas para compor o dispositivo MOSFET (utilizado na implementação de circuitos integrados com a tecnologia CMOS). Contudo, para todo problema existe uma solução.

O problema da limitação física do MOSFET pode ser solucionado com alguma outra tecnologia para implementar o mesmo comportamento binário ou uma que modifique toda a arquitetura da computação do dado (tecnologias que não se baseiam na representação binária da informação). Existem algumas propostas que seguem a mesma teoria do MOSFET, como por exemplo, o CNFET que utiliza nanotubos de carbono em vez da combinação de metal – óxido – silício. Há também a ideia de fazer um transistor que utiliza apenas um elétron parrando pelo canal, que se chama Single Electron Transistor, que utiliza o princípio do confinamento quântico (EKIMOV e ONUSHCHENKO, 1985) para aprisionar um elétron. Essa técnica de poços potenciais também é empregada na tecnologia quantum-dot cellular automata (QCA) e o seu principal diferencial é a transmissão de informação sem corrente.

A primeira proposta formal do quantum-dot cellular automata foi feita por Lent (LENT *et al*., 1993). O QCA foi baseado na teoria do confinamento e do tunelamento quânticos. Temos basicamente quatro pontos de confinamento quântico, como visto na figura 1.1, e cada célula possui dois elétrons, que pela lei de Coulomb ficarão o mais distante possível. A representação da informação se dá através das duas configurações possíveis de menor energia na célula – as duas possíveis diagonais – como mostrado na figura 1.1. Com uma célula é possível representar 1 ou 0 lógicos. Se juntarmos mais células e definirmos uma ou mais entradas e uma saída, o seu estado de menor energia pode ser interpretado como uma função lógica. Os dispositivos mais básicos utilizados são o fio, os inversores e as portas majoritárias (TOUGAW e LENT, 1994).

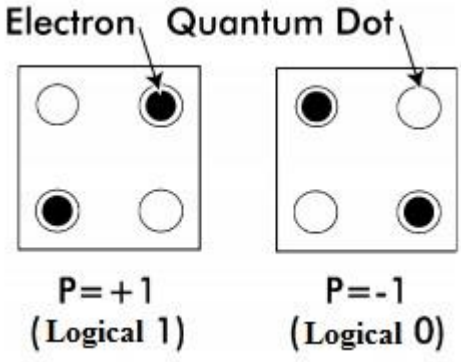

Figura 1.1 - Célula QCA com a codificação binária e seu respectivo potencial (ZHANG, 2004).

Desde sua formalização, muitos pesquisadores têm implementado circuitos digitais como memórias (WALUS et al., 2004), somadores (CHO e ZWARTZLANDER, 2007) e flip flops (TORABI, 2011). Os simuladores utilizados para se fazer implementações são: MÁQUINAS, QBART (OTTAVI et al., 2006) e QCADesigner (WALUS et al., 2004). Ao procurar por MÁQUINAS e QBART, nada foi encontrado na Internet, portanto esses projetos foram provavelmente descontinuados e atualmente são difíceis de achar na Internet, enquanto o QCADesigner, mesmo descontinuado, ainda continua disponível (WALUS, 2002), e será melhor discutido no Capítulo 2.2. Esse trabalho de implementação de circuitos vai desde a tradução de equações Booleanas para equações majoritárias até o desenvolvimento de novos algoritmos ou a adaptação de antigos, que melhor se adaptem à tecnologia do QCA (PERRI e COSTELLO, 2012).

Os somadores possuem diferentes algoritmos para serem implementados. Cada um com seu propósito e bom para algum critério especifico. O *ripple carry adder*, por exemplo, é um somador simples, ideal para soma com poucos bits, mas o tempo de

propagação não escala bem com o aumento de bits, pois o sinal de "vai um" passa por cada um dos somadores completos antes de chegar ao seu valor final. Outro somador bastante usado é o *carry select adder*, que embora seja um somador cujo tempo de propagação escala bem com o aumento do número de bits, a sua área não escala bem por duplicar a soma a cada **x** bits (uma soma para vem um sendo 1 e outra sendo 0). Temos ainda um somador maleável com capacidade de adaptação de acordo com o projeto: o *parallel prefix adder*. Seu algoritmo contempla uma etapa que pode ser otimizada em relação a área, a desempenho ou ainda, e mais importante, a otimização híbrida, onde área e desempenho têm pesos de importância. Esse algoritmo ainda tem nomes para algumas configurações consagradas, como, por exemplo, a configuração que maximiza o desempenho e mantém o número de conexões em 2 do circuito nomeado Kogge-Stone (KOGGE e STONE, 1973), ou ainda uma solução com menor caminho crítico e menos área que Kogge-Stone, mas onde o número de conexões é maior que 2, chamada Ladner-Fischer (LADNER e FISCHER, 1980). Outra especificação do PPA é o *carry look ahead*, que faz um agrupamento focando em área e desempenho, mas perdendo a flexibilidade. Essa solução é muito boa para somadores com grande número de bits. Vendo todas essas possibilidades, percebemos facilmente que o PPA é um ótimo somador para muitos tipos de projetos.

1.3 Proposta

Em sua maioria, as implementações de somadores com tecnologia QCA baseiam-se na tradução da equação Booleana para equação majoritária. Ainda assim, alguns trabalhos propõem otimizações nessa tradução para um melhor uso do potencial da tecnologia QCA (PERRI e CORSELLO, 2012), (ZHANG et al., 2004). É importante que haja muitos algoritmos implementados e testados utilizando o QCA, pois quando houver o uso extensivo em projetos mais complexos, esses preferirão utilizar algoritmos cujas implementações já tenham sido extensivamente estudadas e comprovadas para a tecnologia alvo, nesse caso, a QCA.

Neste trabalho é apresentada uma implementação para um PPA de 4 bits. A metodologia que deve ser utilizada, partindo de seu algoritmo com equações Booleanas e finalizando com o leiaute, também é apresentada. O objetivo é demonstrar a implementação do somador e mostrar que esse método pode ser estendido para qualquer número de bits e para qualquer configuração do PPA. Os resultados de simulação demonstraram pontos de comparação importantes entre a proposta e somadores da literatura.

1.4 Estrutura do texto

Para a melhor compreensão do desenvolvimento deste trabalho é necessário o conhecimento sobre como é feita a computação da informação utilizando o QCA. Tanto sua abstração lógica, mais simples, quanto sua modelagem física. Essa revisão técnica está desenvolvida no subcapítulo 2.1, separado em duas seções, cada uma explicando um dos níveis de abstração. A implementação da proposta desse trabalho é feita utilizando a ferramenta QCADesigner. Uma breve explicação sobre o funcionamento dessa ferramenta de CAD é feita no Subcapítulo 2.2. Sendo o foco desse trabalho a implementação de um PPA, seu algoritmo, explicações e alguns exemplos estão disponíveis no Capitulo 2.3.

No capítulo 3 está desenvolvida detalhadamente a proposta desse trabalho. Outro ponto apresentado no referido capítulo é a metodologia completa utilizada na implementação do PPA, a aplicação passo a passo e os leiautes para cada porta lógica.

A simulação, juntamente com seus resultados são comentados no capítulo 4, onde também é encontrada a lista de parâmetros utilizados no QCADesigner para tal. São discutidas as simulações para cada porta lógica e para o PPA de 4 bits, que é comparado com outros somadores de 4 bits da literatura. Por fim, duas equações calculando o atraso máximo e mínimo para n bits do PPA são utilizadas para extrapolar a comparação para somadores de 8, 16, 32 e 64 bits da literatura.

Por fim, concluímos o trabalho no Capítulo 5, ressaltando suas principais contribuições. São ainda comentadas algumas dificuldades de implementação e discutidos alguns possíveis meios de eliminá-las. Finalmente, possíveis trabalhos futuros são enunciados.

2 BACKGROUND

2.1 Autômato Celular de ponto quantico

A tecnologia foi proposta por Lent (LENT et al., 1993), e para se chegar a tal ideia foram utilizados os conceitos de pontos quânticos e a lei da interação elétrica de coulomb. Após equacionar o comportamento de interação entre várias células é possível mapear um comportamento análogo de um autômato celular, onde a regra de interação entre as células é dada pelo Hamiltoniano, que é o somatório de todas as energias do sistema, (apresentado no capítulo 2.1.1). Diferentemente dos autômatos celulares, o interesse nesse caso não é pelos estados intermediários do autômato, e sim pelo seu estado final – que é chamado de *ground state* e é o estado de menor energia possível no sistema dada uma entrada inicial. Um importante ganho ao utilizar essa tecnologia é o fato da transmissão de informação ser dada por campos, ou seja, não dissipação por causa de corrente elétrica. Embora a interação de campos seja uma vantagem, ela traz um revés: a possibilidade de duas células interagirem e estragarem a computação. Isso pode ser resolvido sincronizando as células e condicionando quando elas podem fazer a transição de estados. Esse sincronismo é chamado de zona de relógio (LENT e TOUGAW, 1997). Atualmente os projetos utilizam quatro zonas de relógio na implementação dos circuitos.

Para um melhor entendimento do funcionamento do QCA, o melhor meio de se apresentar a teoria é separando a explicação física da lógica. Na primeira é apresentado o equacionamento desde sua base nas leis de interação de Coloumb e a equação da onda, até sua equação de transferência do sinal entre as células. A segunda apresenta o ponto de vista da computação da informação utilizando lógica binária com as equações majoritárias. Essa última é a ideia para implementar circuitos digitais, pois partindo do pressuposto que o mapeamento da abstração física para a lógica esteja correta, é muito simples desenvolver circuitos partindo apenas de sua equação majoritária, em vez de utilizar a função de transferência. O mapeamento do resultado da função de transferência para lógica binária é apresentado na figura 1-1, onde os P’s são a saída da função de transferência (será abordado no capítulo 2.1.1). A figura 2-1 mostra um diagrama com o mapeamento das duas abstrações.

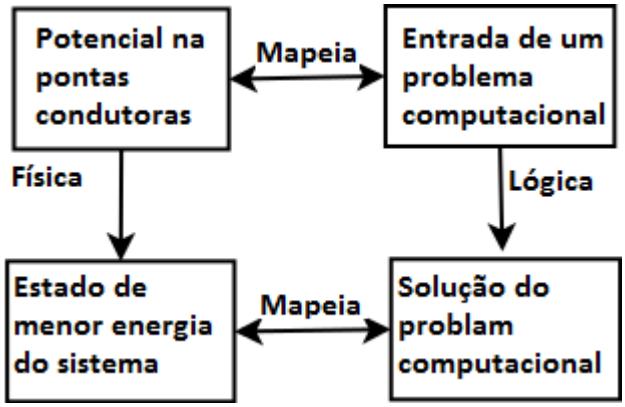

Figura 2.1 – Mapeamento tecnológico da abstração física para a lógica (LENT e TOUGAW, 1994).

2.1.1 Modelagem física

Nesta etapa o equacionamento teórico para a tecnologia será desenvolvido. A intenção não é ensinar mecânica quântica, então o foco será apenas a tecnologia e o desenvolvimento partirá do hamiltoniano para chegar à formulação da polarização. Para melhor trabalhar, supomos que o tunelamento para fora da célula é impossível, os graus de liberdade internos ao ponto quântico serão ignorados. A descrição começa analisando a formula apenas para os pontos quânticos de dentro de uma célula, como mostrado na figura 2.2. Será utilizado o hamiltoniano do tipo Hubbard-extendido (HUBBERD, 1963), mostrado a seguir:

$$H^{cell} = \sum_{i,\sigma}(E_0 + V_i)\hat{n}_{i,\sigma} + \sum_{i>j,\sigma} t_{i,j}\left(\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + \hat{a}^{\dagger}_{j,\sigma}\hat{a}_{i,\sigma}\right) + \sum_{i} E_Q \hat{n}_{i,\uparrow}\hat{n}_{i,\downarrow} + \sum_{i>j,\sigma,\sigma'} V_Q \frac{\hat{n}_{i,\sigma}\hat{n}_{i,\sigma'}}{\left|\mathbf{R}_i - \mathbf{R}_j\right|}. \quad (1)$$

O primeiro somatório é relacionado à energia no ponto quântico para cada ponto i, onde σ representa o spin, E_0 é a energia no estado de menor energia (ground-state) para um ponto que possui um eletron com massa efetiva de $m^*=0.067m_0$, V_i é o potencia devido a cargas externas a célula, e $\hat{n}_{i,\sigma}$ é o operador de numero do ponto *i* com spin σ. O segundo termo é relacionado ao tunelamento de elétrons de um ponto a outro, onde $t_{i,j}$ é a potencia de acoplamento de um ponto i a um ponto *j*, sendo que só é possível para pontos vizinhos, como mostrado na figura 2.2, a expressão $\hat{a}^{\dagger}_{i,\sigma}(\hat{a}_{i,\sigma})$ cria (aniquila) um eletron de spin σ no ponto i. O terceiro somatório é a energia necessária no ponto i para se colocar dois elétrons com spins opostos no local. O último termo representa a interação de *Coulumb* entre os diferentes pontos dentro da célula, onde V_Q é força de acoplamento capacitivo e R_i é o vetor posição do ponto i na célula.

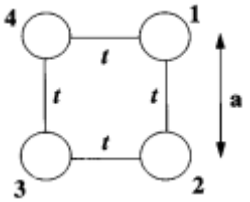

Figura 2.2 – Estrutura da célula QCA e seus potencias de acoplamento t.

Agora podemos resolver a equação de Schrödinger independente do tempo, $|\psi_n >$ para um estado n:

$$H^{cell}|\psi_n\rangle = E_n|\psi_n\rangle. \quad (2)$$

Uma estratégia adotada para medir o grau com que a densidade de carga esta alinha diagonalmente foi a equação a seguir:

$$P = \frac{(p_1 + p_3) - (p_2 + p_4)}{p_1 + p_2 + p_3 + p_4}, \quad (3)$$

essa equação é chamada de equação de polarização onde p_i é a densidade de partícula única que representa a expectativa de existir uma partícula no ponto i e é dado por:

$$p_i = \sum_{\sigma} \langle \psi_0 | \hat{n}_{i,\sigma} | \psi_0 \rangle, \quad (4)$$

onde $|\boldsymbol{\psi_0}\rangle$ é a função de onda para duas-partículas no estado de menor energia, obtido através de (2).

Esta formulação é suficiente para uma célula, mas normalmente possuímos mais de uma célula interagindo. Para adaptarmos as equações para o caso de mais de uma célula interagindo precisamos adicionar outro *Hamiltoniano*, que representa a interação de *Coulomb* entre cada ponto dentro da célula e cada ponto de todas as células vizinhas,

$$H_{inter}^{cell} = \sum_{i,j,\sigma,\sigma',m\neq \mathrm{k}} V_Q \frac{\hat{n}_{i,\sigma} p_{j,\sigma'}^{m}}{\left|\mathbf{R}_i - \mathbf{R}_j^m\right|}, \quad (5)$$

nesta equação temos que *k* é a célula a qual esta tentando se contabilizar a polarização e *m* são todas as outras células vizinhas e $\boldsymbol{p_{j,\sigma}^{m}}$ será dado por:

$$p_{i,\sigma}^{m} = \sum_{\sigma} \langle \psi_{m0} | \hat{n}_{i,\sigma}^{m} | \psi_{m0} \rangle. \quad (6)$$

Para termos a solução para o estado da equação de onda correspondente ao sistema com os vizinhos basta adicionar a equação (5) à equação (2) então teremos:

$$\left(H^{cell} + H_{inter}^{cell} \right) |\psi_n\rangle = E_n|\psi_n\rangle, \quad (7)$$

utilizando a nova equação de onda para duas partículas no estado de menor energia na função de polarização e assim obtemos a polarização dependente das células vizinhas. O próximo passo seria resolver esse problema para a equação de *Schrödinger* dependente do tempo, mas por sua complexidade demasiada, deixaremos para o leitor consultar materiais mais avançados como o artigo (TOUGAW e LENT, 1996).

2.1.2 Abstração lógica

A teoria por trás do QCA é bem complexa, e os projetistas apenas assumem que ela funcione. Através do mapeamento das abstrações, que é mostrado na figura 2.1, o projetista não precisa se preocupar com a complexidade das equações da onda para projetar circuitos com o QCA. É suficiente saber que a computação acontece quando o circuito atinge o estado de menor energia e assumir uma das células como sendo a entrada e outra célula a saída, assim, o estado de menor energia fará com que a célula saída assuma um dos dois valores de polarização com relação à entrada e à função lógica em que está representado o circuito. Os valores de polarização são mapeados para 1 e 0, como mostrado na figura 1.1.

Assim como são utilizadas portas básicas como AND, OR e INV no CMOS, no QCA também existem três dispositivos básicos com os quais os circuitos mais complexos são construídos, propostos em Tougaw (1994). O primeiro deles é o fio, mostrado na figura 2.3(a), que é a conexão entre as células, assim como o fio que conecta os circuitos no CMOS. O segundo é o inversor, mostrado na figura 2.3(b), que, logicamente, é igual ao inversor CMOS. A porta majoritária, mostrada na figura 2.3(c), tem a mesma função lógica de um votador (muito utilizado em circuitos tolerantes a falhas), ou seja, possui três entradas e, se duas delas tiverem um valor igual, a saída assume aquele valor.

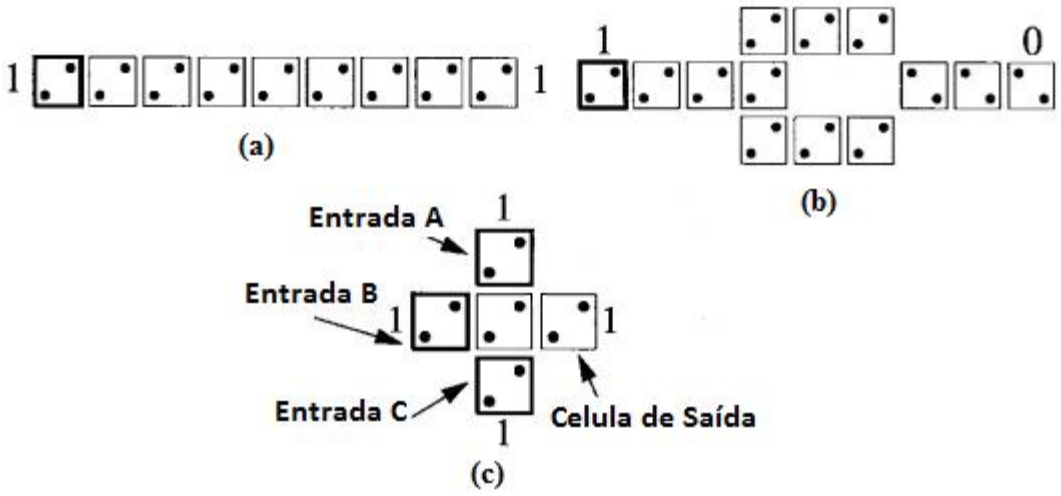

Figura 2.3 – Dispositivos básicos: (a) Fio, (b) Inversor e (c) Porta majoritária (LENT e TOUGAW, 1994).

A porta majoritária é a base para a construção de circuitos em QCA, assim como **NAND** ou **NOR** é a base para construção de circuitos em CMOS, pois adicionando a inversão e os fios é possível implementar qualquer função lógica. A modelagem para a porta majoritária em forma de equação será saída = M(A,B,C), utilizando a figura 2.3(c). Representando M(A,B,C) como uma expressão booleana teremos AB+BC+AC. Através dessa expressão é possível ver que, ao fixarmos uma das entradas em 0, teremos o comportamento de uma **AND** e ao fixarmos uma das entradas em 1, teremos o comportamento de uma **OR**.

A literatura também fala de outros dispositivos menos convencionais. (Lent *et al.* ,1993) propõem algumas estruturas, entre elas, o canto (*corner*), figura 2.4(a) e Célula de conexão, figura 2.4(b), que embora sejam estruturas utilizadas, não são mais referidas pela literatura como dispositivos, e sim apenas como variações do fio. O dispositivo AOI (figura 2.5), proposto por (MOMENZADEH et al., 2005), também não é muito utilizado na literatura.

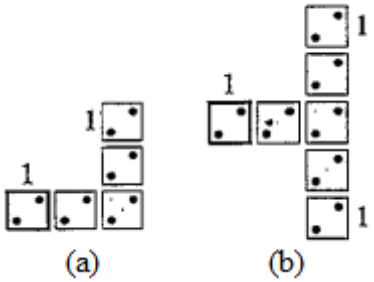

Figura 2.4 - Estruturas pouco referenciadas na literatura: (a) canto e (b) Célula de conexão (LENT *et. al.*, 1993).

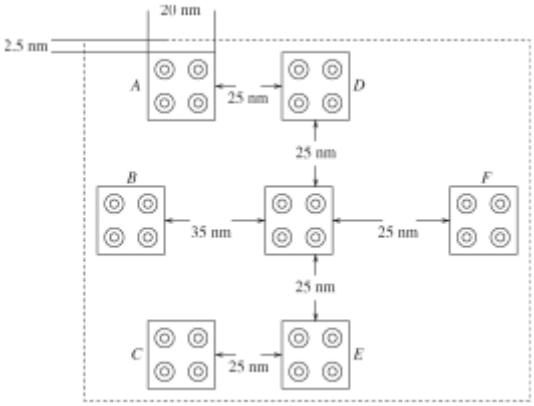

Figura 2.5 – Estrutura central do dispositivo AOI (MOMENZADEH *et. al.*, 2005).

Até este ponto, o leitor é capaz de montar o leiaute de portas simples que implemente funções booleanas simples, como a inversão, a operação de voto, **AND** e **OR**. Para gerar circuitos complexos, como uma **XOR** ou flip-flop, precisamos nos preocupar com a natureza dual do QCA, ou seja, o sinal pode ser transmitido para qualquer um dos dois lados. Assim, ao conectarmos portas majoritárias em série através de um fio, não se pode garantir a integridade do sinal, pois no meio do fio o estado "gravado" na segunda porta majoritária pode influenciar o sinal a voltar, tendo assim um resultado incorreto na saída. A solução para esse problema é a introdução de um sinal que faça a sincronização de cada um dos níveis, de tal modo que o nível subsequente sempre esteja preparado para receber um sinal, mas só possa chaveá-lo após a certificação de que o sinal anterior esteja estável. Para fazer tal sincronização foi proposta por Lent e Tougaw (1997) a introdução de um sinal de relógio. É utilizado quatro fases principais, sendo elas: transição, relaxação, liberação e contenção. A tabela 2.1 enumera e descreve cada uma dessas fases. Adicionalmente, na definição do leiaute, podem-se instanciar quatro zonas de sinal, e cada uma estará sempre em uma fase distinta. Esse sinal é representado por uma onda quadrada (igual ao sinal de relógio utilizado em circuitos síncronos no **CMOS**) e a cada 90º temos uma fase diferente. A figura 2.6 mostra cada uma das zonas de sinal e mostra a progressão das células no decorrer do tempo.

Fase do sinal	Descrição
Transição	A célula assume a polarização da célula anterior.
Relaxação	A célula permanece não polarizada.
Libertação	A célula é despolarizada.
Contenção	A célula mantem a polarização atual.

Tabela 2.1 – Descrição das fases do sinal de relógio.

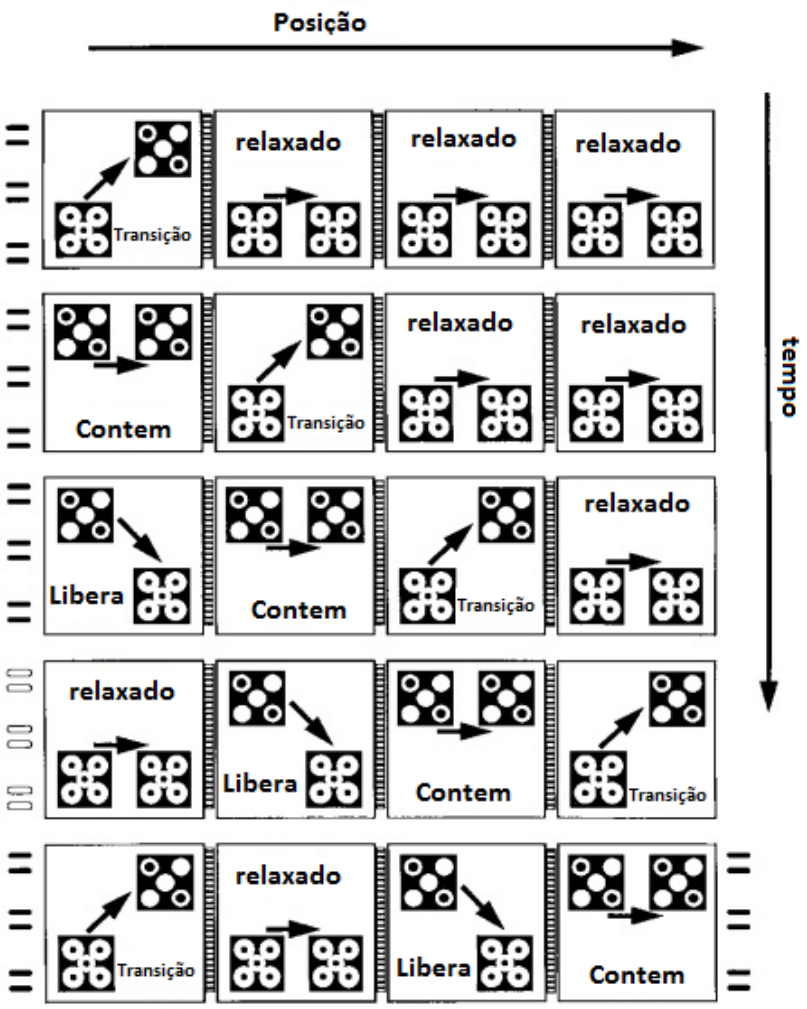

Figura 2.6 – Zonas de relógio progredindo no tempo.

O detalhe final para implementar qualquer circuito é como fazer o cruzamento entre os fios. Inicialmente Tougaw e Lent (1994) propuseram utilizar as células giradas em 90º, como mostrado na figura 2.7, pois essas, quando atravessam um caminho com células "normais", não modificam a informação e vice-versa, como mostrado na figura 2.8. Através de um estudo utilizando modelagem com grafos Lusth e Jackson (1996) chegaram à conclusão de que essa configuração utilizando células giradas pode gerar resultados indesejados: quanto maior o fio, maior será o número de estados de menor energia, e para mesma entrada haverá múltiplas saídas possíveis, podendo trazer esses resultados insatisfatórios. Há uma outra alternativa para cruzar os fios, que é o uso do cruzamento multicamada, proposto por Krause et al. (1998) e Gin et al. (1999) que

consiste em utilizar outras camadas para passar por cima dos fios, como duas especificidades, como é feito no CMOS. Primeiramente foi averiguado que é necessário, no mínimo, uma separação de 2 camadas entre cada célula e a cada camada para cima o sinal é invertido, como mostrado na figura 2.9.

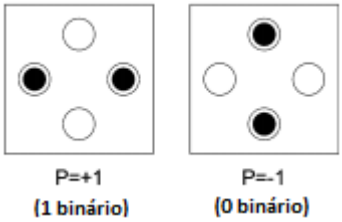

Figura 2.7 – Polarização da célula girada em 90º (CHO at. al., 2004).

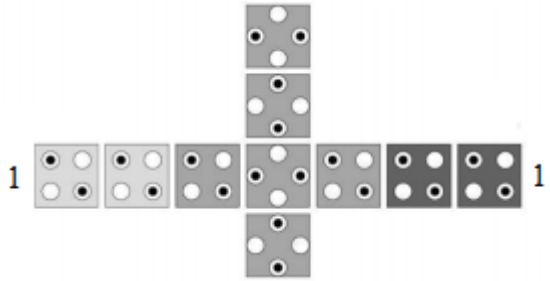

Figura 2.8 – Cruzamento de fios utilizando a célula girada em 90º (CHO *et. al.* 2004).

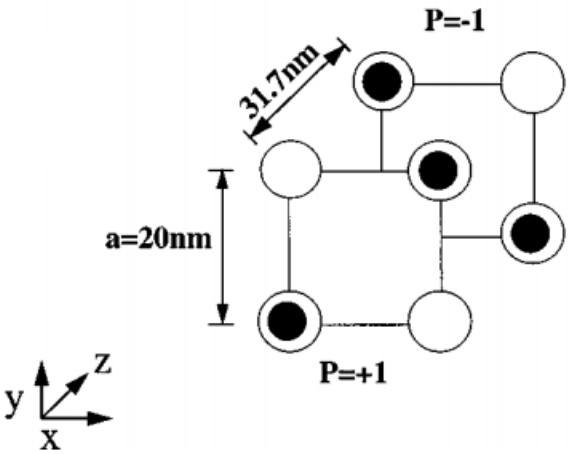

Figura 2.9 – Células em camadas diferentes.

Com todos os detalhes cobertos, é possível agora visualizar um circuito para entender na prática alguns desses conceitos. Na figura 2.10, é mostrado o diagrama da seguinte função majoritária *M(M(A, not(B), 0), M(not(A), B, 0), 1)* que representa a função **XOR**, na figura 2.11 podemos ver o leiaute da **XOR**, com cada um dos dispositivos marcados, onde os círculos marcam os inversores e os quadrados, as portas majoritárias, as duas da esquerda representando a função **AND** e a da direita representando a função **OR**. Ainda na figura 2.11, os quadrados à esquerda que não possuem pontos representam um cruzamento multicamada, e os tons de cinza representam, cada um, uma zona de relógio. Pode-se notar que a porta **XOR** leva um ciclo de relógio para computar as entradas. O leiaute foi montado utilizando a ferramenta QCADesigner (WALUS, 2002), a única ferramenta livre para simulação, elétrica e lógica de QCA hoje em dia.

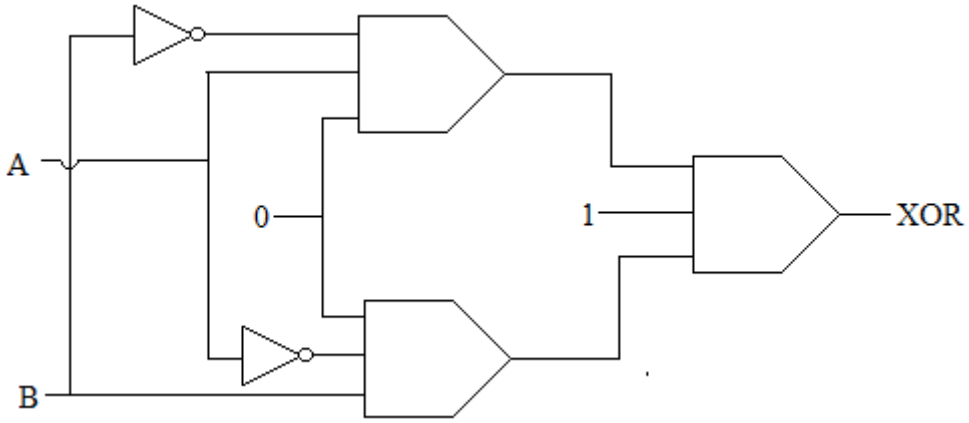

Figura 2.10 – Diagrama para a função **XOR**.

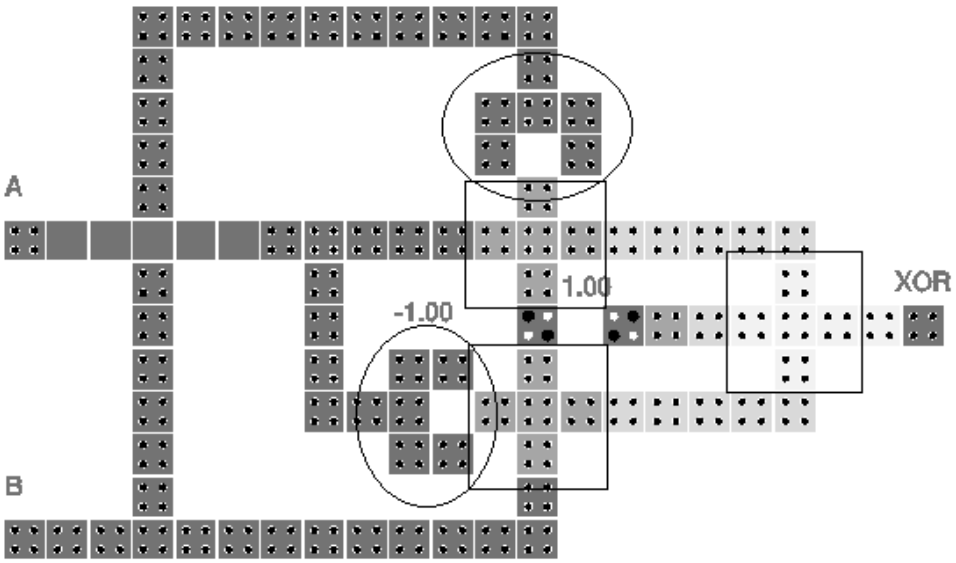

Figura 2.11 – Leiaute para a função **XOR**.

2.2 QCADesigner

Esse simulador elétrico/lógico foi inicialmente desenvolvido no laboratório ATIPS da Universidade de Calgary, pelo pesquisador Konrad Walus. O projeto tinha por finalidade ser um simulador totalmente adaptável, podendo se acoplar módulos de interesse para trabalhar junto com o simulador. Por esse motivo, foi desenvolvido em licença GPL, utilizando C/C++ e GTK+. Sua API é muito bem documentada no site do projeto (WALUS, 2002), que hoje em dia se encontra hospedado no site da Universidade da Columbia Britânica, atualmente encontra-se na versão 2.0.3 e sua última atualização saiu em 2005.

Com essa simulação é possível realizar a simulação com 2 tipos de precisão, o vetor de coerência é a simulação mais preciso que o software oferece, e é baseada no Hamiltoniano de dois estados. Esse é capaz de modelar a interação entre as células utilizando modelos da mecânica quântica. Os parâmetros que podem ser escolhidos nesse módulo encontram-se na tabela 2.2. O segundo método é a aproximação biestável,

baseado na resposta de célula para célula, por esse motivo não sendo tão preciso, mas suficiente se o interesse for apenas o comportamento lógico do sistema. Os parâmetros que podem ser escolhidos para ele se encontram na tabela 2.2.

A montagem do leiaute utilizando o simulador é direta, no momento em que se está montando o circuito, o mesmo já é o leiaute. A figura 2.11 mostra o leiaute produzido pelo **QCADesigner.** O simulador oferece algumas ferramentas para construir o leiaute: possibilidade de seleção, inserção de uma célula, inserção de um conjunto de células, rotação de células, possibilidade de mudar o desenho da célula para célula vertical e célula de cruzamento em multicamada (que servem para mostrar onde mudam as camada, mostradas na figura 2.12), possibilidade de copiar, mover e espelhar uma célula, dar zoom e colocar uma régua. Esse software ainda possibilita o trabalho em múltiplas camadas. Na figura 2.13 temos uma imagem das camadas utilizadas no leiaute da XOR descrita no capítulo anterior. Na figura temos seis camadas: a primeira é o substrato, a segunda é apenas uma camada onde se coloca anotações (não física), a terceira é a camada principal, a quarta e quinta são apenas vias e a última camada é a segunda por onde se faz o cruzamento dos fios. É possível também agrupar entradas em barramentos, como indicado na figura 2.14.

Parâmetros	**Vetor de coerência**	**Aproximação biestável**
Numero de Amostras		x
Tolerância de convergência		x
Raio de efeito [nm]	x	x
Permissividade Relativa	x	x
Sinal de relógio Alto (J)	x	x
Sinal de relógio baixo (J)	x	x
Troca do valor do sinal de relógio	x	
Fator de amplitude do sinal de relógio	x	x
Separação entre camadas	x	x
Numero máximo de iterações por amostra		x
Temperatura (K)	x	
Tempo de Relaxação (s)	x	
Tempo do passo (s)	x	
Tempo total de simulação (s)	x	

Tabela 2.2 – Tabela com os parâmetros existentes em cada simulação.

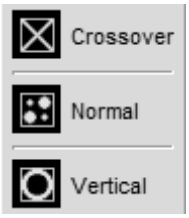

Figura 2.12 – Tipos de células.

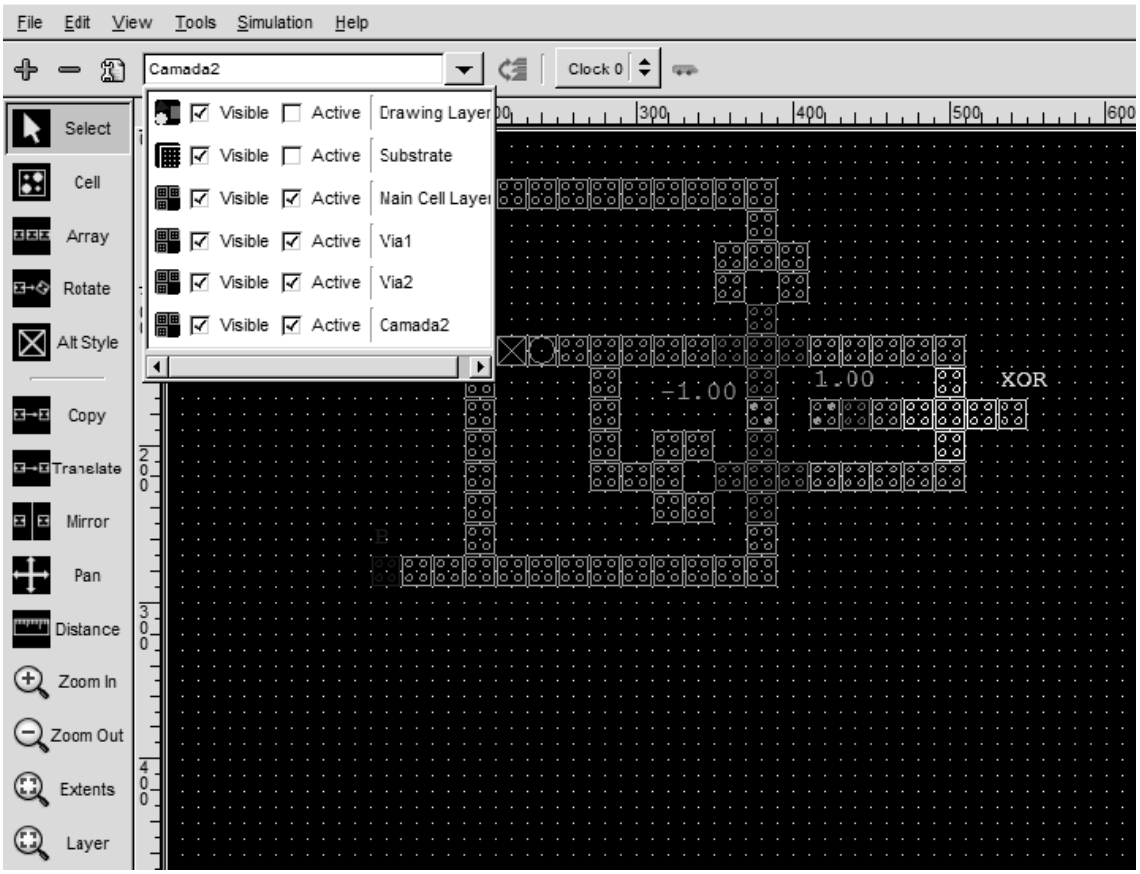

Figura 2.13 – Possibilidade de visualização de cada camada.

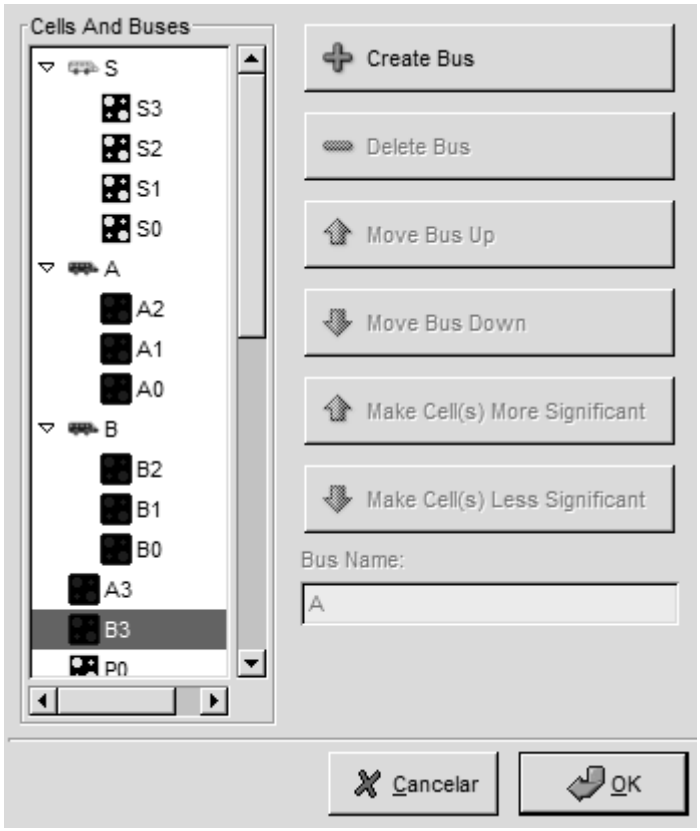

Figura 2.14 – Opções para criar um barramento.

O **QCADesginer** permite analisar a simulação em dois níveis diferentes, o lógico e o elétrico. A única diferença entre os dois é o modo de olhar a simulação, pois eles são feitos simultaneamente. A figura 2.15 mostra a simulação para a porta **XOR**. Na janela de simulação à esquerda temos os sinais simulados, divididos em sinais de entrada, seguidos pelos sinais de saída e por fim as quatro zonas de relógio. À direta os sinais são representados por nível alto e baixo e do lado esquerdo dos sinais temos os níveis energéticos para cada um deles. O eixo x para essa simulação é o número de cada amostra, podendo ser interpretado por unidades de tempo.

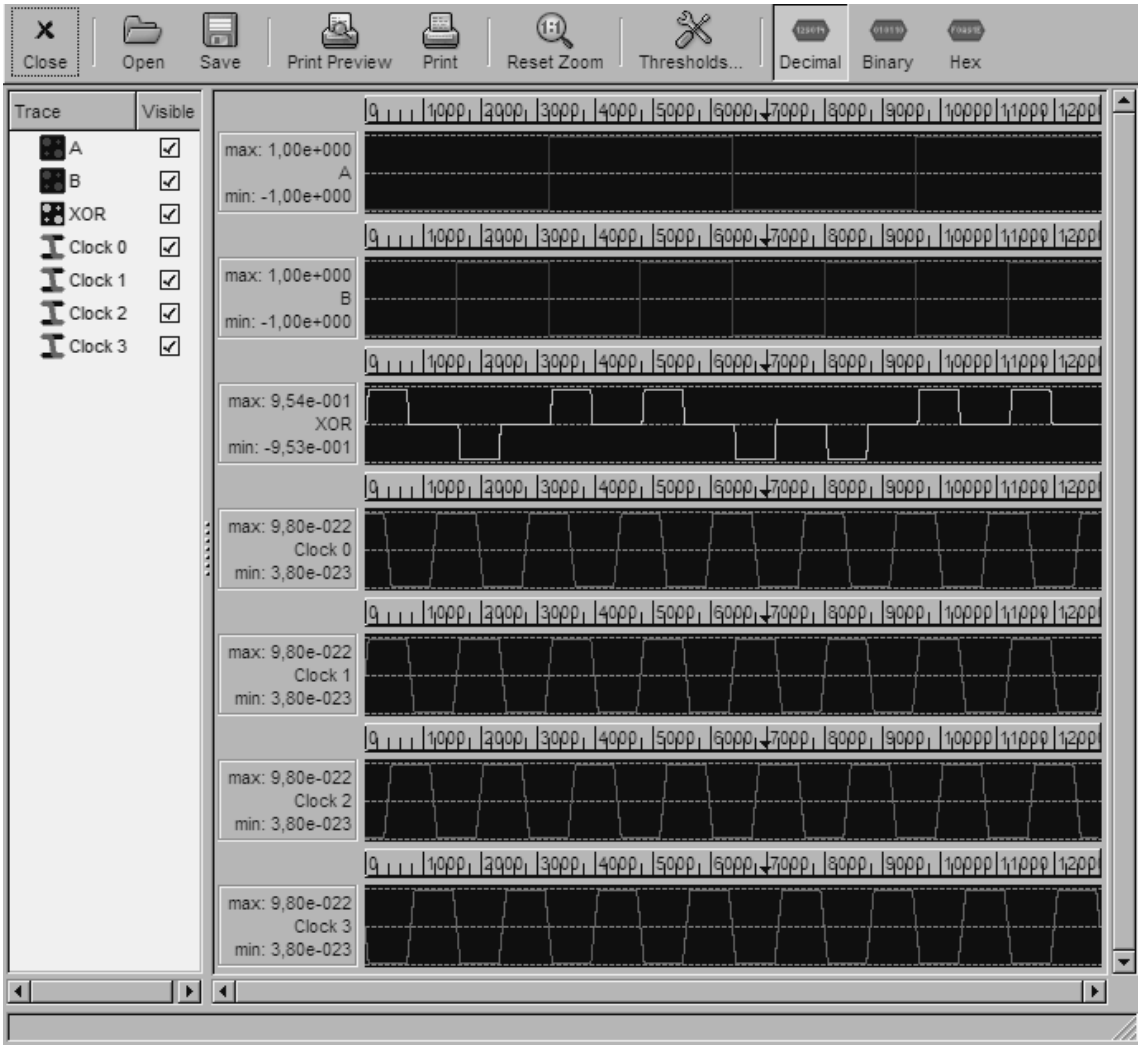

Figura 2.15 – Tela de simulação do **QCADesigner**.

O QCADesginer é, sem dúvida, um excelente software, e atualmente o único disponível gratuitamente na internet para simular circuitos QCA. Contudo, ele possui algumas falhas. A primeira delas é que a biblioteca GTK utilizada é muito antiga, com funções desatualizadas, causando diversos problemas. Embora ele possua suporte para rodar no Windows, não é possível salvar os circuitos, pois acontece um tipo de falha gráfico. Esse software funciona no Linux, mas apenas utilizando um tipo de cerne, especificamente, o cerne utilizado pela versão 15 do Fedora. Para acessar todas as funções disponíveis é preciso utilizar Windows e Linux em sinergia, pois algumas funcionam no Linux e outras, no Windows. Atualmente, esse projeto está parado, não possui nenhuma atualização desde 2006, de acordo com o site oficial. Apesar de todas essas falhas, ainda continua sendo a melhor alternativa para aqueles que querem testar circuitos QCA sem precisar criar uma ferramenta de CAD própria.

2.3 Parallel Prefix Adder

Para entender o funcionamento do PPA, primeiramente é preciso entender os princípios de propagação e geração na soma, que são também utilizados pelo *Skip Adder*. A geração de um "vai um" acontece quando independentemente do sinal de "vem um" a soma de um bit gerará um "vai um", ou seja, quando estamos somando 1 com 1. A propagação de um "vai um" é quando a soma de dois bits consegue levar

adiante o “vem um”, ou seja, quando houver pelo menos um 1 na soma. Na figura 2.16, podemos ver um exemplo mostrando tanto a geração quanto a propagação do “vai um”. Vemos então que a primeira soma dentro do retângulo tem capacidade de propagar o *Cin*, seja ele 0 ou 1. No segundo quadrado pode-se ver que ele irá gerar o vai 1, independente do “vem um” da soma anterior.

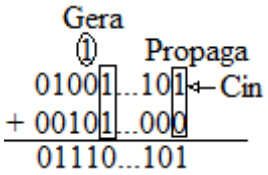

Figura 2.16 – Exemplo da propagação e geração do “vai um”

O algoritmo do PPA é dividido em quatro etapas, sendo a primeira o cálculo das propagações e da geração por cada soma individual. A segunda etapa consiste no cálculo da geração e propagação para agrupamentos de bits partindo do 0, ou seja, precisamos de todos os possíveis conjuntos de 0 até n, onde n varia de 1 até o número total de bits. A terceira é o cálculo dos “vai um” baseando-se nos grupos de propagação e geração da etapa anterior. A etapa final é a computação da soma baseando-se no XOR entre a propagação de um bit e o “vai um” do bit anterior.

A propagação é computada fazendo-se uma **XOR** entre os operandos de mesmo bit (i. e. $A_0 \oplus B_0$). A geração do vai um é calculada fazendo-se o AND entre os operandos de mesmo bit (i. e. $A_0 \bullet B_0$). Em seguida para conseguir propagação para o grupo i até i+1, basta sabermos se as duas propagam (i. e. $Pi_{+1} \bullet P_i$). Para o cálculo da geração para o grupo de i até i+1, precisamos saber se i+1 gera “vai um” ou se i+1 propaga e i gera “vai um” (i. e. $G_{i+1} + P_{i+1} \bullet G_i$). Sabendo como calcular para i+1 e i, basta utilizarmos as funções recorrentemente para bits contíguos, até chegar na combinação do bit 0 até o n, para todo n entre 0 e o número máximo de bits, para melhor entendimento a figura 2.17 mostra um grafo onde os círculos com PGx representam o cálculo da propagação e da geração para o bit x, e os círculos com PGx,y representam o cálculo da propagação e geração de grupo de y até x. Por serem possíveis diversas combinações para se chegar na propagação e geração de 0 até n, essa etapa onde é montada a árvore de propagação e geração é o principal foco de otimização e a principal vantagem do PPA. Para o cálculo do “vai um” do bit i precisamos saber se a soma até o bit i faz com que ele gere 1 ou se a propagação até o bit i é capaz de propagar o “vem um” de entrada (Cin) (i. e. $G_{i,0} + P_{i,0} \bullet C_{in}$). Por fim, para a soma do bit i basta fazer uma XOR entre a propagação desse bit com o “vai um” do bit anterior (i. e. $P_i \oplus Cout_{i-1}$). A tabela 2.3 mostra de forma resumida as equações utilizadas em cada etapa do algoritmo.

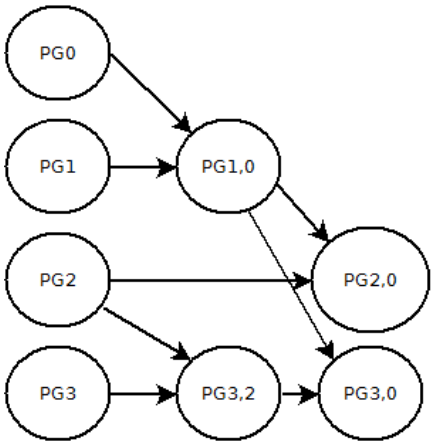

Figura 2.17 – Árvore de propagação e geração.

Etapa	Equações utilizadas
1 - Gerações e Propagações individuais	$P_i = a_i \oplus b_i \mid G_i = a_i \bullet b_i$
2 – Gerações e Propagações de grupo	$G_{i+1,i} = G_{i+1} + P_{i+1} \bullet G_i \mid P_{i+1,i} = P_{i+1} \bullet P_i$
3 – Cálculo dos "Vai Um"	$Cout_i = G_{i,0} + P_{i,0} \bullet Cin$
4 – Cálculo da Soma	$S_i = P_i \oplus Cout_{i-1}$

Tabela 2.3 – Resumo do algoritmo do PPA.

Por se tratar do foco de optimização a árvore de propagação e geração já teve várias estruturas nomeadas. Uma estrutura bem conhecida é a Kogge-Stone que alcança a menor altura da árvore mantendo o numero de conexões sempre em dois (KOGGE e STONE, 1973). O somador intitulado Brent-Kung (BRENT e KUNG, 1987) diz ter a menor área para um dado tempo de resposta, mas foi provado em (ESCOBAR et al., 2010) que não possui a menor área para aquela dada altura. A estrutura Ladner-Fischer consegue encontrar a menor área para a menor altura possível, dado o tamanho dos operandos (LADNER e FISCHER, 1982). Um algoritmo que calcula a área mínima e a altura mínima, dado um grau de importância (em porcentagem) para os dois critérios, foi produzido em (ESCOBAR et al., 2010). Algumas equações definidas nesse artigo foram utilizadas na análise dos resultados no capítulo 4.

3 PROPOSTA E IMPLEMENTAÇÃO

3.1 Proposta

Um grande desafio para os projetistas de circuitos QCA é a falta de metodologia de projeto. Embora um projeto não necessite de uma metodologia para ser feito, a metodologia garante a qualidade do projeto. Alguns métodos já foram propostos na literatura como, por exemplo, (NIEMER e KOGGE, 2001) que propõem um método para melhor utilização da área disponível no projeto. Ele tenta separar as células utilizadas para comunicação e as para fazer cálculos intermediários do bloco. Em (Henderson, *et. al.* 2004) é proposta a adaptação do processo de projeto utilizado em CMOS.

Embora as metodologias descritas a cima sejam ótimas para projetos grandes, esse trabalho não as utilizou por se tratar de um projeto menor. A proposta desse trabalho é implementar um somador que tem alta capacidade para integração em projetos com os mais variados critérios de desempenho. Embora à primeira vista o método para implementação do somador seja bem direta, vários cuidados devem ser feitos em cada etapa. Esse trabalho propõem a documentação e simulação para cada etapa da construção do circuito para provar a funcionalidade e mostrar os cuidados ao se implementar um circuito em QCA. Este trabalho apresentará um somador PPA de 4 bits utilizando uma configuração na árvore de propagação e geração que fornecerá a menor altura possível entre a entrada dos operandos e a saída da soma. A comparação com a literatura é essencial para mostrar a relevância do somador.

3.2 Metodologia

Em um projeto a metodologia serve como um guia para padronizar o trabalho além de agiliza-lo. Nesse projeto não foi diferente, no início do mesmo as primeiras funções booleanas a serem implementadas foram feitas sem um método definido, dando muita margem para erros e gerando dificuldades, como por exemplo, passar da função Booleana diretamente para o leiaute sem os testes de cada etapa, ao verificar a simulação foi constatado um erro no resultado final, não se tinha ideia de onde estaria o problema. Isso fez com que tempo fosse gasto desmontando o circuito até achar o ponto do erro. Devido a esses enganos ao ir construindo circuitos, foi sendo construída uma metodologia, que então foi aplicada ao circuito foco do trabalho, o PPA de 4 bits.

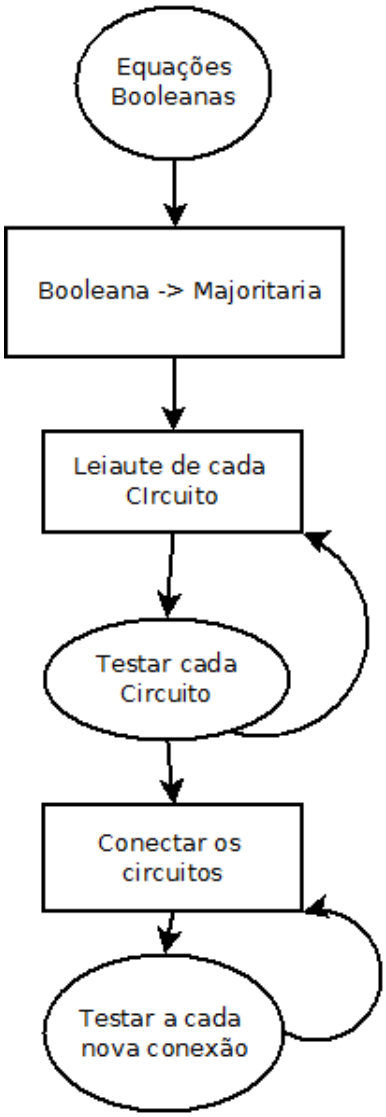

Figura 3.1 – Diagrama do fluxo de projeto utilizado.

Em um projeto utilizando QCA comumente o projetista parte da equação Booleana (que possivelmente é utilizada em um circuito **CMOS**). Esse é o primeiro passo na metodologia, a tradução de equação Booleana para equação majoritária (apresentada no capitulo 2.1.2). Essa tradução pode ser feita de modo trivial utilizando a representação majoritária para A AND B que é M(A,B,0) e para A OR B que é M(A,B,1) que possivelmente dará um circuito QCA não ótimo ou pode-se utilizar algum método de síntese um pouco mais avançada, como o desenvolvido por (KONG, 2010), ou o trabalho desenvolvido por (ZHANG, 2004).

Após termos as equações majoritárias desenharmos o leiaute para cada função utilizando os três dispositivos básicos: fio, inversor e porta majoritária. Após o término de cada leiaute é importante ter um passo de simulação para verificar primeiramente o corretude lógica da função. A verificação de níveis elétricos também é importante. Deve garantir que cada uma das possíveis respostas tenha energia suficiente para polarizar o restante do circuito. Se o circuito é multinível é importante garantir que esta sendo utilizado o menor número de zonas de relógio. Para a escolha do número de zonas de relógio basta verificar quantos níveis tem o circuito, a cada inversor ou porta majoritária em série temos um nível. É importante para garantia da integridade do sinal que o

numero de células em série em uma zona de relógio não seja superior a 10^3 células (NIEMIER, 2001) o circuito tende a cair em um estado excitado ao invés do estado de menor energia como queremos.

A próxima etapa que seria a conexão dos fios pode ou não estar contida na etapa anterior, depende de qual a complexidade do circuito. Por exemplo, para um XOR, cuja a equação majoritária é M(M(!A,B,0),M(A,!B,0),1), é bem simples por tanto pode ser finalizada apenas com alguns testes e a conexão dos fios é bem simples, embora tenha um cruzamento. Caso o circuito seja um pouco mais complexo e tenha vários sub circuitos para serem conectados, a tática adotada para conectar os circuitos é sempre fazer a conexão entre diferentes sub circuitos na quarta camada ou na oitava camada. A cada nova conexão feita é muito importante que seja realizado a simulação para verificar a corretude lógica e os níveis de energia da saída.

3.3 Implementação

Para implementar um PPA, precisamos primeiramente construir cada uma das funções que compõem seu algoritmo. Na tabela 2.3 temos todas as funções necessárias, mas podemos notar que algumas se repetem, por esse motivo precisaremos implementar apenas algumas das funções indicadas na tabela. Na tabela 3.1 podemos ver as funções que serão implementadas, juntamente com suas conversões em equações majoritárias.

Equação Booleana	Equação Majoritária
XOR = A ⊕ B	XOR = M(M(!A,B,0),M(A,!B,0),1)
AND = A • B	AND = M(A,B,0)
F = A + B • C	F = M(M(C,B,0),A,1)

Tabela 3.1 – Tradução das equações booleanas para majoritárias.

Com as equações majoritárias pode-se desenhar o leiaute para cada uma das portas. Na figura 3.2 podemos ver o leiaute da porta lógica AND. Com a porta lógica implementada precisa-se verificar sua corretude lógica e seu nível energético através da simulação mostrada na figura 3.3. Para analisar a corretude precisamos comparar as formas de onda com a tabela verdade da porta lógica AND, mostrada na tabela 3.2. O sinal é computado a cada *nível baixo* do sinal de relógio (Clock 0, na figura). Comparando os dois resultados vemos que logicamente o leiaute implementa uma porta lógica AND. A análise elétrica básica feita é verificar se para nível máximo e nível mínimo todas as saídas chegam sempre o máximo do modulo da amplitude. No exemplo vemos que todas as saídas positivas chegam a 8,52e-001 e todas as saídas negativas chegam a |9,54e-001|.

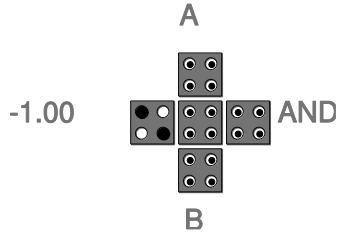

Figura 3.2 – Porta lógica AND implementada em QCA.

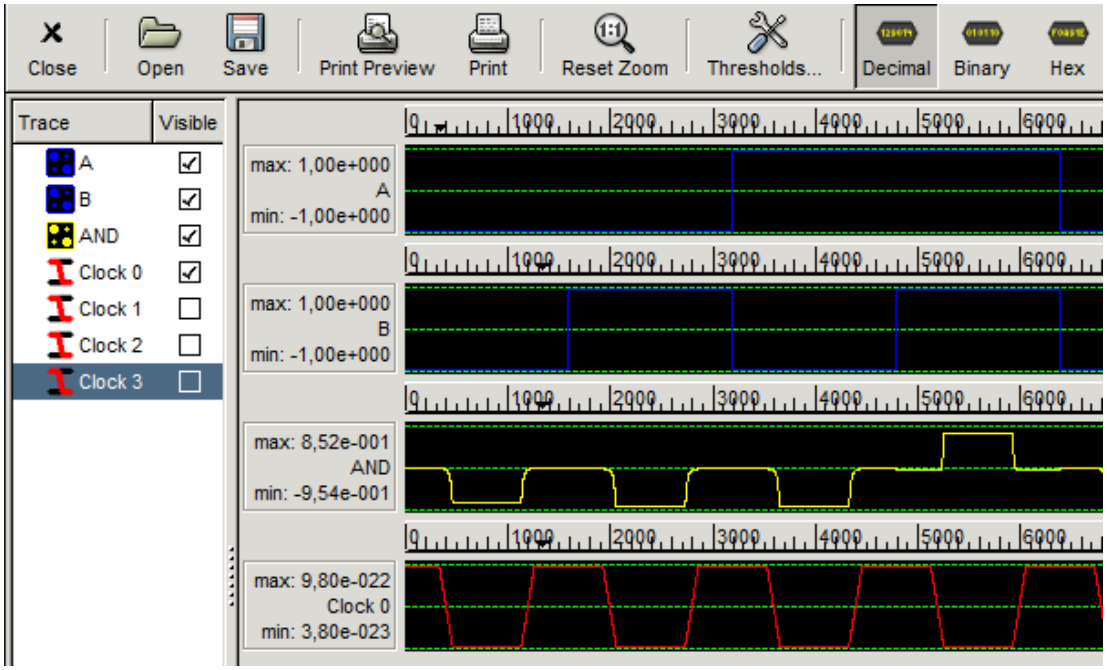

Figura 3.3 – Simulação para a porta lógica AND.

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3.2 – Tabela verdade da porta lógica AND.

Utilizaremos esses mesmos passos para analisar o leiaute das duas portas lógicas restantes. Na figura 2.10 encontra-se o leiaute da XOR. Notamos a presença dos inversores e das portas majoritárias destacadas, respectivamente por círculos e quadrados. Na figura 3.4 vemos as formas de onda para a XOR. Notamos que no caso da XOR a computação ocorre devido a *nível baixo*s da zona de relógio 4 (Clock 4, na figura). A tabela 3.3, mostra a tabela verdade para a XOR. Deve-se tomar cuidado ao analisar as formas de onda da XOR, por esta fazer sua computação apenas na quarta zona de relógio, é preciso esperar até que o dado inicial chega a quarta zona de relógio, no caso da figura 3.4 o primeiro *nível baixo* da zona de relógio 3 é apenas lixo, o resultado de acordo com as entradas A e B começa a ser valido a partir da segunda saída (do segundo *nível baixo* da zona de relógio 3).

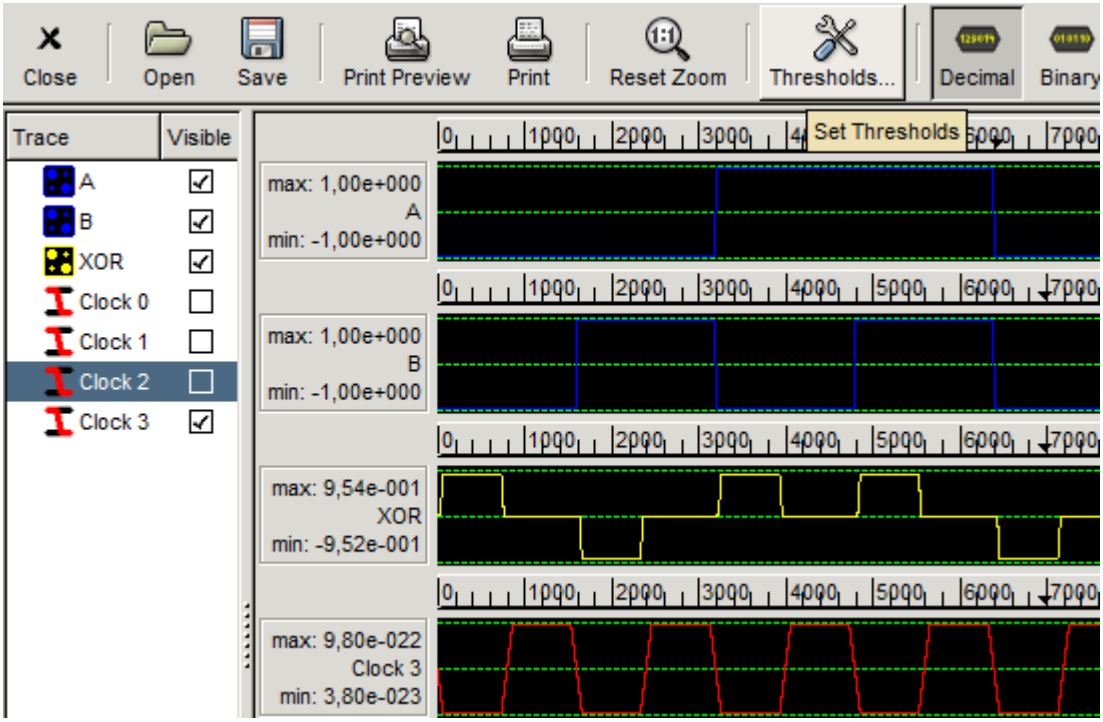

Figura 3.4 - Simulação para a porta lógica XOR.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 3.3 - Tabela verdade da porta lógica XOR.

Na figura 3.5 temos o leiaute da função “F”. O leiaute é bem simples e direto, basta conectar B AND C em uma das entradas da OR com A. Na figura 3.6 encontramos a simulação para a mesma. Nessa simulação precisamos olhar a partir do segundo *nível baixo*, pois novamente a saída não se encontra na zona de relógio 0. A saída para este circuito é analisada a partir do segundo *nível baixo* da zona de relógio 2. Na tabela 3.4 esta a tabela verdade para a função **F**. Ao verificar a corretude e os níveis energéticos se constata que esta tudo certo.

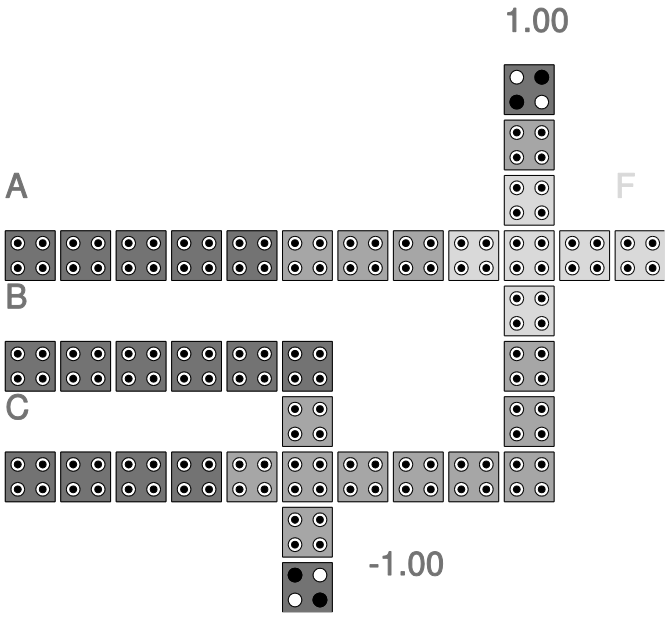

Figura 3.5 - Porta lógica F implementada em QCA.

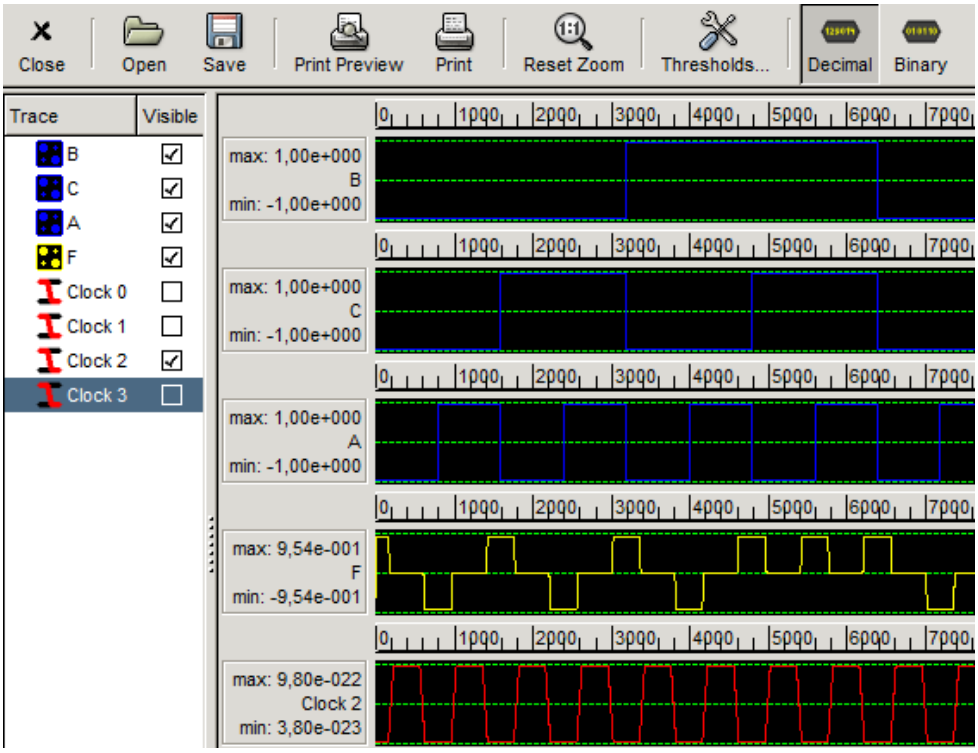

Figura 3.6 - Simulação para a porta lógica F.

B	C	A	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 3.4 - Tabela verdade da porta lógica implementando a função F.

Com todas as partes prontas pode-se realizar a montagem do circuito somador PPA. A figura 3.7 mostra um diagrama de como é o circuito, onde PGx representam as funções de propagação e geral para o bit x, PGx,y representam as funções de propagação e geração de y até x, Cx representa o Cout para bit x e Sx representa a soma para o bit x. A partir desse diagrama podemos montar o circuito PPA de 4 bits, basta juntarmos o leiaute dos circuitos anteriores. A cada novo circuito adicionado a melhor pratica é simular e verificar a corretude do mesmo. Na figura 3.8 temos o leiaute completo para o somador, seu total de células é de 1912 celulas e tem 3,18 um^2. No próximo capitulo será apresentada as formas de onda do mesmo.

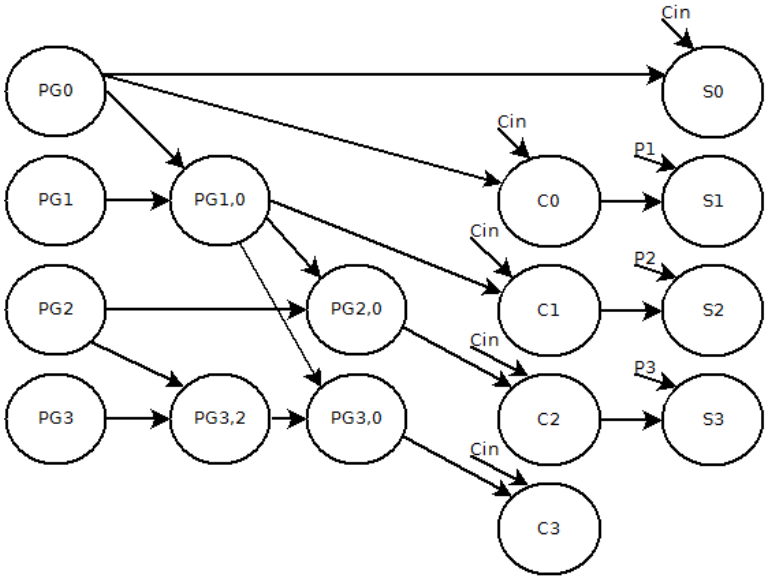

Figura 3.7 – Diagrama do PPA de 4 bits.

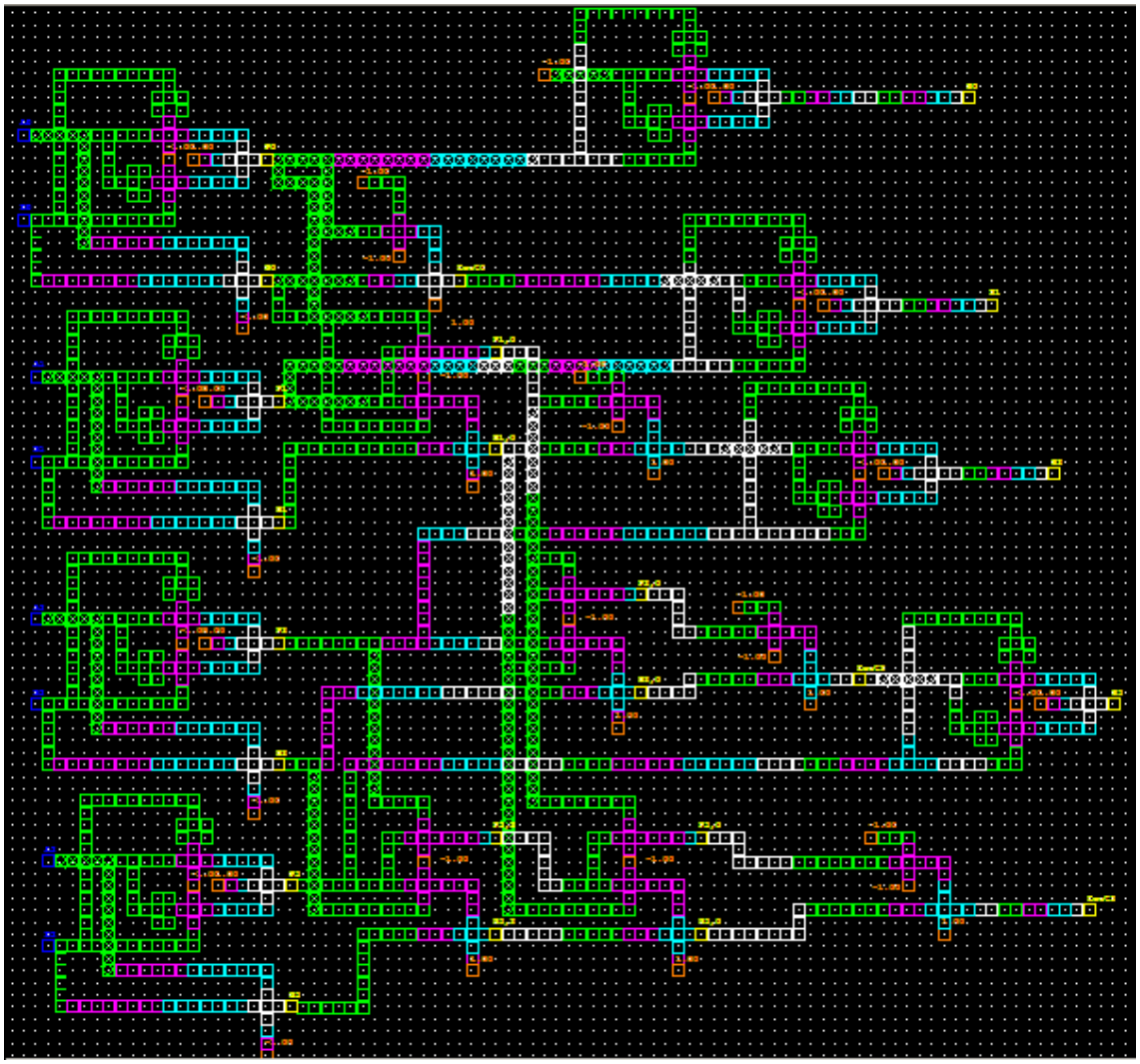

Figura 3.8 – PPA de 4 bits em QCA.

4 RESULTADOS

Todas as simulações feitas neste trabalho utilizaram a aproximação biestável. Na tabela 4.1 estão descritos os parâmetros utilizados para as simulações. As únicas diferenças para os valores padrão da ferramenta são a tolerância de convergência e raio de efeito. A tolerância de convergência foi diminuída em 10^{-2} vezes para melhorar os resultados. O raio de efeito foi modificado para se adequar ao utilizado no artigo (CHO *et. al.*, 2007).

Parâmetro	Valor
Número de Amostras	12800
Tolerância de convergência	0,00001
Raio de efeito	42
Permissividade Relativa	12,900000
Sinal de Relógio Alto	9,800000e-022
Sinal de Relógio Baixo	3,800000e-023
Troca do valor de Relógio	0,000000e+000
Fator de amplitude do sinal de relógio	2,000000
Separação das camadas	11,500000
Numero máximo de iterações por amostra	100

Tabela 4.1 – Parâmetros utilizados na simulação

4.1 Simulação

A figura 4.1 mostra a simulação do PPA de 4 bits para alguns vetores de entrada, pois se fossemos escolher todas as combinações de soma possível não seria pratica a análise. O Cin do somador esta preso em 0. Nessa simulação para facilitar a montagem do leiaute algumas fases foram adicionadas e o atraso para inicio da simulação é de 5 sinais de relógio, mas teoricamente é possível atingir 4.25 ciclos de relógio.

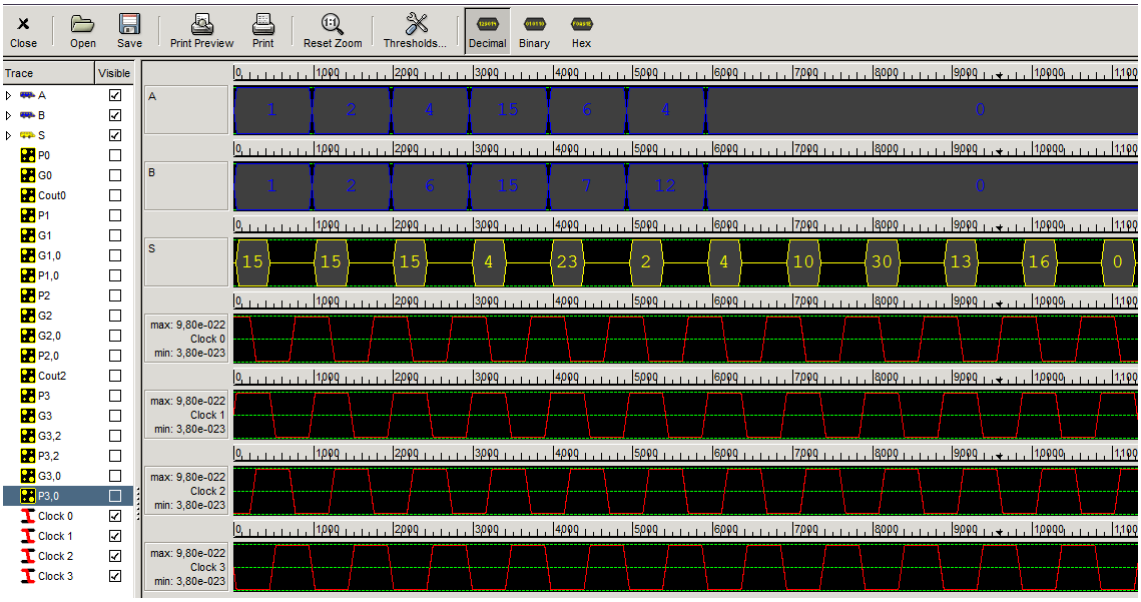

Figura 4.1 – Formas de onda para a simulação do PPA 4 bits.

Os resultados reais começa, a partir do quinto *nível baixo* do sinal de relógio 3 (Clock 3). Podemos verificar que todos os vetores de entrada têm suas somas corretas, inclusive o maior valor possível foi testado e está correto.

4.2 Comparações com a literatura

Primeiramente iremos comparar os resultados para todos os somadores de 4 bits propostos por (CHO e SWARTZLANDER, 2007). Verificando os resultados para o primeiro artigo temos o primeiro dos somadores que é o RCA, este atingiu um atraso de 4,25 ciclos de relógio, uma área de 1,2 um^2 e 651 células, em comparação com os 5 ciclos de relógio, área de 3,18 um^2 e 1912 células. É justo notarmos que o PPA não ganha em nada do RCA para os 4 bits, pois sua lógica de implementação é muito mais complexa, a vantagem do QCA é a adaptabilidade e com isso a escalabilidade do mesmo para com o número de bits a serem somados. Para o CSA proposto no mesmo paper temos os seguintes dados para 4 bits, atraso de 3,75, área de 4,44 um^2 e 1999 células, este se mostra uma boa escolha caso 0,5 ciclos de relógio forem extremamente vitais ao projeto. Agora temos o CLA que é derivado do PPA, este apresenta atraso de 3,5 ciclos de relógio, área de 1,9 um^2 e 1575 células, pode ser uma ótima escolha caso o projetista esteja disposto a ocupar um pouco mais área que o RCA para ganhar 0,75 ciclos de relógio.

Agora para projetarmos o potencial de um somador PPA utilizaremos quatro formulas que representam os melhores e piores casos para área (ou células) e atraso. A fórmula utilizada para o atraso mínimo (melhor caso) tem complexidade $\Omega(\log_2(N))$, onde N é o numero de bits. A função de número de bits com o atraso para o PPA é da forma

$$Atraso(N) = a \cdot log_2(N) + b. \quad (1)$$

A complexidade para o pior caso de atraso é O(N), ou seja, temos um comportamento linear para o pior atraso, da forma

$$Atraso(N) = a \cdot N + b, \quad (2)$$

Onde “a” e “b” são os mesmos da fórmula (1).

Para o pior caso de área, ou seja, melhor caso de atraso, a complexidade é dada por $O(N.\log_2(N))$. A fórmula para o prior caso é dada por

$$Area(N) = c \cdot \frac{N}{2} \cdot log_2(N) + d \cdot N.$$

(3)

Por fim o melhor caso de área segue a mesma formula que o pior caso de atraso, ou seja, (2) mas substituímos "a" e "b" por "c" e "d", respectivamente.

$$Area(N) = c \cdot (N - 1) + d \cdot N.$$

(4)

As constantes das formulas (1) – (4) representam características dos blocos utilizados. Como esses blocos são imutáveis, a principio, podemos utilizar as formulas para aproximar o atraso e área baseados nos blocos já construídos. Os parâmetros "a" , "b", c e "d" representam características estáticas do circuito, como o leiaute para cada bloco, que não mudará. Vamos começar analisando as constantes utilizadas na formula do atraso. A constante "a" é basicamente o atraso de cada um dos circuitos PGx,y de acordo com a figura 3.7. Utilizando a tabela 4.2 podemos calcular "a" = 1. A constante "b" será os circuitos que nunca mudam no caminho, sendo eles PGx, Sx e Coutx, dando então b = 2,75. As constantes "c" e "d" por outro lado podem assumir valor de células ou área. A constante "c" seria basicamente a área ou numero de células de PGy,x, podendo então assumir o valor de 0,09 ou 59. A constante "d" por sua vez assumiria a soma das áreas de Sx, Coutx e PGx, podendo assumir valores de 0,36 ou 276. É importante dizer que as formulas produzem valores aproximados, pois não levam em conta as interconexões entre células.

Circuitos	Área	Células	Atraso
Sx	0,11	89	1
Coutx	0,06	28	0,75
PGx	0,19	159	1
PGy,x	0,09	59	0,75

Tabela 4.2 – Tempo (ciclos de relógio), área(um^2) e Células para cada um dos blocos básicos que compõem o PPA.

Para comparar os resultados de 8 a 64 bits utilizamos as formulas para calcular o atraso, a área e o numero de células aproximada para o PPA, nos piores e melhores casos. É importante lembrar que o pior caso em área corresponde ao melhor caso em atraso e vice-versa. As figuras 4.2, 4.3 e 4.4 mostram gráficos comparando o resultado para o pior e melhor caso em cada uma das dimensões comparadas com os somadores propostos em (CHO e SWARTZLANDER, 2007). As tabelas 4.3, 44 e 4.5 mostram os resultados na forma de mapas de calor, onde o vermelho significa pior caso e verde melhor caso. A primeira vista chegamos a conclusão que em questão de tempo o RCA realmente não escala, enquanto que o PPA (no melhor caso) escala muito bem, embora exista uma tradeoff entre tempo e área. Ainda assim se mostra competitivo quanto a área se comparado com os outros somadores.

bits	PPA min	PPA max	RCA	CLA	CSA
4	4,25	5,75	4,25	3,5	3,75
8	5	8,75	8,25	6,5	7,75
16	5,75	14,75	16,25	10,25	14
32	6,5	26,75	32,25	19	25
64	7,25	50,75	64,25	31,5	45

Tabela 4.3 – Tempos em unidades de ciclos de relógios.

bits	PPA min	PPA max	RCA	CLA	CSA
4	1281	1340	651	1575	1999
8	2621	2916	1499	3988	6216
16	5301	6304	3771	10217	16866
32	10661	13552	10619	25308	45354
64	21381	28992	33531	59030	129611

Tabela 4.4 – Número de células.

bits	PPA min	PPA max	RCA	CLA	CSA
4	1,71	1,8	1,2	1,9	4,44
8	3,51	3,96	3,57	5,53	15,46
16	7,11	8,64	11,78	15,51	48,46
32	14,31	18,72	42,23	42,88	158,38
64	28,71	40,32	159,2	105,2	551,65

Tabela 4.5 – Área da célula em um^2.

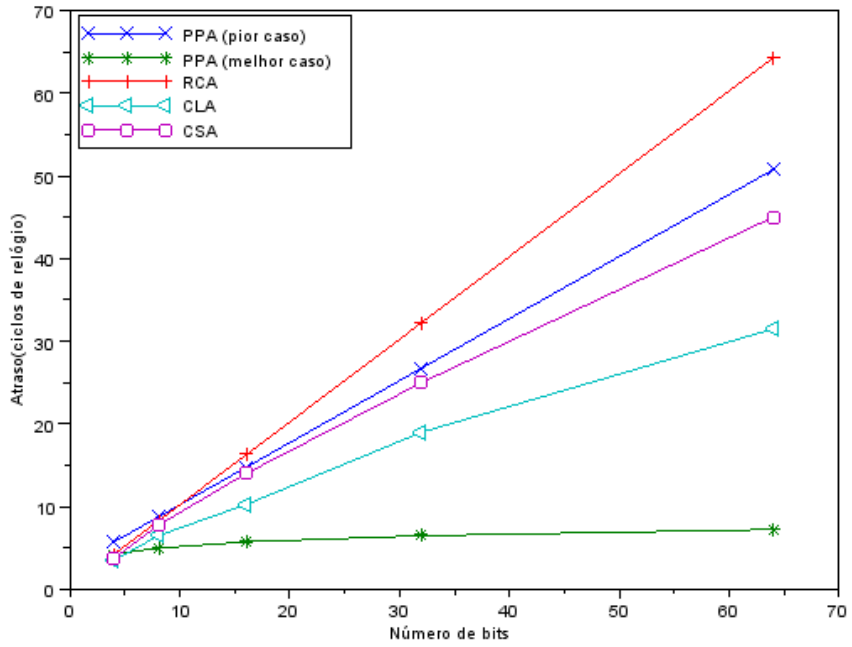

Figura 4.2 – Comparação de tempo entre três circuitos propostos em (CHO e SWARTZLANDER, 2007) e o pior e melhor caso para o PPA.

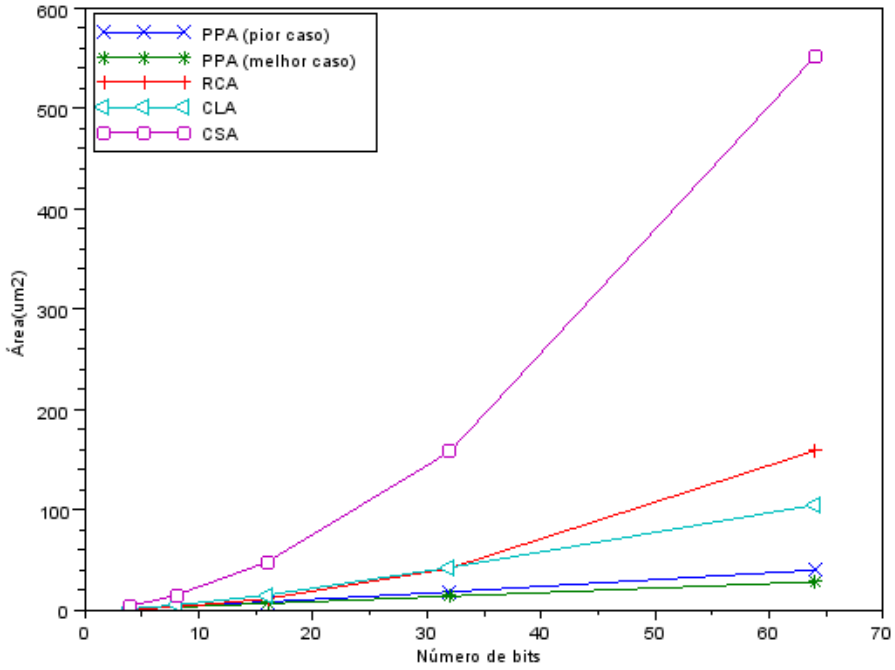

Figura 4.3 - Comparação de área entre três circuitos propostos em (CHO e SWARTZLANDER, 2007) e o pior e melhor caso para o PPA

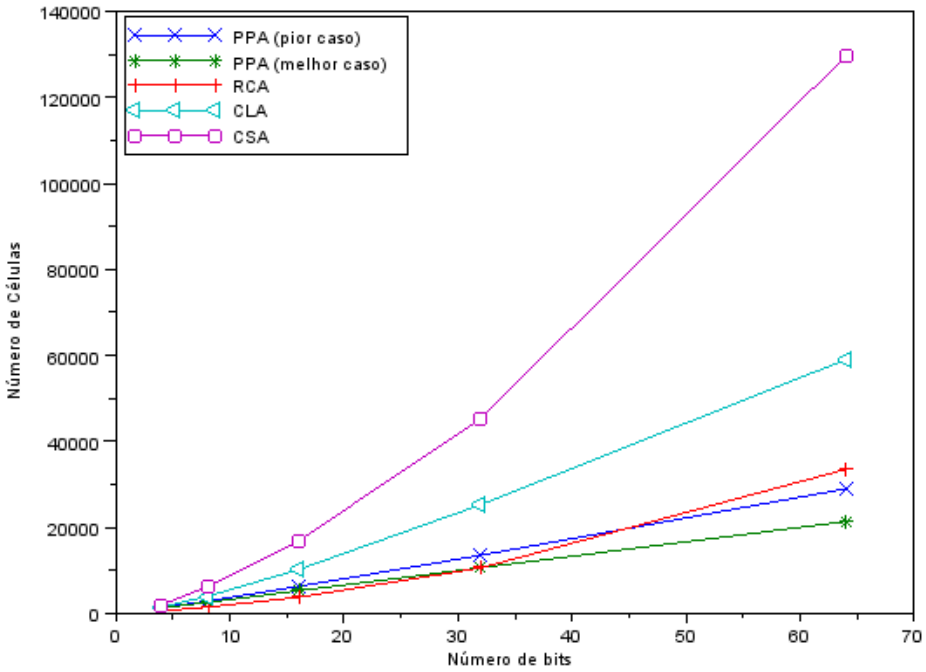

Figura 4.4 - Comparação de número de células entre três circuitos propostos em (CHO e SWARTZLANDER, 2007) e o pior e melhor caso para o PPA

5 CONCLUSÕES

A tecnologia QCA é muito promissora, pois como apresentado neste trabalho é capaz de implementar blocos básicos utilizados em circuitos digitais largamente utilizados em produtos eletrônicos. Por não possuir corrente pode ser considera uma tecnologia com muito pouca dissipação de potencia. O background apresentado nesse trabalho das ferramentas necessárias para o projetista desenvolver qualquer circuito. O PPA desenvolvido com esse trabalho se utiliza apenas do conhecimento apresentado no background.

O PPA é um circuito complexo se comparado com uma XOR, e sua construção utilizando PPA não é tão trivial quanto parece, principalmente por não existir um fluxo de projeto proposto e comprovado para projeto digital utilizando QCA. Neste trabalho um pequeno fluxo foi proposto. Embora simples é bem poderoso para aqueles que nunca fizeram nada utilizando QCA. É proposto também um leiaute para um PPA 4 bits, para aqueles que pretendem estudar somadores utilizando essa tecnologia é importante ter o PPA em seu "portfolio" de portas lógicas, esse trabalho apresenta vários dados para se fazer comparações mais complexas com outros somadores.

Um estudo possível para se melhorar o desempenho do circuito aqui proposto é verificar a possibilidade de implementar portas complexas utilizando apenas uma zona de relógio para se realizar uma função que a principio teria mais de um nível. A figura 5.1 mostra uma das funções propostas no capitulo 3 utilizando o conceito de porta complexa. Embora parece uma ótima ideia um efeito interessante que merece atenção é notado quando a simulação é feita. Esse efeito é circulado na figura 5.2, o mesmo é chamado de "*noise path*" por (KIM *et. al.*, 2007). O "caminho com ruído" (tradução própria) pode causar uma inconsistência no sinal que é propagado por diversos níveis do circuito, no final pode gerar uma resposta inconsistente, simplesmente invertendo um bit no meio do circuito.

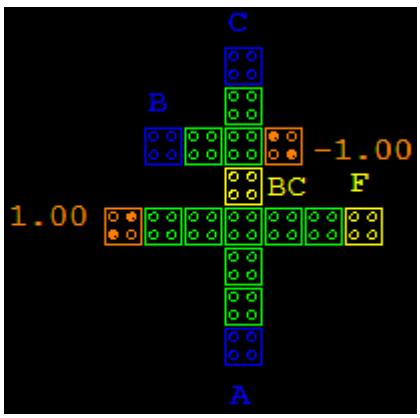

Figura 5.1 – Porta complexa proposta em (TOWNSEND e ABRAHAM, 2004).

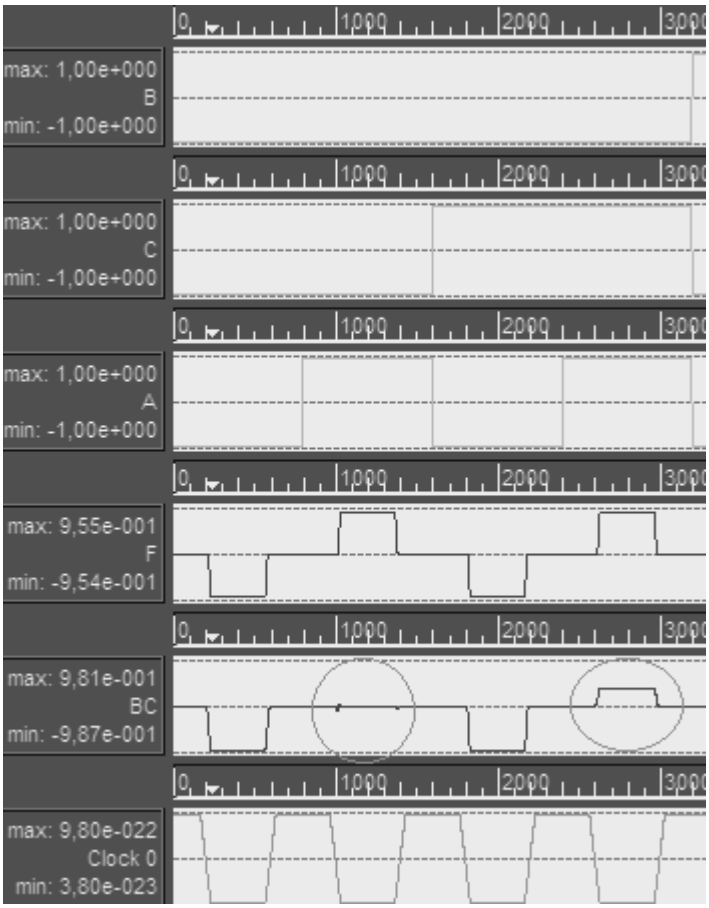

Figura 5.2 – Simulação destacando os “noise paths” de uma porta complexa.

O QCA é uma tecnologia ainda muito nova com muitos desafios pela frente. A criação de um fluxo tão completo quanto o fluxo digital existente para o CMOS é um dos desafios para colocar essa tecnologia em produção de larga escala. É importante que mesmo sem esse fluxo se crie, teste e verifique diferentes arquiteturas para circuitos essenciais para produtos eletrônicos, pois assim que a indústria estiver apta a produzir esta tecnologia ter uma biblioteca de circuitos já testados e homologados vai acelerar em muito o trabalho da indústria na criação de produtos eletrônicos de alto desempenho ou baixo consumo.

REFERÊNCIAS

BRENT, R. P.; KUNG, H. T. A Regular Layout for Parallel Adders. **IEEE Transaction on Computers**, v. c-31, n. 3, mar. 1982.

CHO, H.; SWARTZLANDER JR, E. E. Adder Designs and Analyses for Quantum-Dot Cellular Automata. **IEEE Transaction on Nanotechnology**, v. 6, n. 3, mai. 2007.

EKIMOV, A. I.; ONUSHCHENKO A. A. Quantum size effect in three-dimensional microscopic semiconductor crystals**. JPET Lett**, v. 34, n. 6, set. 1981.

ESCOBAR, K. A.; MANIQUE, L. C.; RIBAS, R. P. Optimal Arrangement of Parallel Prefix Adder. **28th South Symposium on Microlectronics**, Porto Alegre, 2010.

ESCOBAR, K. A.; RIBAS, R. P. Parallel Prefix Adder Design Using Quantum-dot Cellular Automata. **26th Symposium on Integrated Circuits and Systems Design**, Curutiba, set. 2013.

GIN, A.; TOUGAW, P. D.; WILLIAMS, S. An Alternative Geometry for Quantum-Dot Cellular Automata. **Journal of Applied Physics**, v. 85, n. 12, jun 1999.

HAN, T.; CARLSON, D. A. Fast Area Efficient VLSI Adders. **8th Symposium on Computer Arithmetic**, Italy, mai. 1987.

HENDERSON, S. C.; JONHSON, E. W.; JANULIS, J. R.; TOUGAW P. D. Incorporating Standard CMOS Design Process Methodologies into the QCA Logic Design Process. **IEEE Transaction on Nanotechnology**, v. 3, n. 1, mar. 2004.

HENNESSY, K.; LENT, C. S. Clocking of Molecular Quantum-Dot Cellular Automata. **Journal of Vacuum Science and Technology B**, v. 19, n. 5, set/out 2001.

HUBBARD, J. Eletron Correlation in Narrow Energy Band. **Proceedings of the Royal Society of London**, v. 276, n. 1365, 1963.

KIM, K.; WU, K.; KARRI, R. The Robust QCA Adder Designs Using Composable QCA Building Blocks. **IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems**, v. 26, n. 1, jan. 2007.

KOGGE, P. M.; STONE, H. S. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. **IEEE Transaction on Computers**, v. c-22, n.8, ago. 1973.

KONG, K.; SHANG, Y.; LU, R. An Optimized Majority Logic Synthesis Methodology for Quantum-Dot Cellular Automata**. IEEE Transaction on Nanotechnology**, v. 9, n. 2, mar 2010.

KRAUSE, P. G.; MUELLER, R. M.; TOUGAW, P. D.; WEIDNER, J. M. An Alternative Geometry for Quantum Cellular Automata. **VLSI Design**, v. 8, n. (1-4), 1998.

LADNER, R. E.; FISCHER, M. J. Parallel Prefix Computation. **Journal of the Association for Computing Machinery**, v. 27, n. 4, out. 1980.

LENT, C. S.; TOUGAW, D.; PAROD, W.; BERNSTEIN, C. Quantum Cellular Automata. **Nanotechnology**, v. 3, p. 443-450, 1993.

LENG, C.S.; TOUGAW, D. A Device Architecture for Computing with Quantum Dots. **Proceedings of the IEEE**, v. 85, n. 4, abril 1997.

LUSTH, J.; JACKSON, D. J. A Graph Theoretic Approach to Quantum Cellular Design and Analysis. **Journal of Applied Physics**, v. 79, n. 4, fev. 1996.

MOMENZADEH, M.; HUANG, J.; LOMBARDI, F. Defect Characterization and Tolerance of QCA Sequencial Devices and Circuits. **20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems**, 2005.

MOMENZADEH, M.; HUANG, J.; TAHOORI, M. B.; LOMBARDI, F. Characterization, Test and Logic Synthesis of And-Or-Inverter (AOI) Gate Design for QCA Implementation. **IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems**, v. 24, n. 12, dec. 2005.

NIEMIER, M. T.; KOGGE, P. M. Problems in Designing with QCAs: Layout = Timing. **International Journal of Circuit Theory and Application**, v. 29, 2001.

OTTAVI, M.; SCHIANO, L.; LOMBARDI. F.; TOUGAW, D. HDLQ: A HDL Environment for QCA Design. **ACM Journal on Emerging Technologies in Computing Systems**, v. 2, n. 4, oc. 2006.

PASKY J. R.; HENRY, L.; TOUGAW, P. D. Regular Arrays of Quantum-Dot Cellular Automata "macrocells". **Journal of Applied Physics**, v. 87, n. 12, jun 2000.

PERRI, S.; CORSONELLO, P. New Methodology for the Design of Efficient Binary Addition Circuits in QCA. **IEEE Transactions on Nanotechnology**, v. 11, n. 6, nov. 2012.

TORABI, M. A New Architecture for T Flip Flop using Quantum-Dot Cellular Automata**. 3rd Asia Symposium on Quality Electronic Design**, 2011.

TOUGAW, P. D.; LENT, C. S. Logical devices implemented using quantum cellular automata. **Journal on Applied Physics**, v. 75, n. 3, feb 1994.

TOUGAW, P. D.; LENT, C. S. Dynamic behavior of quantum cellular automata. **J. Appl. Phys.**, v. 80, n. 8, out. 1996.

TOWNSEND, W. J.; ABRAHA, J. A. Complex Gate Implementation for Quantum Dot Cellular Automata. **4th IEEE Conference on Nanotechnology**, 2004.

WALUS, K. ATIPS Laboratory QCADesigner Homepage. **ATIPS Laboratory**, Univ. Calgary, Calgary, Canada, 2002. Disponível em: < http://www.mina.ubc.ca/qcadesigner>. Acesso em: out. 2012.

WALUS, K.; DYSART, T. J.; JULLIEN, G. A.; BUDIMAN, R. A. QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata**. IEEE Transaction on Nanotechnology**, v. 3, n. 1, mar. 2004.

ZHANG, R.; WALUS, K.; WANG, W.; JULLIEN, G. A. A Method of Majority Logic Reduction for Quantum Cellular Automata**. IEEE Transaction on Nanotechnology**, v. 3, n. 4, dec. 2004.