Verificando os Limites Energéticos da Computação: A Ferramenta de Análise para Circuitos QCA

Marco Antonio Ribeiro*, Jeferson Figueiredo Chaves*†, Omar Paranaiba Vilela Neto*
*Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais
Email: [marcoantonio, jefchaves, omar]@dcc.ufmg.br
†Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, Minas Gerais

***Resumo*—A tecnologia atual dos transistores CMOS está próxima de atingir seu limite físico de miniaturização, além de apresentar outras desvantagens, tal como o aumento do consumo energético. Diversos nanodispositivos têm sido estudados com o intuito de superar estes limites. Dentre estes, espera-se que dispositivos baseados em interações locais de campo acoplado (do inglês *Field-coupled nanocomputing* ou FCN) atinjam níveis muito baixos de consumo de energia com valores próximos aos limites fundamentais relacionados a perda de informação. Nesse caso, o desenvolvimento de ferramentas para cálculo do limite inferior dessa energia se torna muito importante. Nesse artigo é apresentado uma ferramenta e os seus algoritmos que quantificam essa perda de informação. O *software* lê um arquivo com o *layout* de um circuito combinacional FCN, detecta as portas lógicas através de padrões disponibilizados pelo usuário, cria um grafo representando o circuito e, por fim, calcula o limite inferior para a energia dissipada. Este trabalho apresenta a primeira proposta de automatizar os cálculos mencionados.**

I. Introdução

Eficiência energética é um aspecto importante a ser considerado no desenvolvimento de circuitos digitais, principalmente com o grande aumento no uso de dispositivos embarcados. Apesar de todos os avanços do transistor CMOS, existe uma busca ativa por um substituto. Mesmo não existindo uma tecnologia suficientemente madura, muitos novos dispositivos estão sendo considerados como potenciais alternativas [1].

Dispositivos baseados em interações locais de campo acoplado (FCN) são um dos tópicos de pesquisa promissores na criação de novos componentes de hardware. A transmissão de informação e a computação são realizados nos FCN via interações locais entre blocos de construção em nanoescala organizados em arranjos estruturados. Variações do paradigma FCN estão atualmente sob investigação, incluindo os autômatos celulares de pontos quânticos (*quantum-dot cellular automata* ou QCA), autômatos celulares de pontos quânticos moleculares (*molecular quantum-dot cellular automata* ou MQCA) e lógica nanomagnética (*nanomagnetic logic* ou NML) [2].

Em 1961, Rolf Landauer demonstrou que qualquer processo computacional irreversível, *i.e.*, aquele em que há informação perdida, resulta na perda de $kT\ln(2)$ joules por bit perdido, onde k é a constante de Boltzmann e T é a temperatura em kelvin [3]. Essa perda de informação ocorre, por exemplo, em uma Porta E no seu modo mais convencional de operação. Neste tipo de situação em que existem apenas 2 entradas e 1 saída, ao final de um ciclo de *clock*, isto é, no momento em que as entradas já foram apagadas para que a porta receba nesta conexão seus novos valores, é impossível reverter a operação da porta e obter as entradas originais a partir da saída caso o seu valor seja $false$. Nesse caso, é perdida uma certa quantidade de informação relacionada às entradas desta e, consequentemente, uma certa quantidade de energia é dissipada.

Esse limite termodinâmico em computação, também conhecido como limite de Landauer, tem sua validade discutida desde sua proposta. Apenas recentemente, em 2012, essa proposta foi verificada experimentalmente, confirmando que existe um limite físico na computação irreversível [4]. Mesmo depois deste marco experimental ainda surgem trabalhos que aparentam contradizer o limite de Landauer [5], porém, depois de algum tempo, o impasse acaba sendo esclarecido mantendo-se o princípio fundamental [6].

Ocorre que algumas dessas tecnologias emergentes já operam em níveis energéticos muito baixos, ordem de 1 até 100 kT joules [7], [8]. A tecnologia CMOS, para efeitos comparativos, está limitada a 50 kT joules [9]. Nesse caso, o desenvolvimento de ferramentas para cálculo do limite inferior dessas perdas se torna fundamental.

Nesse artigo é apresentada uma ferramenta e os seus algoritmos que quantificam essa perda de informação. O *software* proposto opera em um circuito combinacional para descobrir sua lógica e executar seus cálculos. Os autômatos celulares de pontos quânticos (QCA) possuem a ferramenta de desenvolvimento mais consolidada quando comparado às outras tecnologias de FCN [10]. Por esse motivo, optou-se por trabalhar inicialmente com os circuitos QCA. Além disso, outra inovação do trabalho é o algoritmo que segue os fluxos de informação, constrói um grafo representando o circuito e detecta sua lógica.

O restante deste artigo está organizado da seguinte forma: Na seção II, é apresentada uma introdução ao QCA. Os trabalhos relacionados são apresentados na seção III. A ferramenta e seus algoritmos são descritos e discutidos na seção IV. Alguns resultados obtidos são mostrados na seção V. Por fim, a seção VI conclui o artigo e mostra algumas extensões relevantes para este trabalho.

II. *Quantum-dot Cellular Automata*

A tecnologia de QCA consiste num agrupamento de células que, quando combinadas e arranjadas em um certo modo, são capazes de executar funções computacionais. A transferência de informação ocorre pelo estado de polarização de várias células, em contraste com computadores tradicionais, que utilizam um fluxo de corrente elétrica para transferir informação [11].

As células de QCA são tipicamente compostas de quatro pontos quânticos localizados nas extremidades das diagonais do dispositivo quadrado. Um ponto, nesse contexto, é uma região onde uma carga elétrica pode estar ou não localizada. Cada célula possui dois elétrons livres e móveis, que são capazes de tunelar entre pontos adjacentes. O tunelamento para fora da célula não é possível graças a uma grande barreira de potencial. As interações Coulombianas entre os elétrons tendem a localizá-los em diagonais opostas, como mostra a Figura 1 (a). Uma célula isolada pode estar em um de dois estados equivalentes de energia. Esses estados são chamados de polarização celular $P = +1$ e $P = -1$. Assim, é possível codificar informações binárias ao considerar que $P = +1$ representa o valor $true$ ou 1 e $P = -1$ representa o valor $false$ ou 0, como também pode ser visto na Figura 1 (a).

Quando duas células são aproximadas, a polarização de uma delas irá influenciar a polarização da outra. Nesse caso, os dois estados de polarização possíveis da segunda célula não serão equivalentes. Por exemplo, considere que uma célula (célula$_1$) tem a polarização fixa em $P1 = +1$ e ela é colocada próxima a uma segunda célula (célula$_2$). A distribuição de cargas da célula$_2$ é influenciada pela distribuição de cargas na célula$_1$. Então, a célula$_2$ tende a ter a mesma polarização que a célula$_1$, reduzindo as interações Coulombianas entre todos os elétrons envolvidos. Seguindo a mesma regra, um fio pode ser construído ao colocar algumas células QCA em uma linha, como mostrado na Figura 1 (b).

Os dispositivos lógicos em QCA são criados pela alocação de células QCA de uma forma em que as interações entre elas possam ser exploradas. As duas portas fundamentais do QCA, o inversor e a Porta Majoritária (*Majority Gate*), são apresentadas e explicadas em detalhes. Quando duas células são colocadas diagonalmente entre si, há uma tendência de reversão das polarizações causada pela repulsão entre elétrons. Essa característica pode ser explorada para construção de um inversor, tal qual é mostrado na Figura 1 (c).

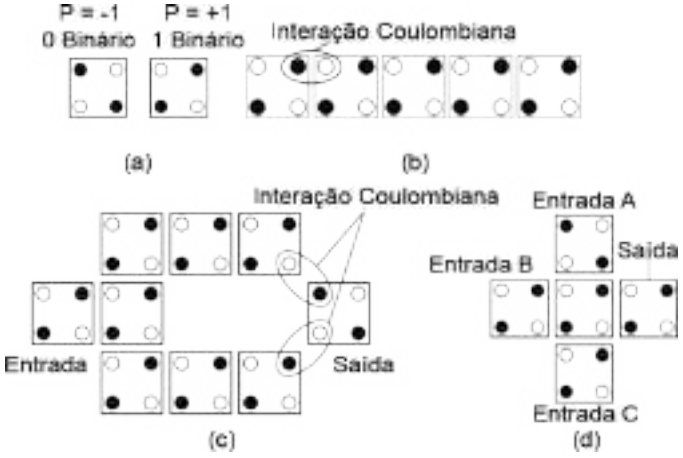

Figura 1: (a) Polarizações possíveis das células de QCA com quatro pontos quânticos. A posição dos elétrons está representada pelos pontos pretos. (b) Fio em QCA. (c) Inversor em QCA. (d) Porta Majoritária em QCA.

A Porta Majoritária (Figura 1 (d)) é a porta lógica básica mais importante no QCA, pois ela pode ser utilizada para construção de portas E e OU, podendo assim ser utilizada para construção de circuitos mais complexos. A célula central da Porta Majoritária tem seu estado de menor energia quando ela assume a polarização da maioria das três células de entrada pois essa configuração faz a repulsão entre os elétrons das três entradas e da célula central ser mínima. Observe na Figura 1 (d) que, mesmo que a célula de entrada A tenha a polarização que representa o nível lógico $false$, a célula de saída tem a mesma polarização que as células B e C, que são a maioria nesse caso. Além disso, se a célula de entrada A estiver fixada em $true$, uma porta OU é criada. Do contrário, ou seja, quando a célula de entrada A estiver fixada em $false$, é criada uma porta E. Com Es, OUs, fios e inversores, qualquer circuito lógico poderá ser implementado [12]. Dessa forma, qualquer circuito computacional pode ser criado usando o paradigma QCA.

Para construir dispositivos mais complexos em QCA será necessária não apenas a seleção cuidadosa da localização de suas células, mas também a sincronização da informação para evitar a chegada e propagação de um sinal em uma porta lógica antes das outras entradas. Essa característica é extremamente importante em circuitos QCA, garantindo seu funcionamento correto. Isso é solucionado pelo relógio (*clock*) do QCA. Tal *clock* pode funcionar através de um campo elétrico, controlando as barreiras de tunelamento dentro de uma célula, coordenando o momento em que uma célula pode ou não estar polarizada.

O *clock* pode ser aplicado a grupos de células (zonas de *clock*). Em cada zona, um potencial pode transformar as barreiras entre os pontos quânticos. Esse esquema de zonas de *clock* permite que um agrupamento de células QCA execute certa computação e congele seu estado para que suas saídas sejam utilizadas como entradas na próxima zona.

A principal ferramenta utilizada atualmente para criação de circuitos em QCA é o QCADesigner [10], mas seu formato de arquivo contém apenas uma descrição de *layout* do circuito, utilizado para simulação física da interação entre as células, sem constar informação de qual lógica é realizada.

III. Trabalhos Relacionados

Hänninen analisou as perdas de informação em somadores baseando-se na estrutura da teoria da informação de Shannon [13]. Este trabalho mostra que a perda de informação depende do nível de abstração em que o sistema é considerado. Isso ocorre porque quando se avalia as portas individualmente, deixa-se de contabilizar a informação que é perdida por uma delas mas preservada por outra porta que está em paralelo.

No caso monolítico, *i.e.*, quando analisadas apenas as relações entre entrada e saída, sem levar em consideração os subsistemas que o compõem, apenas um limite inferior teórico para a perda de energia é dado. Na medida em que o nível de abstração diminui, é possível observar que a perda de energia aumenta. Não apenas a mudança de níveis, mas também o tipo de elemento lógico utilizado como o módulo básico do sistema determina e influencia mais realisticamente o limite inferior para o consumo de energia.

Ercan foi além no desenvolvimento de uma estrutura com o mesmo objetivo [14]. Além dos elementos da teoria da informação de Shannon, seu método contempla um modelo termodinâmico, que provê um melhor entendimento da ligação entre os aspectos lógicos e físicos dos dispositivos. A abordagem modular é baseada em (1) decompor o circuito em zonas menores; (2) obter limites de dissipação para as zonas individuais do circuito; (3) combinar os resultados da análise de zonas individuais em um limite único para todo o circuito.

Srivastava e colaboradores propuseram uma ferramenta de modelagem probabilística para estimar o erro na polarização e um limite superior de perda de energia em circuitos QCA operando sob condições de transições abruptas [15]. Apesar da preocupação com energia, o trabalho deles foca em um caso específico incomum, o de trocas não adiabáticas.

Ainda assim, este trabalho apresenta a primeira tentativa de criar uma ferramenta que calcula automaticamente o limite inferior do gasto energético referente à perda de informação em circuitos FCN. Assim como os outros trabalhos relacionados que tratam de limite inferior de energia, a análise restringe-se a circuitos combinacionais. Outra inovação deste trabalho é o algoritmo de caminhamento, que segue os fluxos de informação, constrói um grafo representando o circuito e detecta sua lógica, sendo fundamental para a evolução da tecnologia pois possibilita verificação lógica deste.

IV. A Ferramenta

A ferramenta desenvolvida neste trabalho calcula o limite inferior de energia dissipada baseando-se na perda de informação. Para tal, ela lê um arquivo com a descrição do *layout* de um circuito QCA criado pelo QCADesigner [10] e uma biblioteca de portas lógicas.

Os padrões da biblioteca são então contrastados com o circuito de modo que a ferramenta detecte quais portas lógicas estão presentes no circuito. As entradas do circuito são preenchidas com seus valores prováveis considerando todas as combinações possíveis em cada entrada ($true$ e $false$ para as entradas variáveis e o valor inicial para as entradas fixas). Os dados são propagados a partir das entradas no sentido do fluxo de informação, o que é chamado aqui de caminhamento. A estrutura principal da implementação é exibida na Figura 2 e descrita nas próximas subseções.

A. Entrada

O QCADesigner utiliza um arquivo semi-estruturado para armazenar cada projeto criado, dividindo o circuito em camadas. São essas,

- **Substrato:** contém a informação do tamanho da grade de células utilizado na discretização do circuito e a dimensão deste;
- **Níveis:** contém as camadas de células por nível do circuito do maior para o menor;
  - **Células:** armazenadas nas camadas de células e contém a polarização, posição, função (entrada, saída, cruzamento, célula comum) da célula.

O outro tipo de entrada é o arquivo *.lunit*, que descreve uma porta lógica e contém o detalhamento de suas funções lógicas e seu padrão de células. Uma porta exemplo e sua estrutura está descrita em Formato 1.

```
1  ...
2
3  Porta Majoritária
4
5  ...
6
7  (0 & 1) | (0 & 2) | (1 & 2)
8
9  ...
10
11 # Padrão
12 e0 i0 e0
13 i0 c1 o2
14 e0 i0 e0
15
16 ...
```

Formato 1: *Arquivo .lunit descrevendo uma Porta Majoritária*

A linha 7 representa a expressão lógica que descreve a porta, onde os valores numéricos representam as entradas por ordem de aparição. Além das entradas, essa expressão suporta os operadores & (E), | (OU), $\wedge$ (XOR), as constantes T ($true$) e F ($false$) e parêntesis encadeados. No caso de portas com mais saídas, ocorreriam várias linhas, uma para cada saída.

As linhas 12 a 14 representam uma matriz que descreve o formato físico da porta lógica, onde cada par $[CaractereNúmero]$ representa uma célula$_p$ (para simplificação, a célula dentro do *layout* do circuito será chamada de célula$_c$ e a célula descrita nesse padrão será chamada de célula$_p$), sendo:

- **Caractere**: representa o tipo da célula$_c$ que deve ser encontrada naquela posição
  - *: qualquer tipo de célula$_c$ ou vazio
  - *e*: nenhuma célula$_c$ (vazio)
  - *c*: qualquer célula$_c$
  - *i*: célula$_c$ de entrada da porta lógica
  - *o*: célula$_c$ de saída da porta lógica
- **Número**: suportado apenas pelos tipos *c*, *i* e *o*, representa o deslocamento de *clock* daquela célula$_p$ com relação a um valor base global. Este valor base é calculado quando a primeira célula$_c$ suportada é encontrada e é igual ao *clock* desta subtraído pelo valor do deslocamento de *clock* da célula$_p$ de mesma posição no *.lunit*. Para que uma célula$_c$ seja aceita, o seu *clock* subtraído

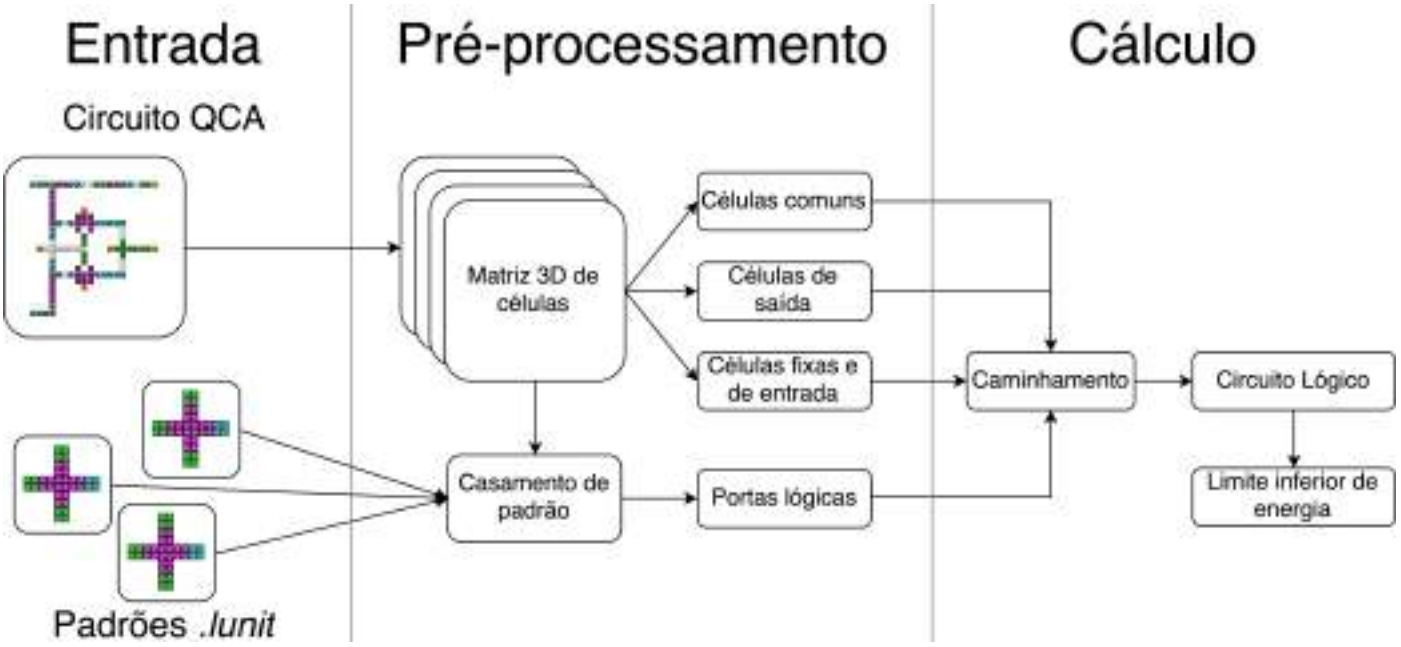

Figura 2: Visão geral da ferramenta.

do deslocamento da célula$_p$ de mesma posição deve ser igual ao valor base.

Cada linha contendo os pares representa uma linha da porta lógica descrita e cada bloco de linhas representa uma camada desta porta, da menor para a maior.

B. Pré-Processamento

O algoritmo de pré-processamento tem início convertendo o circuito QCA para uma matriz tridimensional. As duas primeiras dimensões indicam a posição (x, y) da célula em uma camada, *layer*, representada pela terceira dimensão (z). Inicialmente, o tamanho de cada posição da grade de células é obtido da camada de substrato. Então, para cada camada (*layer*) de nível, as células são extraídas com o valor de sua zona de *clock*, posição (x, y), polarização e tipo. Esses tipos podem variar entre células de entrada, células fixas (tratadas como entrada), células comuns (utilizadas como meio de propagação da informação) e células de saída.

Essas células são inseridas na matriz de acordo com sua posição discretizada, de uma forma em que a diferença da posição de células adjacentes seja de apenas uma unidade. Este é o único passo que deverá ser substituído para atender outros tipos de tecnologias FCN, tal como NML. O analisador que lê o circuito e extrai a informação necessária para criar a matriz tridimensional de células depende da especificação do formato do arquivo de circuito.

Cada porta lógica é buscada na matriz por um processo de casamento de padrão onde cada célula da porta é comparada a cada célula da matriz.

O casamento de padrão é um processo em que, dada as células nas suas respectivas posições (x, y, z) da matriz, testa se existe a possibilidade da porta lógica existir naquela posição, comparando célula a célula, como descrito no Algoritmo 1. Este processo é repetido para cada célula da matriz e para cada porta lógica buscada. A porta é inserida logicamente nas posições onde o casamento ocorre.

Algoritmo 1: Função de casamento de padrão

```
1  Função Casamento de Padrão (matriz, padrão, cx, cy, cz)
       parâmetro: matriz = matriz de células
       parâmetro: padrão = padrão da porta lógica buscada
       parâmetro: cx, cy, cz = posição da célula inicial
       retorno  : true se porta existe naquela posição ou
                  false caso contrário
2      para z ← 0 até padrão.níveis faça
3          mz ← cz + z
4          para y ← 0 até padrão.linhas faça
5              my ← cy + y
6              para x ← 0 até padrão.colunas faça
7                  mx ← cx + x
8                  se matriz[mx, my, mz] ≠ padrão[x, y, z] então
9                      retorna false
10     retorna true
```

Durante esse processo, duas portas poderão ocupar uma mesma célula da matriz (superposição). Nesses casos, uma das seguintes decisões é tomada:

- Se a interseção ocorrer em todas as células da menor porta lógica (porta$_1$), então esta porta está contida dentro da maior (porta$_2$). A porta$_2$ é inserida na matriz e a

porta$_1$ é desabilitada. Um exemplo pode ser visto na Figura 3, onde a porta em amarelo foi encontrada dentro da porta em azul.
- Nos outros casos, ambas são inseridas na matriz.

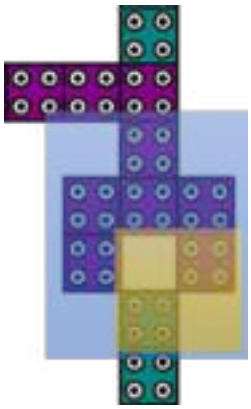

Figura 3: Superposição de padrões.

Cada porta lógica tem suas células comuns desabilitadas para evitar entrada externa de dados. Nos casos de um circuito com portas que não estão descritas em nenhum dos arquivos *.lunit* fornecidos, o caminhamento tentará tratá-las como conjuntos de fios. Nesses casos, o primeiro fluxo que atingir uma determinada célula será considerado aquele que efetivamente passará por ela, podendo, assim, levar a resultados indesejados. O mesmo acontecerá para circuitos não combinacionais.

Agora que todas as células e portas lógicas foram inseridas na matriz, a informação é propagada das entradas do circuito (variáveis e fixas) para criação de um grafo representando a lógica do circuito, processo que é chamado de caminhamento.

C. Cálculo

O caminhamento é formado de fluxos que representam os caminhos tomados pela informação, que, por sua vez, são compostos de configurações, cada uma representando o estado inicial das entradas do circuito e o bit de informação sendo propagado. Assim, cada configuração contém uma lista de N condições iniciais e um valor booleano. Cada posição i dessas condições referencia o estado inicial da i-ésima entrada variável (as entradas fixas são desconsideradas por não possuírem mais de um estado). Cada estado é um conjunto de valores possíveis. Como as entradas são binárias, os estados são:

Iniciou como $false$ = { 0 }
Iniciou como $true$ = { 1 }
Pode ser $true$ ou $false$ = { 0 , 1 }

O terceiro estado ocorre quando uma entrada não foi encontrada e, consequentemente, não foi unida com esse fluxo. O conjunto vazio é um estado de erro, indicando que aquele fluxo contém duas configurações conflitantes (junção de duas configurações onde o valor inicial de certa entrada é diferente) e é utilizado para filtragem.

Quando n fluxos se encontram dentro de uma porta lógica, ocorre uma junção de todas as configurações, estado a estado, de uma forma que uma configuração onde uma entrada i iniciou como $false$ nunca irá se juntar a uma configuração em que essa mesma entrada i iniciou como $true$. Essa operação tem um custo exponencial por tratar de todas as combinações de entrada possíveis, situação necessária para o cálculo que será apresentado. A tabela I indica o resultado de cada junção, que é uma operação comutativa.

Tabela I: Resultado da junção de dois estados na mesma posição em configurações

Estado$_1$	Estado$_2$	Resultado
{ 0 }	{ 0 }	{ 0 }
{ 1 }	{ 0 }	$ERRO$
{ 0 , 1 }	{ 0 }	{ 0 }
{ 0 , 1 }	{ 1 }	{ 1 }
{ 0 , 1 }	{ 0 , 1 }	{ 0 , 1 }

Essa operação é aplicada exaustivamente combinando todas as configurações de fluxo$_1$ para todas as de fluxo$_2$. O resultado é então aplicado a todas as de fluxo$_3$ e assim sucessivamente até que todos os fluxos de entrada tenham sido consumidos. No fim, todas as configurações que não obtiveram erro tem seu novo valor booleano calculado, onde tem-se dois casos:

1) **Caso inicial:** o valor booleano desta configuração será igual ao valor booleano propagado pela entrada. Assim, cada entrada variável terá um fluxo com 2 configurações, uma $true$ e uma $false$ e cada entrada fixa terá um fluxo com apenas uma configuração com seu valor fixo.
2) **Junção:** os valores booleanos das configurações utilizadas para criação da nova configuração são extraídos e colocados em um vetor na mesma ordem de chegada dos fluxos originários de cada. Como a configuração final será composta pela junção de uma configuração de cada um dos fluxos de entrada (sem repetição dos mesmos), este vetor indica as entradas da porta lógica onde ocorrerá a junção para aquela configuração. Então, a expressão lógica da porta é executada levando em consideração que suas entradas $0, 1, ..., n-1$ são respectivamente as mesmas posições deste vetor e, o resultado dessa expressão lógica, é o valor booleano da configuração final.

O fluxo resultante é a lista de todas as configurações que não geraram erro durante o processo de junção. Este será propagado a partir da porta lógica.

A Figura 4 mostra um exemplo de circuito. As entradas variáveis estão marcadas em azul (Entrada$_1$ e Entrada$_2$), as entradas fixas (Entrada$_3$ e Entrada$_4$) são laranja e a única saída (S$_3$) é amarela. As portas lógicas (Porta Majoritária$_1$ e Porta Majoritária$_2$) estão nas áreas destacadas em azul. Os fluxos iniciados antes do caminhamento estão expostos na Tabela II. Nesta, os fluxos de Entrada$_1$ e Entrada$_2$ são compostos de 2 configurações e os fluxos de Entrada$_3$ e Entrada$_4$ são compostos de 1 configuração.

Tabela II: Fluxos iniciais do circuito na Figura 4

Origem do fluxo	Booleano	Entrada_1	Entrada_2
Entrada_1	$false$	$\{ 0 \}$	$\{ 0 , 1 \}$
	$true$	$\{ 1 \}$	$\{ 0 , 1 \}$
Entrada_2	$false$	$\{ 0 , 1 \}$	$\{ 0 \}$
	$true$	$\{ 0 , 1 \}$	$\{ 1 \}$
Entrada_3	$false$	$\{ 0 , 1 \}$	$\{ 0 , 1 \}$
Entrada_4	$true$	$\{ 0 , 1 \}$	$\{ 0 , 1 \}$

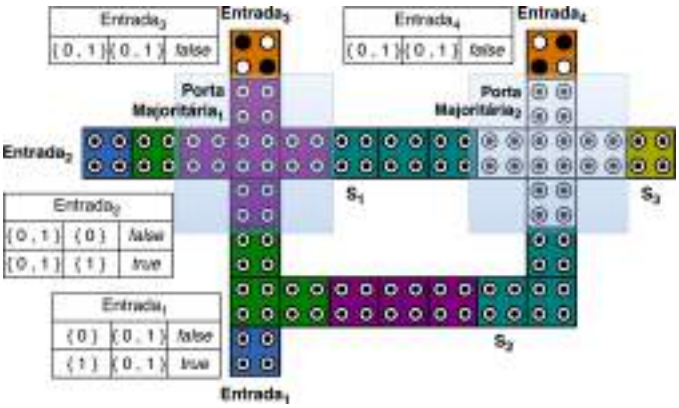

Figura 4: Fluxo de dados no primeiro passo do algoritmo de caminhamento.

Na Figura 4, os fluxos da Entrada_1, Entrada_2 e Entrada_3 propagam-se através das células até que eles encontram-se na Porta Majoritária$_1$. O fluxo resultante está exposto na Tabela III. A Porta Majoritária$_1$ é marcada como usada e seu fluxo propaga para a Porta Majoritária$_2$ como mostrado na Figura 5.

Tabela III: Fluxo após junção na Porta Majoritária$_1$

Origem do fluxo	Booleano	Entrada_1	Entrada_2
Porta Majoritária$_1$	$false$	$\{ 0 \}$	$\{ 0 \}$
	$false$	$\{ 0 \}$	$\{ 1 \}$
	$false$	$\{ 1 \}$	$\{ 0 \}$
	$true$	$\{ 1 \}$	$\{ 1 \}$

Na Figura 5, os fluxos da saída da Porta Majoritária$_1$, Entrada_1 e Entrada_4 chegam na Porta Majoritária$_2$. Algumas configurações do fluxo da Porta Majoritária$_1$ e da Entrada_1 não poderão ser unidas nesta, visto que elas possuem diferentes valores para o estado inicial da Entrada_1. O fluxo resultante é exposto na Tabela IV. A Porta Majoritária$_2$ é marcada como usada e seu fluxo propaga para a saída, como mostrado na Figura 6.

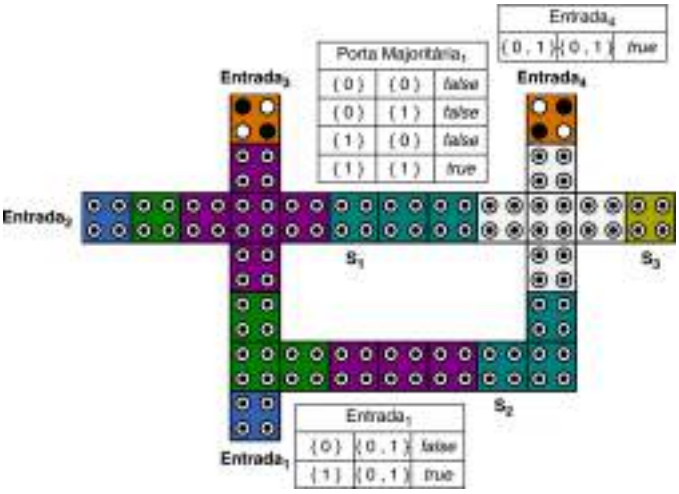

Figura 5: Fluxo de dados no segundo passo do algoritmo de caminhamento.

Tabela IV: Fluxos após a junção na Porta Majoritária$_2$

Origem do fluxo	Booleano	Entrada_1	Entrada_2
Porta Majoritária$_2$	$false$	$\{ 0 \}$	$\{ 0 \}$
	$false$	$\{ 0 \}$	$\{ 1 \}$
	$true$	$\{ 1 \}$	$\{ 0 \}$
	$true$	$\{ 1 \}$	$\{ 1 \}$

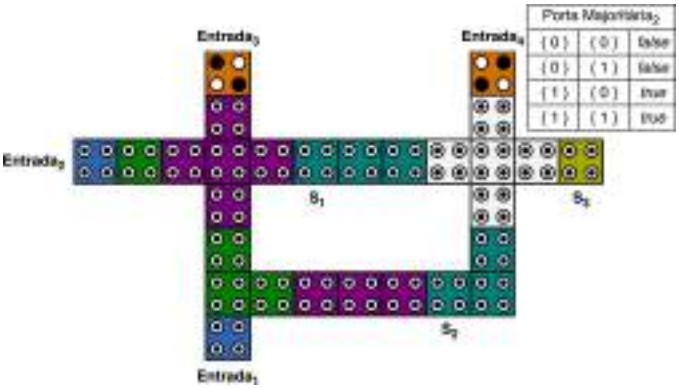

Figura 6: Fluxo de dados no terceiro passo do algoritmo de caminhamento.

O pseudocódigo do caminhamento é mostrado no Algoritmo 2. O algoritmo anda exaustivamente pelo circuito (linhas 1 a 18) até que nenhum movimento possa ser feito, o que é indicado pela variável $mudou$. Para cada $célula$ encontrada (linhas 4 a 17) são salvas as vizinhas para as quais ela vai propagar a informação em um conjunto chamado $próximas$ (linha 5).

As linhas 6 a 10 iteram sob todas as portas lógicas que possuem a $célula$ atual como entrada, marcando sua posição

Algoritmo 2: Pseudocódigo do caminhamento

```
   dado      : clocks = lista de entradas ordenada por clock
   resultado: grafo representando circuito
1  repita
2      mudou ← false
3      para cada clock em clocks faça
4          para cada célula em clock faça
5              próximas ← célula.transfere()
6              para cada porta em célula.comoEntrada() faça
7                  porta.marcaEntrada(célula)
8                  se porta.encontrouTudo() então
9                      entrada.juntaFluxos()
10                     próximas = próximas ∪ porta.saídas
11             para cada porta em célula.comoSaída() faça
12                 se não porta.encontrouTudo() então
13                     porta.desabilita()
14             para cada próxima em próximas faça
15                 clocks.insere(próxima)
16                 próxima.anterior ← célula
17                 mudou ← true
18 até não mudou
```

como encontrada (linha 7). Se tal porta teve todas as entradas encontradas (linha 8), ocorre o processo descrito de junção dos fluxos (linha 9) e suas saídas são inseridas no conjunto de $próximas$.

Nas linhas 11 a 13, se tal $célula$ é saída de uma porta ainda não completa (linha 12), essa porta é desabilitada pois ela não poderá mais propagar para a $célula$ (linha 13).

As linhas 14 a 17 inserem as células do conjunto de $próximas$ na lista a ser explorada (linha 15) e no grafo resultante com a $célula$ atual como antecessora (linha 16). A variável indicando mudança é ativada para que a lista continue sendo explorada (linha 17).

Após o processo do caminhamento, um grafo conectando as células de entrada e portas lógicas até as saídas, representando o circuito, é obtido. O cálculo do limite inferior de energia poderá, então, ser executado de duas formas, sendo por porta lógica ou por zona de *clock*.

1) Por porta lógica: Para cada porta lógica p encontrada durante o caminhamento é computado o conjunto de todas as entradas e saídas binárias. Para cada entrada e, sua probabilidade de aparição $P(e)$ considerando-se todas as outras é dada na Equação 1a. Através desta, é possível calcular sua auto-informação $I(e)$ como na Equação 1b. Por fim, a entropia da informação $H(e)$ é dada na Equação 1c.

$$P(e) = \frac{aparições(e)}{\sum_{n=0}^{n=total_e-1} aparições(n)} \tag{1a}$$

$$I(e) = -\log_2(P(e)) \tag{1b}$$

$$H(e) = P(e)I(e) \tag{1c}$$

O mesmo é feito para cada saída s, como descrito nas equações 2a, 2b e 2c.

$$P(s) = \frac{aparições(s)}{\sum_{n=0}^{n=total_s-1} aparições(n)} \tag{2a}$$

$$I(s) = -\log_2(P(s)) \tag{2b}$$

$$H(s) = P(s)I(s) \tag{2c}$$

Então, é possível calcular o resultado do limite inferior de energia para tal porta lógica pela Equação 3.

$$E(p) = \left(\left(\sum_{n=0}^{n=total_e-1} H(e)\right) - \left(\sum_{n=0}^{n=total_s-1} H(s)\right)\right) kT \tag{3}$$

O resultado para o circuito será a soma do limite inferior de energia por todas as portas lógicas, formalizado na Equação 4.

$$E_{total} = \sum_{p \in portas\ lógicas} E(p) \tag{4}$$

Essa análise aplicada no circuito da Figura 4 tem a Porta Majoritária$_1$ e a Porta Majoritária$_2$ como portas lógicas, assim, o cálculo será a soma da energia destas.

A Porta Majoritária$_1$ tem Entrada$_1$, Entrada$_2$ e Entrada$_3$ como entradas e S_1 como saída. O cálculo para esta porta lógica está descrito na Tabela V.

Tabela V: Cálculo do limite inferior de energia da Porta Majoritária$_1$

Entrada					
Entrada$_1$	Entrada$_2$	Entrada$_3$	P(e)	I(e)	H(e)
$false$	$false$	$false$	0.25	2	0.5
$false$	$true$	$false$	0.25	2	0.5
$true$	$false$	$false$	0.25	2	0.5
$true$	$true$	$false$	0.25	2	0.5
				Resultado	2.0

Saída			
S_1	P(e)	I(e)	H(e)
$false$	0.75	0.415037499	0.311278124
$false$			
$false$			
$true$	0.25	2	0.5
		Resultado	0.811278124

Resultado Porta Majoritária$_1$
$2.0 - 0.811278124 = 1.18872188\ kT$ joule

A Porta Majoritária$_2$ possui Entrada$_1$, Entrada$_4$ e S_1 como entradas. O cálculo para esta porta lógica está descrito na Tabela VI.

Tabela VI: Cálculo do limite inferior de energia da Porta Majoritária$_2$

Entrada					
Entrada$_1$	Entrada$_4$	S$_1$	P(e)	I(e)	H(e)
$false$	$true$	$false$	0.5	1	0.5
$false$	$true$	$false$			
$true$	$true$	$false$	0.25	2	0.5
$true$	$true$	$true$	0.25	2	0.5
				Resultado	1.5

Saída			
S$_3$	P(e)	I(e)	H(e)
$false$	0.5	1	0.5
$false$			
$true$	0.5	1	0.5
$true$			
		Resultado	1.0

Resultado Porta Majoritária$_2$
$1.5 - 1.0 = 0.5\ kT$ joule

O limite inferior de energia deste circuito é dado pela soma do resultado para as suas portas lógicas, o que é mostrado na Equação 5.

$$E_{total} = 1.18872188 + 0.5 = 1.68872188\ kT \text{ joule} \quad (5)$$

2) Por zona de clock: Cada zona de *clock* será considerada uma porta lógica onde as suas primeiras células serão consideradas as entradas e as suas últimas serão as saídas. O cálculo do limite inferior de energia de cada uma será, então, análogo ao por porta lógica.

O resultado do circuito será a soma dos limites inferiores de energia de cada zona de *clock*, formalizado na Equação 6.

$$E_{total} = \sum_{z \in \ zonas\ de\ clock} E(z) \quad (6)$$

Essa análise aplicada no circuito da Figura 4 tem as zonas de *clock* como descritas na Figura 7 onde o bloco azul representa a Zona 1, o bloco laranja representa a Zona 2, o bloco verde representa a Zona 3 e o bloco amarelo representa a Zona 4.

A Zona 1 possui a Entrada$_1$ e a Entrada$_2$ como entradas e (A), (B) e (C) como saídas. As saídas (B) e (C) são fios com o valor da Entrada$_1$ e a saída (A) é um fio com o valor da Entrada$_2$. O cálculo para esta zona de *clock* está descrito na Tabela VII.

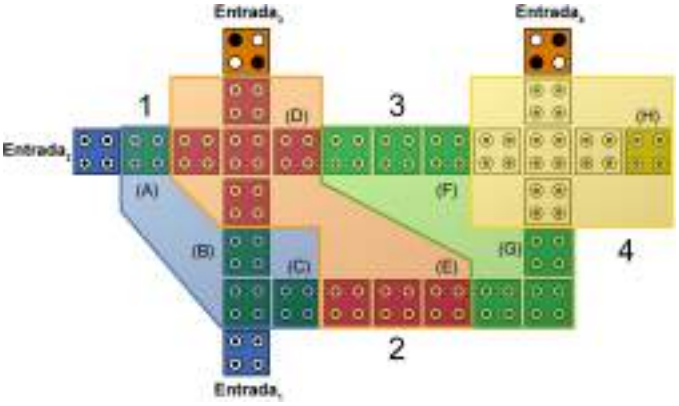

Figura 7: Zonas da Figura 4.

Tabela VII: Cálculo do limite inferior de energia da Zona 1

Entrada				
Entrada$_1$	Entrada$_2$	P(e)	I(e)	H(e)
$false$	$false$	0.25	2	0.5
$false$	$true$	0.25	2	0.5
$true$	$false$	0.25	2	0.5
$true$	$true$	0.25	2	0.5
			Resultado	2.0

Saída					
(A)	(B)	(C)	P(e)	I(e)	H(e)
$false$	$false$	$false$	0.25	2	0.5
$true$	$false$	$false$	0.25	2	0.5
$false$	$true$	$true$	0.25	2	0.5
$true$	$true$	$true$	0.25	2	0.5
				Resultado	2.0

Resultado Zona 1
$2.0 - 2.0 = 0\ kT$ joule

A Zona 2 possui (A), (B), (C) e Entrada$_3$ como entradas e (D) e (E) como saídas. A saída (D) é o resultado das entradas (A), (B) e Entrada$_3$ aplicadas na Porta Majoritária$_1$ e a saída (E) é um fio com o valor de (C). O cálculo para esta zona de *clock* está descrito na Tabela VIII.

Tabela VIII: Cálculo do limite inferior de energia da Zona 2

Entrada						
(A)	(B)	(C)	Entrada$_3$	P(e)	I(e)	H(e)
$false$	$false$	$false$	$false$	0.25	2	0.5
$true$	$false$	$false$	$false$	0.25	2	0.5
$false$	$true$	$true$	$false$	0.25	2	0.5
$true$	$true$	$true$	$false$	0.25	2	0.5
					Resultado	2.0

Saída				
(D)	(E)	P(e)	I(e)	H(e)
$false$	$false$	0.5	1	0.5
$false$	$false$			
$false$	$true$	0.25	2	0.5
$true$	$true$	0.25	2	0.5
			Resultado	1.5

Resultado Zona 2
$2.0 - 1.5 = 0.5\ kT$ joule

A Zona 3 possui (D) e (E) como entradas e (F) e (G) como saídas. As saídas (F) e (G) são, respectivamente, fios com o valor das entradas (D) e (E). O cálculo para esta zona de *clock* está descrito na Tabela IX.

Tabela IX: Cálculo do limite inferior de energia da Zona 3

Entrada				
(D)	(E)	P(e)	I(e)	H(e)
$false$	$false$	0.5	1	0.5
$false$	$false$			
$false$	$true$	0.25	2	0.5
$true$	$true$	0.25	2	0.5
			Resultado	1.5

Saída				
(F)	(G)	P(e)	I(e)	H(e)
$false$	$false$	0.5	1	0.5
$false$	$false$			
$false$	$true$	0.25	2	0.5
$true$	$true$	0.25	2	0.5
			Resultado	1.5

Resultado Zona 3
$1.5 - 1.5 = 0\ kT$ joule

A Zona 4 possui (F), (G) e Entrada$_4$ como entradas e (H) como saída. (H) é o resultado das entradas (F), (G) e Entrada$_4$ aplicadas na Porta Majoritária$_2$. O cálculo para esta zona de *clock* está descrito na Tabela X.

Tabela X: Cálculo do limite inferior de energia da Zona 4

Entrada					
(F)	(G)	Entrada$_4$	P(e)	I(e)	H(e)
$false$	$false$	$true$	0.5	1	0.5
$false$	$false$	$true$			
$false$	$true$	$true$	0.25	2	0.5
$true$	$true$	$true$	0.25	2	0.5
				Resultado	1.5

Saída			
(H)	P(e)	I(e)	H(e)
$false$	0.5	1	0.5
$true$	0.5	1	0.5
		Resultado	1

Resultado Zona 4
$1.5 - 1 = 0.5\ kT$ joule

O limite inferior de energia deste circuito é dado pela soma do resultado para suas zonas, o que é mostrado na Equação 7.

$$E_{total} = 0 + 0.5 + 0 + 0.5 = 1\ kT \text{ joule} \tag{7}$$

V. Resultados

A ferramenta foi executada em diversos circuitos disponíveis na literatura. Contudo, por questões de limitação de espaço, aqui serão apresentados dois estudo de casos.

A. Porta XOR

Figura 8: Porta XOR.

A porta XOR (Figura 8) utilizada é composta de três portas majoritárias, sendo que a Porta Majoritária$_1$ e a Porta Majoritária$_2$ têm uma das entradas invertidas. Sua equação lógica é descrita na Equação 8.

$$S = E_1 \oplus E_2 \tag{8}$$

- Resultado por porta lógica: $2.87744\ kT$ joules
- Resultado por zona de *clock*: $1\ kT$ joule

Tabela XI: Energia dissipada por porta lógica da Figura 8

Porta Lógica	Limite Inferior
Porta Majoritária$_1$	1.18872
Porta Majoritária$_2$	1.18872
Porta Majoritária$_3$	0.5
Resultado	2.87744

B. Porta de Feynman

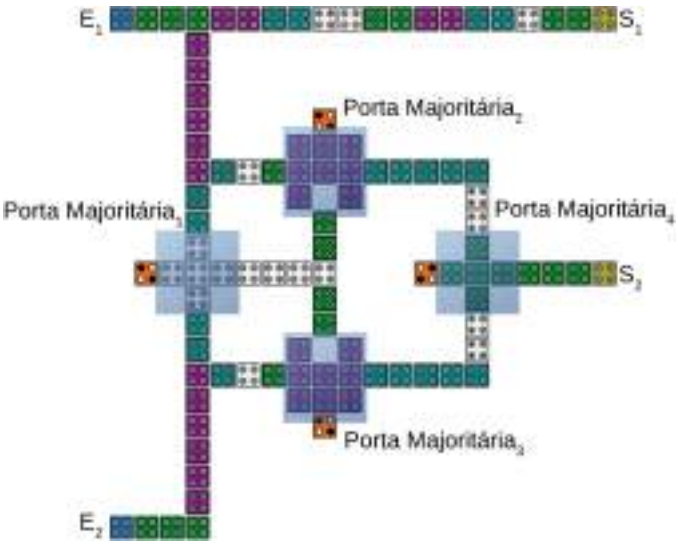

Figura 9: Porta de Feynman.

A Porta de Feynman (Figura 9) utilizada é composta de quatro portas majoritárias, sendo que a Porta Majoritária$_2$ e a Porta Majoritária$_3$ têm uma das entradas invertidas. Suas equações lógicas estão descritas nas Equações 9a e 9b.

$$S_1 = E_1 \tag{9a}$$
$$S_2 = E_1 \oplus E_2 \tag{9b}$$

- Resultado por porta lógica: $3.06617\ kT$ joules
- Resultado por zona de *clock*: $0\ kT$ joule

Tabela XII: Energia dissipada por porta lógica da Figura 9

Porta Lógica	Limite Inferior
Porta Majoritária$_1$	1.18872
Porta Majoritária$_2$	0.688722
Porta Majoritária$_3$	0.688722
Porta Majoritária$_4$	0.5
Resultado	3.06616

A porta de Feynman é um caso particularmente interessante por implementar uma função bijetora, *i.e.*, uma operação logicamente reversível. Essa é exatamente a razão pela qual os resultados dos dois casos têm valores tão diferentes. A perda por zonas de *clock* é igual a zero porque a informação perdida em alguma porta pode estar sendo preservada em outra parte paralela do mesmo circuito como ocorre, por exemplo, na zona que contém a Porta Majoritária$_2$ e Porta Majoritária$_3$ e o pedaço de fio que liga a entrada E_1 a saída S_1.

VI. Conclusão

Neste trabalho foi proposta e implementada uma ferramenta que obtém o limite inferior de energia dissipada para circuitos com dispositivos FCN. Apesar dessa versão funcionar apenas para circuitos QCA, uma simples alteração é necessária para aceitar outros dispositivos FCN.

Para trabalhos futuros, pretende-se aumentar o suporte para outros dispositivos FCN e incluir esses cálculos de energia no *software* QCADesigner para guiar a criação de circuitos QCA energicamente eficientes, além construir uma ferramenta para circuitos NML incorporando os mesmos cálculos de energia.

O algoritmo de caminhamento também poderá ser utilizado para verificação lógica do circuito, visto que este é capaz de detectar a lógica através da organização das células, sendo uma ferramenta muito importante para a evolução da tecnologia.

Referências

[1] R. K. Cavin, P. Lugli, and V. V. Zhirnov, "Science and engineering beyond moore's law," *Proceedings of the IEEE*, vol. 100, no. 13, pp. 1720–1749, 2012.

[2] N. G. Anderson and S. Bhanja, *Field-Coupled Nanocomputing*. Springer, 2014.

[3] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, 1961.

[4] A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz, "Experimental verification of landauer/'s principle linking information and thermodynamics," *Nature*, vol. 483, no. 7388, pp. 187–189, 2012.

[5] M. Lopez-Suarez, I. Neri, and L. Gammaitoni, "Sub-kbt micro-electromechanical irreversible logic gate," *Nature Communications*, vol. 7, p. 12068, 2016.

[6] L. B. Kish, "Comments on "sub-kbt micro-electromechanical irreversible logic gate"," *Fluctuation and Noise Letters*, vol. 15, no. 04, p. 1620001, 2016.

[7] J. Hong, B. Lambson, S. Dhuey, and J. Bokor, "Experimental test of landauer's principle in single-bit operations on nanomagnetic memory bits," *Science Advances*, vol. 2, no. 3, p. e1501492, 2016.

[8] J. Timler and C. S. Lent, "Maxwell's demon and quantum-dot cellular automata," *Journal of Applied Physics*, vol. 94, no. 2, pp. 1050–1060, 2003.

[9] S. Agarwal, J. Cook, E. DeBenedictis, M. P. Frank, G. Cauwenberghs, S. Srikanth, B. Deng, E. R. Hein, P. G. Rabbat, and T. M. Conte, "Energy efficiency limits of logic and memory," in *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[10] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman, "Qcadesigner: a rapid design and simulation tool for quantum-dot cellular automata," *Nanotechnology, IEEE Transactions on*, vol. 3, no. 1, pp. 26 – 31, march 2004.

[11] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, pp. 49–57, 1993.

[12] W. Wernick, "Complete sets of logical functions," *Transactions of the American Mathematical Society*, vol. 51, no. 1, pp. 117–132, 1942.

[13] I. K. Hänninen, C. S. Lent, and G. L. Snider, "Quantifying irreversible information loss in digital circuits," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 11, no. 2, p. 10, 2014.

[14] I. Ercan and N. G. Anderson, "Heat dissipation in nanocomputing: lower bounds from physical information theory," *Nanotechnology, IEEE Transactions on*, vol. 12, no. 6, pp. 1047–1060, 2013.

[15] S. Srivastava, A. Asthana, S. Bhanja, and S. Sarkar, "Qcapro-an error-power estimation tool for qca circuit design," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 2377–2380.