Reescrita e verificação de circuitos digitais representados sob a forma de Grafos de ANDs e Inversores

Marcos Henrique Backes, André Inácio Reis
UFRGS, Porto Alegre, Brasil
{mhbackes, andreis}@inf.ufrgs.br

OBJETIVO O objetivo desse trabalho é apresentar um método de reescrita de AIGs baseado nas funções XOR e XNOR e investigar a eficiência de usar BDD e SAT para checar a equivalência dos AIGs produzidos.

Reescrita de AIGs através das funções XOR/XNOR

GRAFO DE ANDS E INVERSORES

Um Grafo de ANDs e Inversores (AIG) é um grafo dirigido e acíclico composto de nodos AND de duas entradas (AND2), nodos de entrada (PI) e de saída (PO), conectados por arestas diretas e negadas [1]. Um possível AIG para a função $O = \overline{(xy + (x + y)z)}$ é mostrado na Figura 1. AIGs não são estruturas canônicas, ou seja, a mesma função pode ser representada por vários AIGs diferentes. A Figura 2 mostra um exemplo disso.

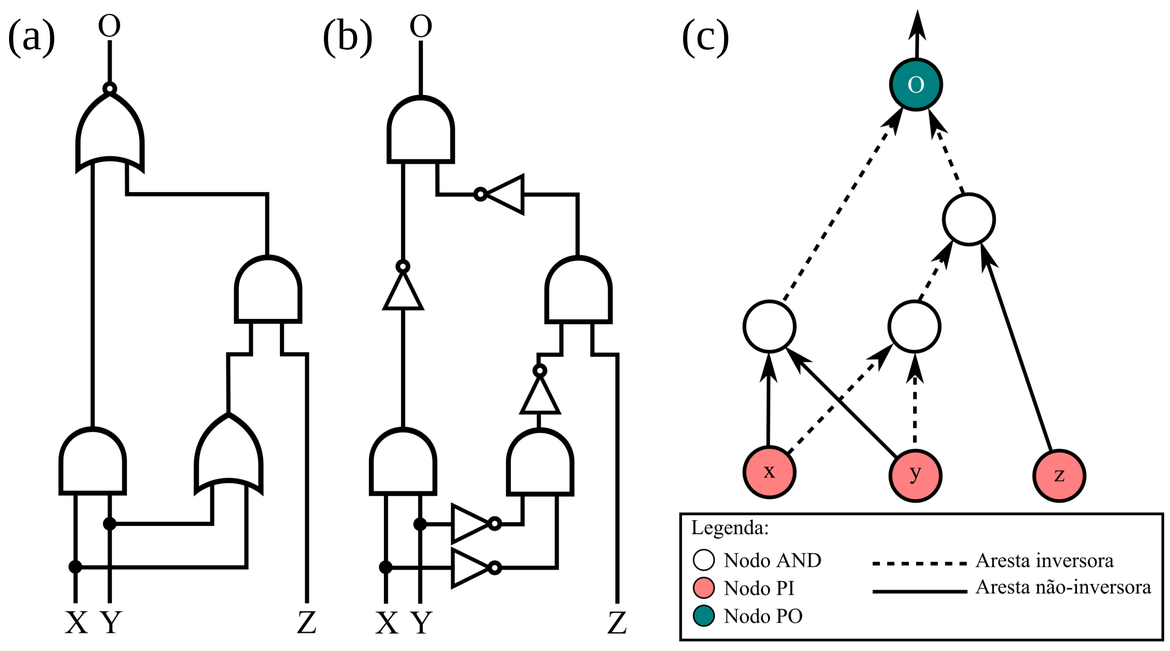

Figura 1: Um circuito combinacional (a), sua representação com portas AND2 e inversores (b), e uma representação de AIG (c).

AIG E FUNÇÕES XOR/XNOR

Observando o número de nodos, as representações de AIG para as funções XOR e XNOR têm pelo menos 3 nodos AND2. A Figura 2 ilustra esses AIGs. Note que todos os nodos destacados como *XOR/XNOR tipo 2* têm suas entradas com a mesma polaridade. Esse trabalho verifica se o *tipo 2* pode gerar melhores implementações usando portas lógicas simples.

METODOLOGIA DE REESCRITA DE XOR/XNOR

Para substituir *XOR/XNOR tipo* 1 por *XOR/XNOR tipo 2*, primeiro, o padrão *tipo 1* é procurado no AIG. Então, para cada padrão encontrado, a polaridade das arestas é reorganizada para transformá-lo em no *tipo 2*. A Figura 3 mostra um exemplo da metodologia proposta.

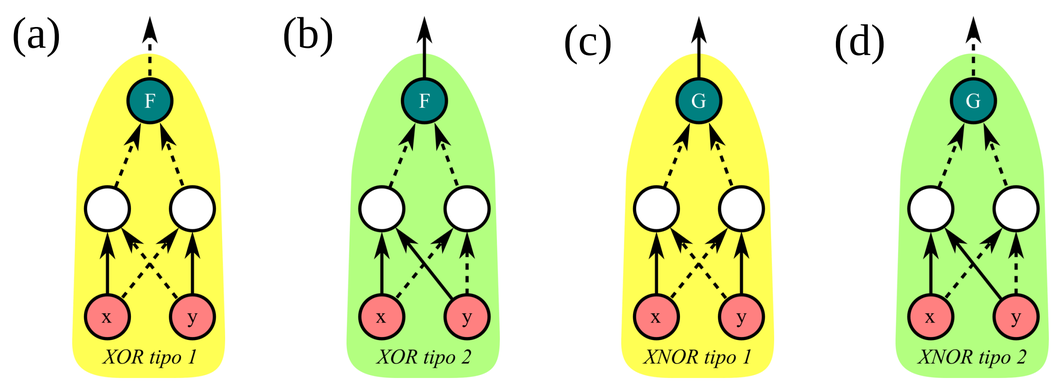

Figura 2: Exemplo de não-canonicidade do AIG: representação de $F = x \oplus y$ (a) e (b); representação de $G = \overline{(x \oplus y)}$ (c) e (d).

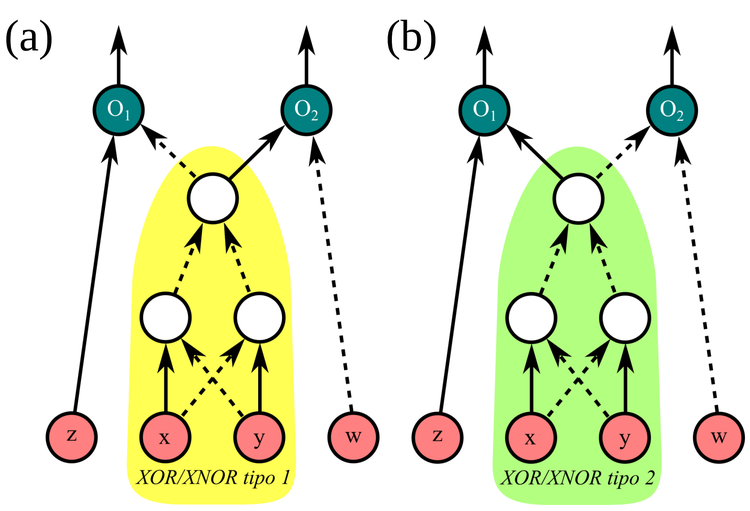

Figura 3: Exemplo de reescrita: AIG contendo *XOR/XNOR tipo 1* (a); o resultado depois de aplicar a metodologia proposta (b).

Verificação de Equivalência usando BDD e SAT

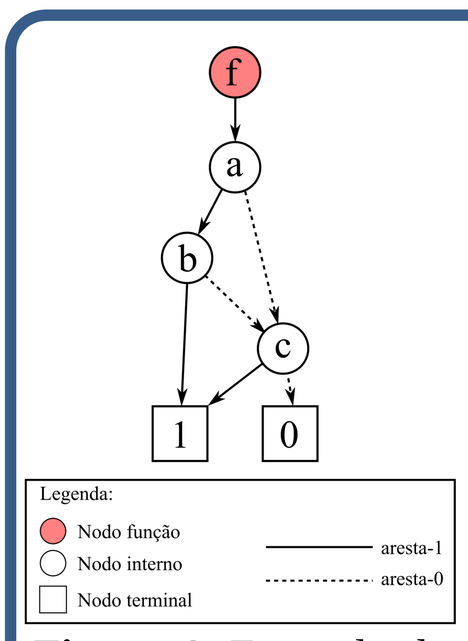

Figura 4: Exemplo de BDD representando a função $f = ab + c$.

DIAGRAMA DE DECISÃO BINÁRIA

Um Diagrama de Decisão Binária (BDD) é um grafo representando um conjunto de decisões com valor binário, culminando em uma decisão global que pode ser *1* ou *0* [3]. Uma vez que o BDD Reduzido e Ordenado (ROBDD) é uma estrutura canônica, ele pode ser usado para checar a equivalência de funções booleanas. A figura 4 ilustra o ROBDD representando a função $f = ab + c$.

COMPARANDO AIGs USANDO BDDs

Para checar a equivalência de dois AIGs X e Y, é criado um BDD para cada saída de X e Y. Dessa forma, é explorada a canonicidade dos BDDs. Se os BDDs gerados são os mesmos para cada saída de X e sua saída correspondente em Y, então, os AIGs são equivalentes.

PROBLEMA DE SATISFATIBILIDADE BOOLEANA

O Problema de Satisfatibilidade Booleana (SAT) [2] é um problema no qual, dado uma função booleana $f(x_1, \ldots, x_n)$, tenta-se determinar se existe uma combinação de assinalamento dos valores de entrada de f que avalia a função para *1*. Se existe tal combinação, então, a função é chamada de *SAT*, senão, é chamado de *UNSAT*.

COMPARANDO AIGs USANDO SAT

Para checar a equivalência de dois AIGs X e Y, é criado um novo AIG $Z = X \oplus Y$. Uma formula CNF é derivada de Z usando a Transformação Tseitin [4] e essa fórmula é usada como entrada de um solucionador SAT [2]. X e Y são equivalentes se e somente se Z é *UNSAT*.

RESULTADOS

A Tabela 1 apresenta os principais resultados obtidos. A coluna rotulada "#NODOS" apresenta, o número de nodos AND do AIG. "#TIPO 1", "#TIPO 2" representam, respectivamente, o número de transistores de suas implementações utilizando portas lógicas simples antes e depois de aplicar o método proposto. "%" representa a diferença percentual entre as duas implementações apresentadas. "TEMPO" e "MEMÓRIA" apresentam o tempo de execução e uso de memória para comparar ambos os AIGs inicial e final usando BDD e SAT.

Tabela 1: Análise do número de transistores do método proposto e verificação de equivalência.

CIRCUITO	#NODOS	#TIPO 1	#TIPO 2	%	TEMPO		MEMÓRIA	
					SAT	BDD	SAT	BDD
C432	127	608	608	0,00%	58 ms	29.6 min	94.4 MB	3.7 MB
C499	386	1912	1840	-3,77%	367 ms	> 27 h	99.4 MB	2.3 MB
C880	306	1448	1412	-2,49%	84 ms	1 h	94.4 MB	65.7 MB
C1355	390	1928	1848	-4,15%	777 ms	> 27 h	98.3 MB	2.2 MB
C1908	354	1704	1672	-1,88%	490 ms	25.4 h	98.4 MB	44.1 MB
C2670	534	2546	2466	-3,14%	351 ms	> 27 h	101.3 MB	85.6 MB
C3540	918	4084	4084	0,00%	4 s	> 27 h	110.3 MB	62.3 MB
C5315	1323	6078	6078	0,00%	1.3 s	> 27 h	113.4 MB	8.4 MB
C6288	1870	8776	8662	-1,30%	> 27 h	> 27 h	1.1 GB	2.6 GB
C7552	1377	6580	6520	-0,91%	701 ms	> 27 h	132.4 MB	42.3 MB
i10	1799	8284	8274	-0,12%	1.5 s	> 27 h	128.4 MB	102 MB

REFERÊNCIAS
[1] A. Mishchenko, S. Chatterjee, and R. Brayton. DAG-aware AIG rewriting: a fresh look at combinational logic synthesis. In Proc. of Design Automation Conf. (DAC), 2006.
[2] N. Een and N. S¨ orensson. MiniSat: A SAT solver with conflict-clause minimization. Sat, 5, 2005.
[3] S. B. Akers. Binary Decision Diagrams. IEEE Trans. on Computer, 27(6):509–516, 1978.
[4] G. S. Tseitin. On the complexity of derivation in propositional calculus. Studies in constructive mathematics and mathematical logic, 2, 1968.

AGRADECIMENTOS
Pesquisa parcialmente financiada por CNPq, FAPERGS sob a concessão 11/2053-9 (Pronem), e o European Community's Seventh Framework Programme sob a consessão 248538 (Synaptic).