EQUIVALÊNCIA LÓGICA DE FUNÇÕES BOOLEANAS ATRAVÉS DE REPRESENTAÇÃO POR NÚMEROS INTEIROS

MICAEL MARTINS POPPING; RENATO SOUZA DE SOUZA; FELIPE DE SOUZA MARQUES; LEOMAR SOARES DA ROSA JUNIOR

Universidade Federal de Pelotas - {mmpopping; rsdsouza; felipem; leomarjr}@inf.ufpel.edu.br

1. INTRODUÇÃO

O fluxo de projeto de um circuito integrado é definido por uma série de etapas, que vai desde a especificação mais alto nível do circuito até o desenho final do projeto (GEREZ, 1999). Dentre estas etapas, temos, a especificação do projeto, a descrição em linguagem de hardware, a síntese de alto nível, a síntese lógica, a síntese física e a parte de verificação. Cada etapa é constituída por centenas ou milhares de algoritmos responsáveis por otimizar o processo e as funções de custo dos sistemas, como área, performance e dissipação de potência.

A etapa de síntese lógica é uma etapa fundamental para o projeto de um circuito integrado. Esta etapa costuma ser dividida em síntese independente de tecnologia e mapeamento tecnológico (HASSOUN, 2012). Na parte de síntese independente de tecnologia, otimizações são feitas na estrutura, com o objetivo de otimizar as funções custos finais do circuito. Um passo importante durante estas otimizações, é a verificação da equivalência lógica. Este passo, garante que qualquer otimização realizada no processo, não tenha alterado o funcionamento lógico do circuito.

Portanto, este trabalho apresenta um desenvolvimento de um método para verificação de equivalência lógica. O método é baseado na utilização de números inteiros (REIS, 2009), extraídos da tabela verdade, com o objetivo de tornar fácil e rápido aplicar as operações lógicas sobre uma determinada função.

2. METODOLOGIA

O método constitui-se em gerar a árvore binária de operadores da expressão Booleana e com auxílio de inteiros associados aos nós, calcular e verificar a equivalência de expressões booleanas.

Dada uma função Booleana como entrada, é montada a árvore binária de operadores. Percorre-se a expressão procurando os operadores e para todo operador encontrado cria-se um nó. Com isso, atribui-se ao seu filho direito o operando direito e o novo nó pode ser inserido na árvore. Esse processo de inserção deve ser feito respeitando a precedência de operadores. Portanto, valem as seguintes regras:

1. No caso da árvore está vazia o novo nó torna-se a raiz e o operando esquerdo é atribuído ao seu filho esquerdo, como ilustra a Figura 1.

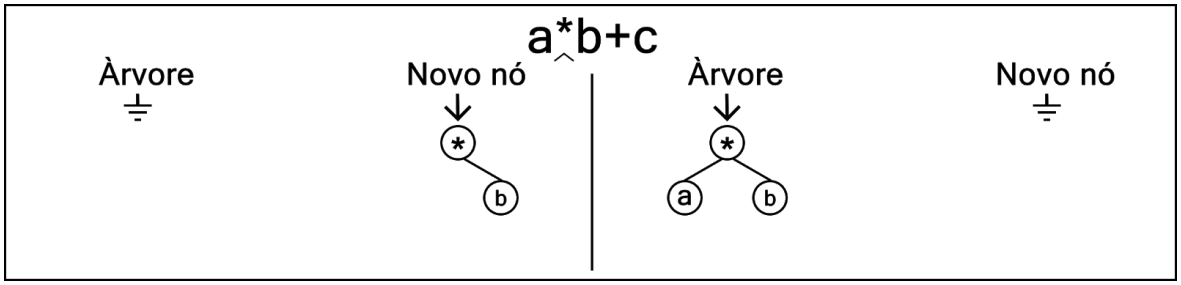

Figura 1. Caso 1 de inserção na árvore

2. No caso da raiz ser uma AND (*) e o novo nó ser uma OR (+) o filho esquerdo do novo nó aponta para a raiz tornando o novo nó a raiz da árvore, como mostra a Figura 2.

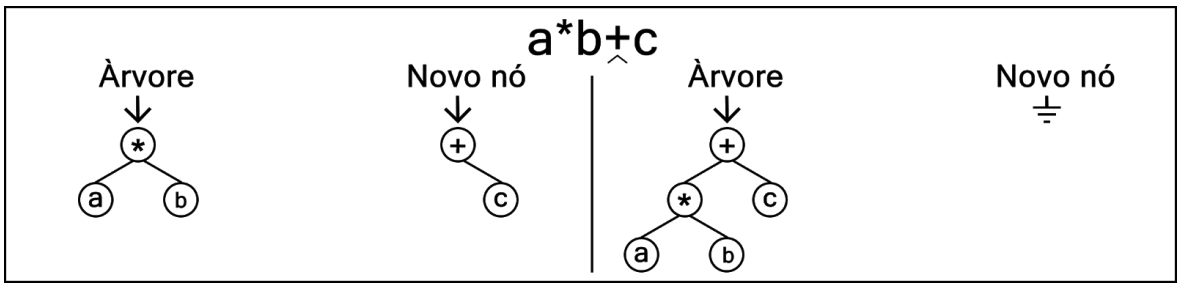

Figura 2. Caso 2 de inserção na árvore

3. No caso da raiz ser uma OR e o novo nó ser uma AND o filho esquerdo do novo nó aponta para o filho direito da raiz e o filho direito da raiz aponta para o novo nó, como demonstra a Figura 3.

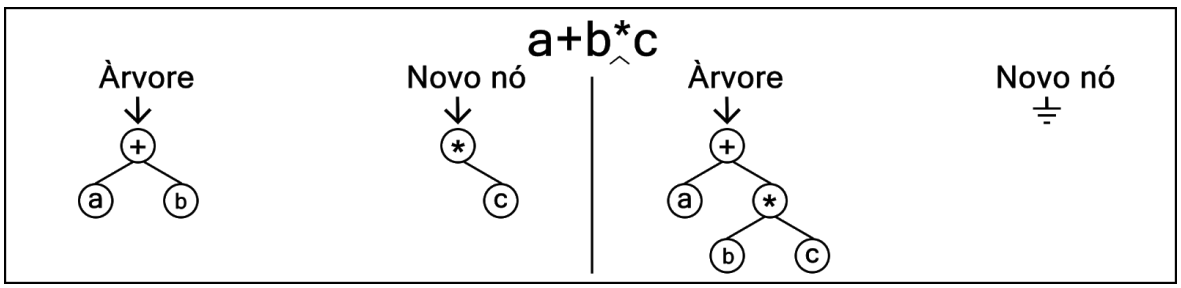

Figura 3. Caso 3 de inserção na árvore.

4. Quando a raiz e o novo nó possuem o mesmo operador o processo é similar a regra 3, como mostra a Figura 4.

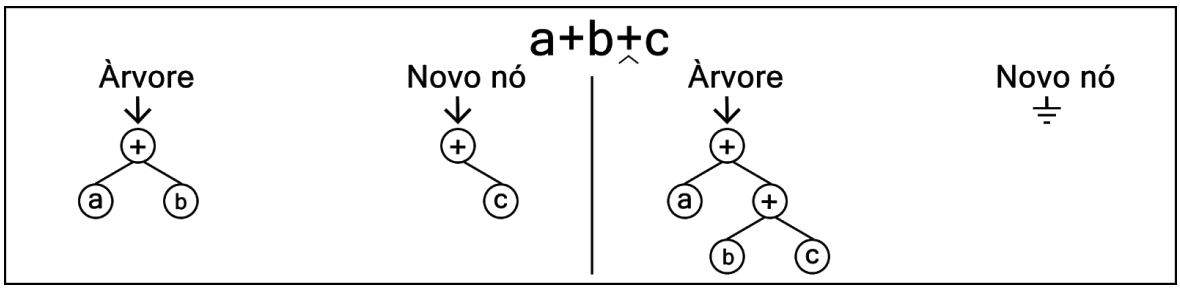

Figura 4. Caso 4 de inserção na árvore.

Todos os nós da árvore binária de operadores possuem um número inteiro associado a eles. Porém quando a árvore é montada, apenas as folhas (todas as variáveis) guardam um valor válido. Ele é diferente para cada variável e muda de acordo com a quantidade de entradas. Esse valor vem da tabela verdade, onde cada variável recebe o número binário de sua coluna convertido para um decimal, como mostra a Figura 5.

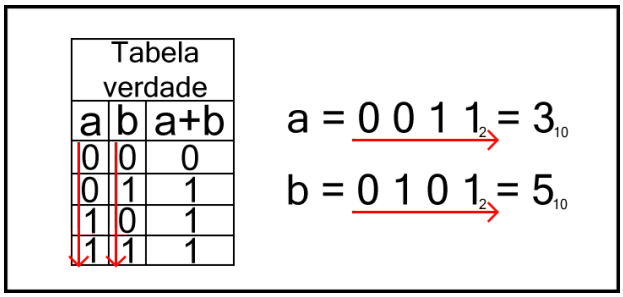

Figura 5. Demonstração de como é gerado o número inteiro de uma variável.

Para calcular a expressão booleana percorre-se a árvore binária de operandos, onde cada nó possui um valor associado. Isso trás a possibilidade de calcular as operações lógicas diretamente, como uma única AND ou OR entre dois números, sem a necessidade de percorrer a tabela verdade. O cálculo é realizado a partir das operações de maior prioridade, como podemos ver na Figura 6.

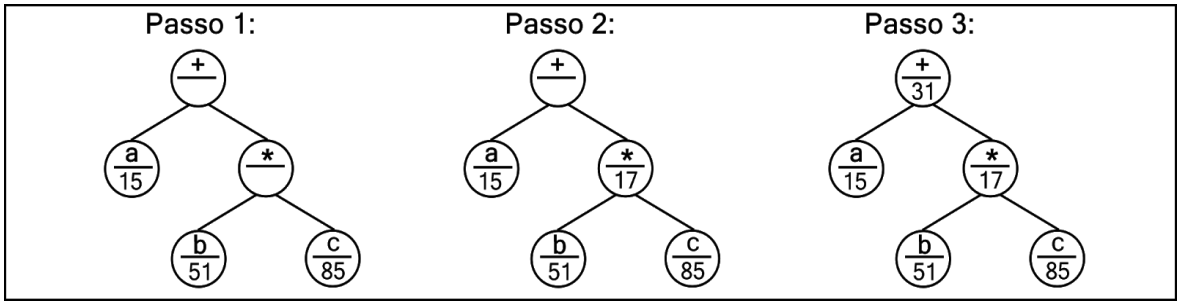

Figura 6. Cálculo da árvore de operadores.

Com isso, dada duas expressões, consegue-se gerar e calcular suas árvores, verificando sua equivalência lógica. A equivalência ocorre quando o valor inteiro associado às raízes das árvores são iguais.

3. RESULTADOS E DISCUSSÃO

O método proposto foi desenvolvida na linguagem C. Para avaliar a eficiência, foi utilizado como *benchmark* todas as funções *p-class* de quatro entradas (*p-class4*), no qual é composta por 3982 funções. Cada função foi avaliada consigo mesma para validar a equivalência lógica. Foi aferido o tempo total de execução para todo o conjunto de funções. A Tab. 1 ilustra o tempo total de execução do método proposto para a *p-class4*. O experimento foi realizado em um computador com processador Intel I5-3470, 8GB de memória ram, utilizando um sistema operacional Linux.

Tabela 1. Tempo de execução para a *p-class4*.

Benchmark	***Número de Variáveis***	***Número de Funções***	***Tempo de execução***
p-class 4	4	3982	3,86s

4. CONCLUSÕES

O método apresentado tem como objetivo verificar a equivalência lógica entre expressões Booleanas, usando árvores binárias de operadores e atribuindo valores inteiros que representam uma tabela verdade. Devido a isso o método é considerado uma forma canônica para representação de funções. O método apresentou um tempo de execução aceitável para um número considerável de funções. Sendo assim, o método proposto pode ser incorporado em ferramentas de projetos de circuitos digitais. Para realizar a equivalência lógica, como exemplo, na etapa de minimização lógica ou na parte de mapeamento tecnológico.

5. REFERÊNCIAS BIBLIOGRÁFICAS

GEREZ, S. H. ***Algorithms for VLSI design automation***. Vol. 8. Nem York: Wiley, 1999

HASSOUN, S., TSUTOMU S., eds. ***Logic synthesis and verification***. Vol. 654. Springer Science & Business Media, 2012.

REIS, A., RASMUSSEN, A. B., DA ROSA Jr., L. S., RIBAS, R. P. Fast boolean factoring with multi-objective goals. In: Proceedings of the International Workshop on Logic and Synthesis. IWLS'09, Berkeley, CA, USA. 2009