Gerador de Estímulos para Teste de Circuitos Integrados

Autor: Pablo Rafael Bodmann
Orientador: Renato Perez Ribas
Instituto de Informática – UFRGS

1. INTRODUÇÃO

A etapa de teste dentro de um projeto de um circuito integrado é muito importante para que o circuito projetado esteja dentro das especificações estipuladas no projeto e testar seus limites operacionais em ambientes extremos. Para isso, é desejável cobrir todas as possibilidades de entradas e suas transições, mas variando apenas uma entrada por vez.

2. METODOLOGIA

Como uma combinação pode transicionar para mais de uma possibilidade pode-se modelar o problema como um grafo. A figura 1 contem um grafo exemplo que representa as entradas e as transições de um circuito de três entradas.

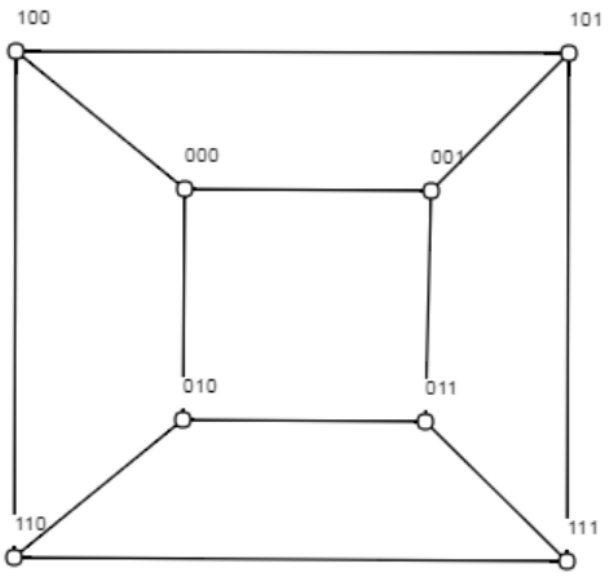

Figura 1: Grafo para um circuíto de 3 entradas

Neste grafo deseja-se passar por todos os vértices e arestas, para isso transforma-se o grafo em uma árvore onde a raiz contém todas as entradas em zero e ordena-se os filhos utilizando a codificação de Gray deixando que os filhos sejam sempre maiores que os pais. Repete-se alguns vértices para ter todas as transições possíveis. A figura 2 contém a árvore criada a partir do grafo da figura 1.

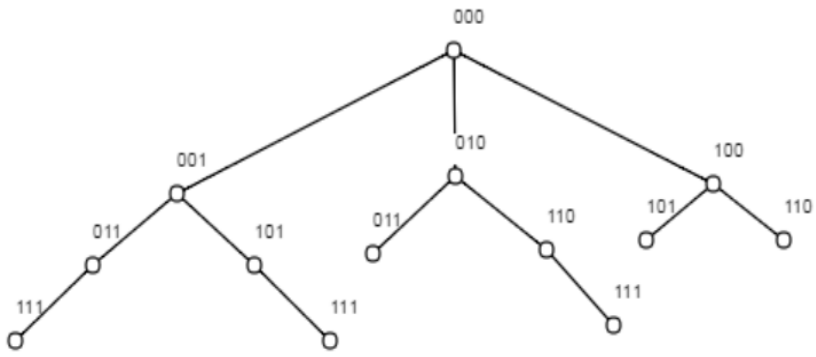

Figura 2: Arvore criada a partir do grafo (Figura 1)

Tendo a árvore, percorre-se ela em profundidade e lista os nós que foram percorridos formado um vetor de excitação.

Para testar o algoritmo foi criado um programa na linguagem Java que criava a árvore representando-a em uma tabela que contém os nós e seus filhos (Tabela 1) e monta o vetor de excitação.

Nós	000	001	010	011	100	101	110	111
Nós alcançáveis	001	011	011	111	101	111	111	
	010	101	110		110			
	100							

Tabela 1: Implementação da árvore

Na aplicação desenvolvida o vetor de excitação é criado percorrendo a tabela mostrada acima(tabela 1). Começa-se no índice 000 da tabela e transiciona-se para o primeiro estado da lista. O mesmo é feito para os nós conseguintes, mas se não houver um próximo estado retorna-se para o estado do qual se partiu, marca-se aquela transição como já visitada e transiciona-se para o estado seguinte da lista. O algoritmo termina quando chega-se ao final da lista do nó 000.

3. RESULTADOS

Para testar o vetor de excitação foi, descrito o comportamento de 4 circuito combinacionais, (portas lógicas AND, OR, *exclusive*-OR, somador completo) e 8 circuitos sequências (*latch* tipo D, *latch* tipo D com sinal de *set* assíncrono, *latch* tipo D com sinal de *reset* assíncrono, *latch* tipo D com sinais de *set* e *reset* assíncronos, *flip-flop* tipo D, *flip-flop* tipo D com sinal de *set* assíncrono, *flip-flop* tipo D com sinal de *reset* assíncrono, um *flip-flop* tipo D com sinais de *set* e *reset* assíncronos). Com o comportamento se extrai um gabarito estático (contém as entradas e suas saídas esperadas) e um gabarito dinâmico (transições das entradas e as transições esperadas nas saídas) para cada porta. Os circuitos foram estimulados com a sequência criada e calculado a cobertura dos gabaritos. Os circuitos combinacionais tiveram cobertura de 100% tanto nos gabaritos de estado quanto nos gabaritos de transição, mas os circuitos sequenciais testados tiverem coberturas menores que 100% nos gabaritos de estado e nos gabaritos de transição. A tabela 2 contem os resultados de cobertura para as portas sequências.

Circuito	Cobertura Estática	Cobertura Dinâmica
latch tipo D	100,00%	66,66%
latch tipo D com sinal de set assíncrono	100,00%	80,00%
latch tipo D com sinal de reset assíncrono	96,67%	80,00%
latch tipo D com sinal de set e reset assíncronos	100,00%	88,89%
flip-flop tipo D	93,75%	50,00%
flip-flop tipo D com sinal de set assíncrono	97,22%	66,66%
flip-flop tipo D com sinal de reset assíncrono	94,44%	66,66%
flip-flop tipo D com sinal de set e reset assíncronos	100,00%	80,00%

Tabela 2: Resultados de cobertura dos circuitos sequenciais

4. CONCLUSÃO

Isso se deve ao fato que circuitos sequenciais têm o estado atual dependente das entradas e da sequência de entradas antecessoras do mesmo enquanto os circuitos combinacionais dependem apenas das entradas atuais. Uma solução encontrada foi utilizar vários circuitos iguais mas com as entradas permutadas ou negadas.

5. REFERÊNCIAS

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Third Edition. MIT Press and McGraw-Hill, ISBN 0262032937.
2. Albert Nijenhuis and Herbert S. Wilf. *Combinatorial Algorithms for computer and calculators,* Second Edition. Academic Press, ISBN 1483273458