Implementarea unui algoritm de sinteză automat

Iuliana Arcana

Rezumat

Lucrarea de față își propune implementarea unui algoritm de sinteză automat pentru *limbajul de descriere hardware* **Verilog**. Acest limbaj este folosit pentru modelarea sistemelor electronice, cel mai adesea fiind utilizat în proiectarea și verificarea circuitelor digitale la nivelul transferului de date între registre (RTL – *Register Transfer Level*).

Tehnologia circuitelor integrate bazată pe materiale semiconductoare a progresat rapid, ajungându-se să se producă circuite cu peste un milion de dispozitive (*Very Large Scale Integration* technology). Datorită complexității acestora, verificarea clasică a lor pe o plăcuță de testare (breadboard) a devenit imposibilă. În acest sens, s-au dezvoltat programe software care să ajute în proiectarea și verificarea acestor circuite, limbajul Verilog fiind un punct de plecare în acest sens.

Faza de proiectare a unui astfel de circuit presupune trei etape: conceptualizare și modelare, sinteză și optimizare, și validare. Obiectul acestei lucrări îl constituie a doua etapă. Prin sinteză se urmărește rafinarea modelului, de la unul abstract (descris în Verilog în acest caz) la unul detaliat, care are toate caracteristicile necesare fabricării. Optimizarea merge adesea mână în mână cu sinteza, scopul fiind îmbunătățirea calității circuitului. Prin calitate se înțelege performanță și o arie cât mai mică folosită pentru realizarea fizică a circuitului.

Tehnicile de sinteză și optimizare sunt folosite în cazul majorității circuitelor digitale și, fără îndoială, puterea lor nu a fost exploatată la maxim. Astfel, proiectul își propune analizarea și realizarea sintezei plecând de la o descriere hardware a unui circuit și obținerea în final a structurii microscopice (i.e., gate-level) a acestuia.

Punctul de plecare în realizarea celor propuse îl constituie interpretarea limbajului în care este descris circuitul digital, adică crearea unui parser pentru limbajul Verilog. Uneltele software folosite pentru implementarea interpretorului sunt Flex (pentru analiza lexicală) și Bison (pentru analiza semantică). Rezultatul acestui proces este arborele abstract de sintaxă (AST – *Abstract Syntax Tree*), în care se reține toate informațiile necesare pe baza cărora se va obține netlist-ul – descriere hardware prin porți logice a circuitului inițial. Netlist-ul poate fi folosit ulterior pentru simularea circuitului și verificarea funcționalității acestuia înainte de a fi fabricat pe cip.

Aplicația a fost dezvoltată în limbajul de programare C, pe sistemul de operare Debian GNU/Linux, versiunea 9 Stretch. Rezultatele pot fi vizualizate grafic cu ajutorul aplicației GraphViz. Interpretarea limbajului Verilog a presupus o etapă de preprocesare a codului, în care fișierul/fișierele cu extensia „.v” sunt prelucrate în așa fel încât să se obțină noi fișiere sursă mai ușor de analizat de către interpretor. Prin prelucrare se înțelege eliminarea comentariilor, substituirea liniilor ce încep cu `*include* cu textul header-ului inclus și substitirea în cod a constantelor simbolice cu valoarea acestora. În cele ce urmează sunt prezentate cele mai semnificative rezultate ale simulării aplicației pe un fișier sursă.

Tabel 1. Rezultatul preprocesării

Înainte de preprocesare	*După preprocesare*

//file test.v
/*comments to be removed*/
 	
module test(a,b,is_lower);
 	
parameter TRUE = 1;
 	
input a,b;
output is_lower;
 	
assign is_lower = (a<b)? TRUE:0;
 	
endmodule
	
	module test(a,b,is_lower);
	
	parameter TRUE = 1;
	
	input a,b;
	output is_lower;
	
	assign is_lower = (a<b)? TRUE:0;
	
	endmodule

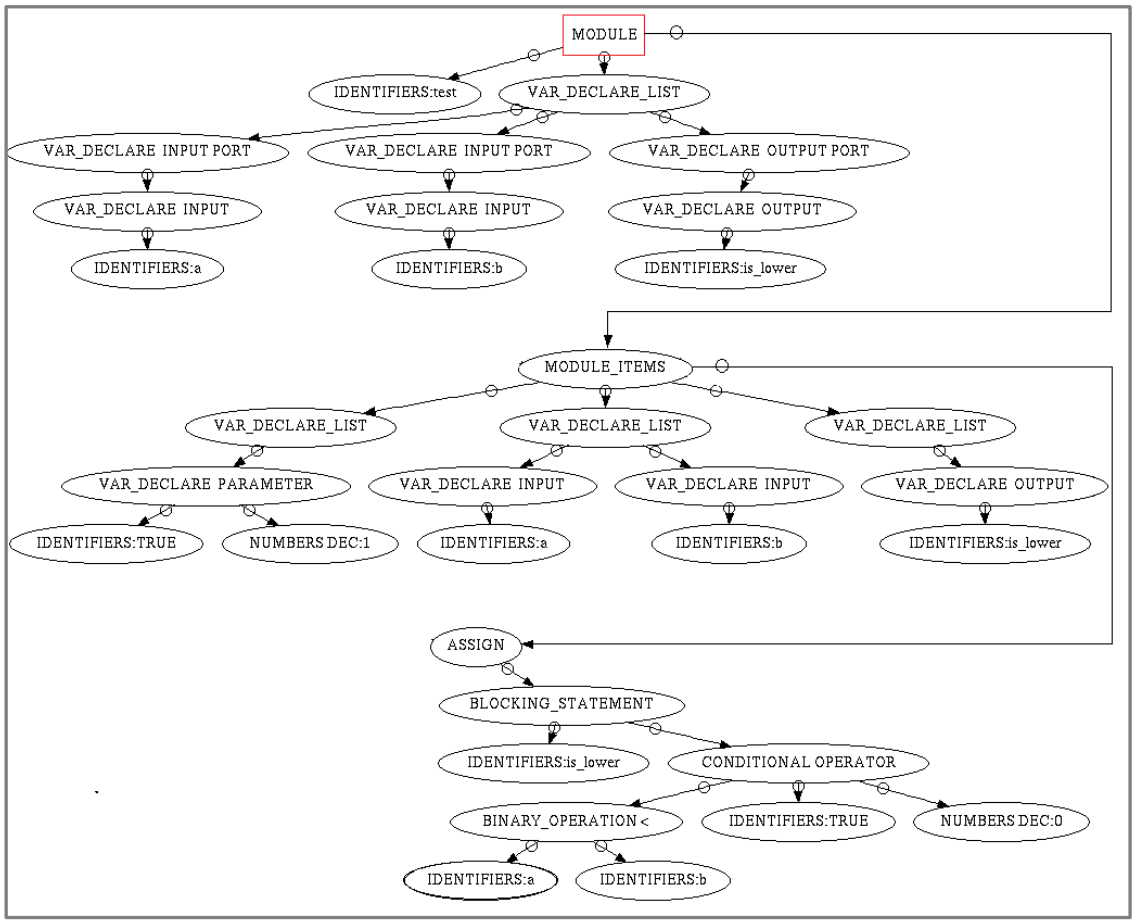

Figura 1. Arborele de sintaxă

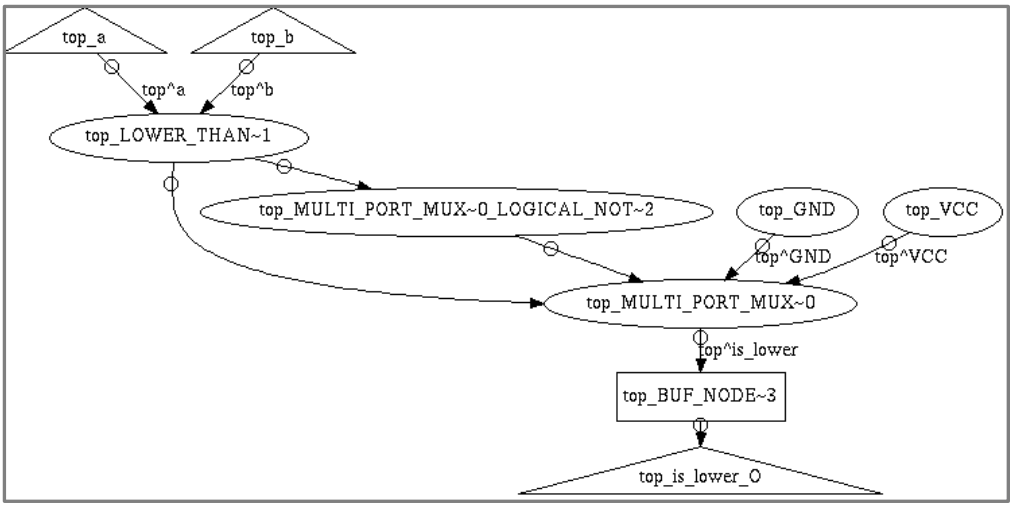

Figura 2. Vizualizarea grafică a netlist-ului obținut