BDD Variable Ordering in HSIS

Adnan Aziz
Email: adnan@cs.berkeley.edu

The time and space needed for verifying a system on interacting FSMs is closely linked to the size of the BDD's created in this process. For a given function, the size of the BDD representing it can be very sensitive to the ordering of the variables in the support of the function. The variable ordering routine provided by HSIS uses a communication complexity based approach which is described in Aziz el. al. [1]. The goal is to minimize the transition relation of the product machine. There are situations where the BDD for the transition relation is not the dominating factor. In this case dynamic reordering should be invoked via the *dro_on* command.

Several variations of the basic algorithm exist. In addition, varying amounts of information on the ordering process can be obtained. This can be done by using the set command at the prompt before reading blif_mv. Experience indicates that no single approach works best - you can experiment with variations by setting flags.

order_mode - { *manual_order, local, global, global_interleaved* } (Default - *global*) If this flag is set to *manual_order*, the system will ask for the name of the file defining the ordering. This file should contain the hierarchical names of the wires corresponding to the outputs of the latches in the design, separated by white space. The order in which the latches occur is the ordering given to the latch variables.
As a rule of thumb, if the design contains units which operate on bit vectors, then the *global/global_interleaved* options should be used; otherwise the local option is appropriate.

rlmv_pass - { *1, 2* } (Default - 1) If this flag is set to 2, the true dependencies of variables will be extracted, and hence the communication complexity can be better estimated.

order_time - { *0, 1* } (Default - 0) If this flag is set to 1, the time taken at various stages of the ordering will be printed out.

order_print - { *0, 1* } (Default - 0) If this flag is set to 1, the ordering of the latches is printed out. A latch is identified by the hierarchical name of the output wire, eg processor1.cache3.reg5

order_heuristic - { *la1, la2, touati* } (Default - *la1*) Using *la1* will invoke a greedy algorithm with a lookahead of 1, similarly *la2* will yield a greedy algorithm with a lookahead of 2. For 50 or more latches, *la2* takes unacceptably long. Using *touati* will lead to an algorithm that orders latches by minimizing the cumulative support.

order_debug - { *0, 1* } (Default - 0) If this flag is set to 1, intermediate results of the ordering will be printed. This is meant for developers only.

References

1. A. Aziz, S. Tasiran, and R. K. Brayton. BDD Variable Ordering for Interacting Finite State Machines. In *Proc. of the Design Automation Conf.*, June 1994.

This article was processed using the LaTeX macro package with LLNCS style