ABSTRACT

Application of behavioural transformations for obtaining optimal performance, energy and/or area on a given platform during embedded system design is now a common practice. Verifying correctness of these transformations is an important step in ensuring dependability of embedded systems. This thesis addresses verification methodologies, primarily by way of equivalence checking, for six behavioural transformations that are applied during embedded system design. The transformations considered cover code motion, generation of register transfer level (RTL) design after carrying high-level optimizations and also RTL transformations, loop and arithmetic transformations on array based programs, transformations on array based programs leading to the generations of Kahn process networks (KPN) to achieve high degree of parallelism and also transformations applied at the KPN level.

Verification methods for the first three transformations on programs not involving arrays employ the model of finite state machines with datapaths (FSMD). FSMDs are generalizations of FSMs to capture data transformations taking place between control states. FSMD based equivalence checking method consists in introducing cutpoints in one FSMD, visualizing its computations as concatenation of paths from cutpoints to cutpoints and finally, identifying equivalent finite path segments in the other FSMD; the process is then repeated with the FSMDs interchanged. Verification methods for the last three transformations involving arrays employ the array data dependence graphs (ADDG). The ADDG model primarily captures computation of a range of elements of an array from ranges of elements of other arrays. ADDG based equivalence checking attempts to show that the overall computations depicted in the ADDG to define the final ADDG nodes in terms of the initial ADDG nodes are equivalent, both in terms of computation and range of array elements, with respect to those in the ADDG being compared.

The FSMD based methods developed handle both uniform and non-uniform code motion transformations. For non-uniform code motions model checking of some data-flow properties is also carried out. A rewriting mechanism covering both pipelining and multicycling is used to construct an FSMD from register transfer operations for RTL related equivalence checking. In our ADDG equivalence checking method we have introduced the notion of slices to guide the checking along the sequences in which operations are used to define array elements in the original behaviour. Normalized representation of arithmetic and conditional expressions play a central role in handling arithmetic transformations for both FSMD and ADDG based equivalence checking. Our ADDG based checking has been extended to check the equivalence of KPN networks derived by parallelizing sequential array based programs. Potential deadlocks in the KPN are also detected through the ADDG based modelling.

Correctness and complexity of the all the developed methods have been treated formally. The methods have been implemented and tested on several benchmarks. This work represents useful application of equivalence checking

as a verification method for verification to several aspects of the embedded system design flow.

Keywords: Embedded System, Formal Verification, Equivalence Checking, Behavioural Transformation, Code Motion, Loop Transformation, Register Transfer Level (RTL), Kahn Process Network, Finite State Machine with Datapath (FSMD), Array Data Dependence Graph (ADDG).