Extracting Finite State Machine with Datapath models from the synthesized behavior in High Level Synthesis

A Synopsis submitted in partial fulfillment of the requirements for the degree of

Master of Technology in Computer and Information Technology

by

Satish Bonagiri
(06cs6006)

Under the guidance of

Dr. Dipankar Sarkar
and

Dr. Chittaranjan Mandal

Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur

April 2008
1 Introduction

1.1 High-level Synthesis

Synthesis is the process of translating a behavioral description into a structural description. Synthesis is also defined as the process of interconnecting primitive components at a certain level of abstraction (target level) to realize a specification at a higher level of abstraction (source level).

Several levels of the synthesis process are system synthesis, high level synthesis, logic synthesis and layout synthesis. High level synthesis (HLS) takes algorithmic or high level behavior as input and outputs the register transfer level (RTL) description consisting of functional units, storage and interconnecting units. Individual synthesis systems cater to different constraint goal sets. Typical user constraints are area, clock speed and power.

The high-level synthesis (HLS) process consists of translating a behavioral specification into an RTL structural description containing a data path and a controller so that the data transfers under the control of the controller exhibit the specified behavior. The input behavioral description to the HLS is represented as a control data flow graph (CDFG) and the output of the HLS is a structural data path of interconnected components and a controller. The synthesis process consists of several interdependent sub-tasks such as, scheduling, allocation, and binding and controller design. The operations in the behavioral description are assigned control steps through a scheduling process. Each control step is the basic time unit of the synchronous digital system. The allocation process computes the minimum number of functional units and registers required to synthesize the design based on the scheduling information of the operations and the operators available in the component library. The variables are mapped to registers and the operations are mapped to functional units by the binding process. Finally, the controller is designed based on the data transfer required among the data path elements in different control steps.

1.2 High level synthesis in SAST

SAST is a genetic algorithm based HLS tool designed to synthesize the behavioral specification into a structured architecture framework. This tool is depicted in figure 1. This structured architecture (SA) leads to interconnect optimization. The datapath is organized as architectural blocks (A-block). Each A-block has a local functional unit (FU), local storage and internal interconnections. SA also permits the use of memories as architectural components. These are connected to the global buses like the A-blocks. These structured data paths avoid random interconnections between datapath elements. Each A-block has a
simple implementation. This makes the generated design easy to implement on programmable devices such as FPGAs.

The correctness of HLS process is verified in three phases. The phase-I verifies the scheduling process. This phase is also called as *scheduling verification*. The input of this phase is the CDFG and the output is the scheduled behavior. In phase-II, the datapath generated after allocation and binding is verified against the scheduled behavior. We will verify the registers sharing among the variables of the input specification. This phase is called as *datapath verification*. In phase-III, the controller will be verified against the data path. This phase is called *controller verification*. Verification task of this phase involves checking the correctness of the control signals.
1.3 F SMD

An FSMD (finite state machine with data-path) is a universal specification model, proposed by Gajski, which can represent all hardware designs. An FSM (finite state machine) model works well for up to several hundred states. Beyond that, the model becomes incomprehensible to human designers. To adapt the FSM model for more complex designs, FSMD was introduced by Gajski. Each storage elements like the register is replaced by a variable in the FSMD. So, each variable replaces thousands of different states. For example, a 16-bit register represents $2^{16}$ different states in an FSM; thus, introduction of a 16-bit variable reduces the number of states in the FSM model by $2^{16}$. The use of variables leads to the concept of an FSM with a datapath (FSMD).

2. Problem Identification

The interfaces are added at various stages of the HLS process to extract the FSMDs required for verification. In the existing SAST there is no FSMD construction module after register allocation and binding, which is needed for verification.

3. Work done

The goal of present work is related to generating the FSMD after register allocation and binding phase for verifier.

We have started formation of FSMD from register allocation and binding results. In a typical HLS process after Allocation and Binding, data path of the input specification is formed. Data path consists of three register transfer logic (RTL) components: functional, storage and interconnections. The input variable or signal gets mapped to a storage units in the data path namely register. Functional units perform the operations assigned to it in each control step by its composition. Data transfers between the functional and interconnection units in the data path comes from or goes to storage units.

Moreover the same register can be used to represent different variables, provided they are live at non-overlapping intervals. FSMD formed after register allocation and binding is a mapping of register names to the variables and signal names. FSMD formed at this phase is used for verification of correct register sharing among the variables in the CDFG. FSMD formation at this stage is forming a function which has as inputs the scheduled CDFG and the register
liveness information (intervals in which variables to certain register) or register sharing information.

In SAST we have two types of variables involved in our data path namely permanent and temporary. Permanent variables are one which binded permanently to a register in a Alu-block for their entire lifetime. Temporary variables on the other hand can be binded to more than one register in different Alu-blocks in their lifetime, as they are required as operands to some functional operation scheduled in other Alu-block. So firstly we extract the lifetimes of variables with their span registers along with the control step in which they bind to the register on a per block basis. Example of such extracted register

Lifetime information for GCD is:

```
NV q-1 q-1 R1
y1 q0 q1 R0
y1 q-1 q2 R0
y1 q0 q3 R0
y1 q1 q3 R0
y1 q3 q4 R0
y1 q2 q5 R0
y1 q3 q5 R0
y2 q0 q1 R1
y2 q-1 q2 R1
y2 q0 q3 R1
y2 q1 q4 R1
y2 q2 q4 R1
y2 q4 q5 R1
z q5 q6 R1
```

After extracting the variable-register binding information, we traverse the already formed scheduled FSMD and map the variables with their register counterparts depending upon their states or control states. This forms the FSMD with variables mapped to registers and the 2 FSMDs namely scheduled and this one are used for verifying correct register sharing. The FSMD formed after scheduling (GCD) is shown in Figure 3.1 and Figure 3.2 shows FSMD formed after allocation and binding for GCD.
Figure 3.1  FSMD formed after scheduling phase

Figure 3.2  FSMD formed after allocation and binding phase
References:


