Embedded System Design and Modeling

EE382V, Fall 2008

Lecture Notes 6

System Specification and Analysis

Dates: Oct 2 & 7, 2008
Scribe: Mahesh Prabhu

Capturing the Specification Model:

We start by creating a golden model that has the following properties:
(1) should be abstract and should only capture the functionality.
(2) should not have implementation details.

In reality it is hard to have such a model. For example consider the problem of matrix multiplication. It would be wise to give a specification that captures the data accesses taking into consideration the cache structure. Similarly in hardware we can have pipelined execution included in the specification. Hence for some problems it would be unavoidable not to include some implementation details within the model specification.

Test Bench Creation:

The test bench has 3 parts: Design Under Test (DUT), Stimulus and Monitor.
The stimulus and monitor can use all the syntax and semantics of the SpecC language without worrying about the restrictions as the test bench won't be synthesized. The DUT however will be synthesized hence would have to adhere to SpecC modeling guidelines.

Below is an example of a stimulus and a monitor interacting with a DUT.

```
Behavior Main {
        c_queue CI(5ul), C)(5ul);
        Stimulus s(CI);
        Design dut(CI, CO);
        Monitor m(CO);

        void main ( void) {
                par {
                        s;
                        m;
                        d;
        }
}
```

```
Behavior Stimulus (i_sender Q) {
        void main (void) {
                FILE *f = fopen(“input”, “r”);
                while(fread(..) ) {
                        waitfor(20000); // Simulate the feeding of an input every 20ms
                        Q.send(...);
                }
                waitfor(30000); // wait for sometime in the end so that the design is
                                 // able to process the last packet
                exit(0);        // Exit the simulation. Some form of Exit should always
                                // be there.
        }
};

#include <sim.h>
Behavior Monitor (i_receiver Q) {
        void main(void) {
                fopen(..);  // Open the file into which the result will be written
                while (1) {
                        Q.reveive(...);
                        fwrite(...); // Write the result into the file
                        printf(“%llu”, (unsigned long long)now()); // Print the current
simulation time
                }
        }
};
```

Test bench should be sophisticated enough to check for the timing as well as the functionality of the design.

Modeling Guidelines

The hierarchy should be clean and the communication should be through the standard channels. Sophisticated software mechanisms like callback functions should not be allowed.

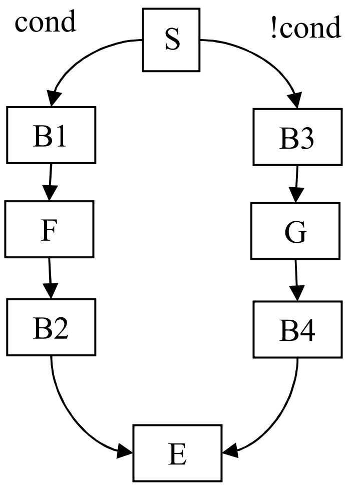
Converting C code to SpecC :

(1) Move global variables low into behaviors and communicate their values through channels.
(2) Partitioning arrays helps in splitting up data to exploit parallelism.
(3) Floating point to fixed point conversion required for H/W implementation. This conversion should be done so that accuracy of computation is not compromised.

Consider the conversion of the following if-else code in C to a SpecC implementation.

```
If (cond) {
        ...  // Convert this code to Behavior B1
        f(); // Convert this to behavior F
        ...  // Convert this code to Behavior B2
} else {
        ...  // Convert this code to Behavior B3
        g() // Convert this code to Behavior G
        ...  // Convert this code to Behavior B4
}
…
```

We can convert the above piece of C code to a SpecC FSM with a clean hierarchy.



Profiling and Estimation:

Fidelity: this term is used indicate the relative accuracy of an estimate, i.e. that estimates for different targets will give accurate trends in relation to each other. In the cases where fidelity is used to measure the accuracy of an estimation, the absolute values of estimate numbers must be ignored. What matters in these cases is only the difference in numbers from the estimate from one model to another.

Static Analysis Tools:
Problems:

(1) Relies on annotation from the user and this annotation might be inaccurate.
(2) These tools generate false paths which might not be a possibility in the actual model.
(3) Not very effective on high level models because its difficult to analyze statically some behavioral level constructs like loops(how to identify the number of times the looping happens).

Dynamic Analysis Tools:
These tools work mostly by performing simulations and back annotating information obtained after first round/s of profiling. For example in case of SCE, the tool back annotates waitfors into the code to simulate timing.

Instrumentation Based Profiling:
This profiling gives an idea of the amount of computation involved for a given set of test-vectors for a given model.

Re-targeting:
Profiling after re-targeting involves inserting waitfors for each operation. The waitfor time amount is determined based on how computation intensive the operation is. This is called weight table based model. Finally the exact simulation cycles are not important, only relative numbers are important.
The other alternative to get more accurate estimates is to do more accurate estimation. The software part of the model can be compiled into intermediate code and then use this code to perform symbolic simulation on a micro-architecture model of the target processor. This would give better estimates because of the more fine-granular modeling of the target hardware.