On verifying AhirV2 generated VHDL using software testbenches

Madhav Desai
Department of Electrical Engineering
Indian Institute of Technology
Mumbai 400076 India

August 16, 2011

The AhirV2 tool chain can be used to convert parts of a C program to VHDL (essentially, some of the functions in a program are mapped to VHDL). To verify the resulting VHDL, one would like to simulate it in a VHDL simulator (such as Modelsim from Mentor Graphics). The most natural way to do this is to use the original program itself as a testbench for this purpose.

- Stubs are created for the set of functions which are mapped to VHDL by the AhirV2 flow.
- The software testbench is compiled and linked with these stubs.
- Whenever a stub function is called, it tries to connect with a server created by the VHDL simulation process.
- The VHDL simulation process listens for calls from the stubs and exchanges data between the stubs and the actual VHDL being simulated.

1 An example

Consider the following program (lets say it is in file "prog.c"):

```
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include <iolib.h>

uint32_t sum;

void set_sum(uint32_t x)
{
    sum = x;
```

```
}

uint32_t get_sum()
{
    return(sum);
}

void accumulate()
{
    while(1)
    {
        int nxt = read_uint32("in_data");
#ifdef SW
        printf("read %u\n", nxt);
#endif
        sum = (sum + nxt);
        write_uint32("out_data",sum);
#ifdef SW
        printf("wrote %u\n", sum);
#endif
    }
}
```

This program describes a *system* which listens for data on a pipe "in_data", and sends data out on a pipe "out_data". The incoming data is accumulated into the variable *sum*, and there are two methods to set and get the value of *sum*.

Now to test this program, we can write a test-bench such as this one (lets call this file "testbench.c").

```
#include <pthread.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

#ifdef SW
// for the read_*/write_* methods
#include <iolib.h>
#include "prog.h"
#else
// includes for the read_*/write_* methods
// as well as stubs for functions moved
// to hardware.
#include "vhdlCStubs.h"
#endif
```

```
void Exit(int sig)
{
   fprintf(stderr, "## Break! ##\n");
   exit(0);
}

void *accumulate_(void* fargs)
{
   accumulate();
}

void *write_pipe_(void* a)
{
   write_uint32_n("in_data",(uint32_t*)a, 10);
}

void *read_pipe_(void* a)
{
   read_uint32_n("out_data",(uint32_t*)a, 10);
}

int main(int argc, char* argv[])
{
        signal(SIGINT,  Exit);
        signal(SIGTERM, Exit);

        uint32_t data_in[10], data_out[10];
        int i;

        // initial value of sum.
        set_sum(1);

        for(i = 0; i < 10; i++)
        {
          data_in[i] = i;
        }

        pthread_t acc_t, wpipe_t, rpipe_t;

#ifdef SW
        pthread_create(&acc_t,NULL,&accumulate_,NULL);
#endif
        pthread_create(&wpipe_t,NULL,&write_pipe_,(void*)data_in);
        pthread_create(&rpipe_t,NULL,&read_pipe_,(void*)data_out);

        pthread_join(wpipe_t,NULL);
```

```
        pthread_join(rpipe_t,NULL);


        fprintf(stdout,"from out_data, we read ");
        for(i=0; i < 10; i++)
          fprintf(stdout," %u ", data_out[i]);
        fprintf(stdout,"\n");
        fprintf(stdout,"final sum is %u\n", get_sum());

#ifdef SW
        pthread_cancel(acc_t);
#endif
}
```

The test-bench sets an initial value for sum, and starts three threads: one to write data to pipe in_data, one to read data from pipe out_data, and one to run the accumulate function. After the last data is read back from out_data, the test-bench gets the value of sum and prints it out.

Obviously, we would prefer to use the same test-bench to verify that the VHDL system generated from "prog.c" functions correctly. The difference is that instead of using methods in iolib, the test-bench now uses methods in SocketLib. Further, the VHDL is executed in a VHDL simulator; the simulator communicates with the testbench using sockets. The *ifdef's* in the test-bench and the system program indicate the difference between the pure software version of the system-test-bench combination and the hardware-software version.

2 An example

Look at the subdirectory "example" which contains the files "prog.c" and "test-bench.c". A Makefile and README are also present. Its all quite self-explanatory.