Inserting an AHIR module in the NetFPGA framework

Madhav Desai

May 30, 2011

1 Overview

This document describes the mechanism by which an AHIR generated VHDL system can be included in the NetFPGA packet processing implementation framework.

The AHIR generated VHDL system is required to fit into a NetFPGA user-module template. It is relatively easy to achieve this.

2 Mechanism

The user module template has the form shown in Figure 1. The interface is very simple: an input data and control bus, and an output data and control bus.

A generic AHIR system can have the form:

Thus, in order to insert a generic AHIR system into a NetFPGA user-module template:

- we ensure that the only interaction of the AHIR system with the outside world is through four pipes: two input pipes for the input data and control, and two output pipes for the output data and control.
- The top-level modules in the AHIR system are free-running. One solution would be to include two top-level free-running modules which are responsible for interactions with the outside world (through the four pipes discussed above).
  - an input-module which will collect data from the input pipes and forward to appropriate pipes/modules in the AHIR system.
  - an output-module which will collect data from internal pipes/modules in the AHIR system and forward it to the the system output pipes.
- The AHIR system would be instantiated in a *fixed* wrapper which would be responsible for the protocol matching between the AHIR system interface and the NetFPGA user module interface.

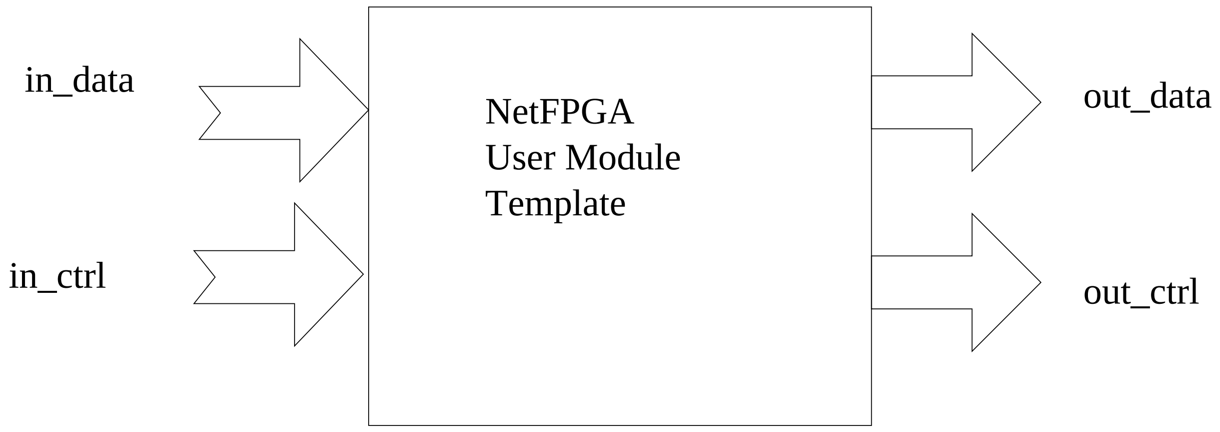

Figure 1: NetFPGA User Module

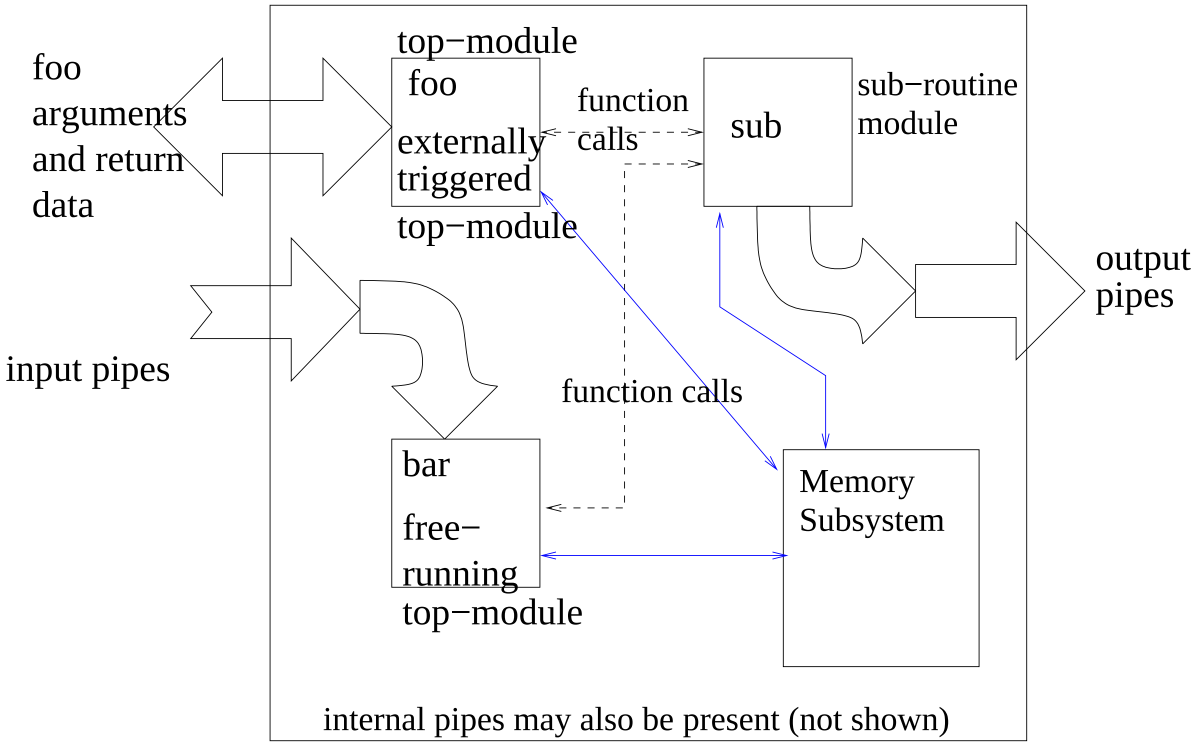

Figure 2: Generic AHIR system

3 Issues: memory management

One issue that needs to be resolved is that of memory management. This relates to two processes:

- Initialization: if the memory subsystem in the AHIR module needs to be initialized, then this must be done explicitly by the outside world. The llvm2aa tool can generate memory initialization modules that initialize the data structures in the system to their initial values as declared in the original C program. These modules are auto-generated and need to be called once after the AHIR system is reset (this call-once feature can be incorporated into vc2vhdl).
- Exchange of information between the AHIR memory subsystem and the outside world: If this is needed, it would need to be explicitly built into the input/output modules. Note that an exchange of information (in general) may be initiated from "inside" or from "outside" the AHIR system.

4 Looking Ahead: a proposal for rapid deployment

The process of including the AHIR system in a NetFPGA user template is exactly the same as that of verifying an AHIR system using a C-testbench with the assumption that the testbench only sends/receives packets to the AHIR system. Most of the basic infrastructure for this is in place. The only necessary components are the input and output modules outlined in the previous section.

Originally, Sameer had implemented the input and output module for a specific scenario (as documented in the previous plan). In this scenario, the input module would receive the data from the NetFPGA surroundings, would put the data packet into memory and would trigger the AHIR module. On completion of the AHIR module, the output module would take the modified data from memory and would forward it to the NetFPGA environment. I have looked at the code and it seems reasonably clean (will probably have to redo parts of it to fit into the new AhirV2 scheme).

What would be easiest, in my opinion would be

- use the parts of Sameer's code that do the protocol translation (between the NetFPGA user-module and an AHIR module).
  - we implement a VHDL entity *NetFpgaAhirSystemWrap* which would instantiate an AHIR system with four interface pipes (see Figure 3).
- write the input and output modules in C/Aa and add to the AHIR flow to generate the final system.

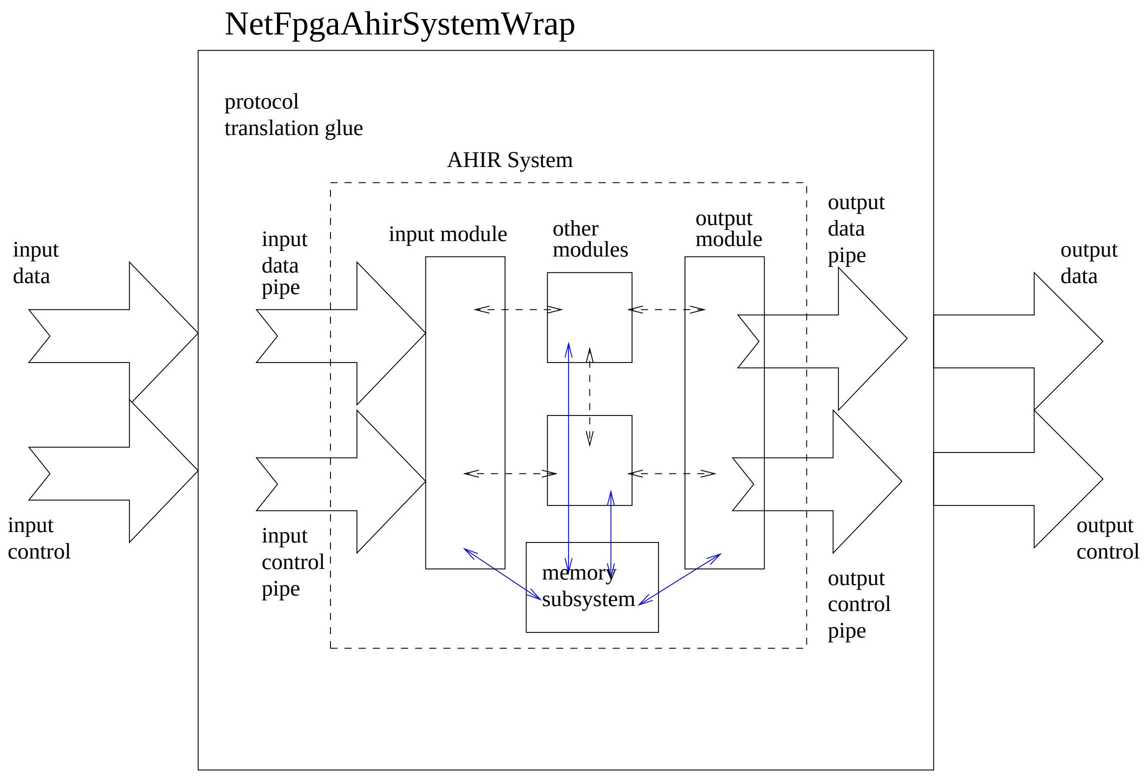

Figure 3: NetFpgaAhirSystemWrap Schematic

  - this can be user-written (and eventually auto-generated) for the specific system being generated. Note that we will have to pay some attention to memory initialization and transfer of information between the AHIR memory subsystem and the outside world.

- generate the final AHIR system by including the input/output modules written in C/Aa.

- Instantiate the final AHIR system in the NetFpgaAhirSystemWrap wrapper.

The process is indicated in Figure 4. The only additional effort is to write the input and output modules (in C or Aa).

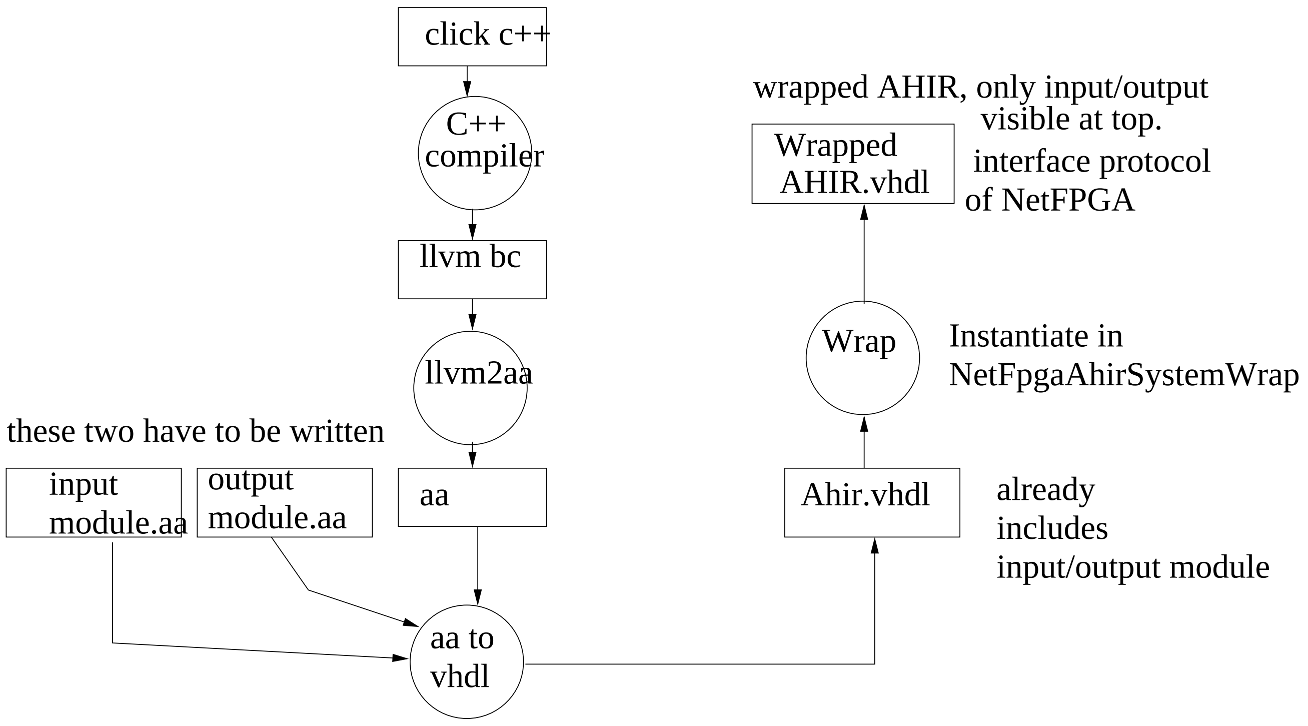
click c++
C++ compiler
llvm bc
llvm2aa
these two have to be written
input module.aa
output module.aa
aa
aa to vhdl
Ahir.vhdl
already includes input/output module
Wrap
Instantiate in NetFpgaAhirSystemWrap
Wrapped AHIR.vhdl
wrapped AHIR, only input/output visible at top.
interface protocol of NetFPGA

Figure 4: NetFPGA Wrap Flow