IITB AI accelerator architecture

Madhav P. Desai
Department of Electrical Engineering, IIT Bombay
email: madhav@ee.iitb.ac.in

March 8, 2021

1 Overview

We describe the architectural base of the AI accelerator algorithm description. The algorithm itself will be described by

- C code.
- **Aa** code.
- Hierarchical system description code (**hsys** code).

The system being described is hierarchical and the hiearchy can be visualized as shown in Figure 1.

The hierarchical description consists of a top-level system, which can in turn instantiate sub-system instances, which are in turn constructed using sub-systems. There are two kinds of sub-system instances

- Hierarchical sub-system instances: these are themselves constructed using lower level instances.
- Leaf sub-system instances: these are defined using **C**, **Aa** or mixed **C/Aa** descriptions and consist of a collection of algorithmic threads.

1.1 Hierarchical sub-systems and their internal connections

Hierachical sub-systems are purely structural and allow us to instantiate sub-systems and connect them up. Connections between sub-system instances can be of two types

- FIFO connections: Information from one sub-system instance can be sent to another sub-system using FIFO buffers. These buffers have two parameters, the width (in bits) of the data that is written into and read from the FIFO, and the depth of the FIFO, i.e., the number of elements it can store.

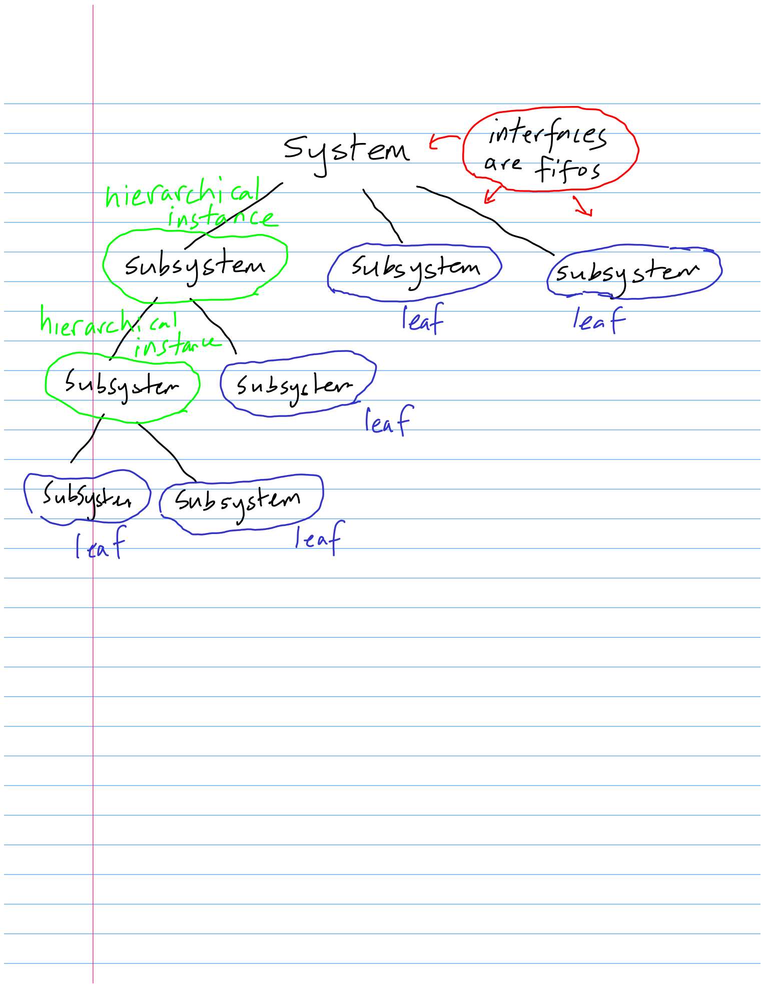

Figure 1: Visualization of the hierarchy

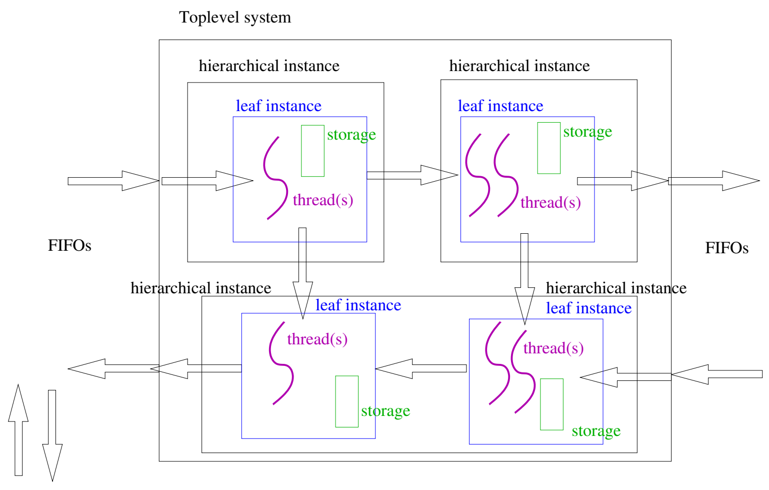

Figure 2: Visualization of a typical system

  - Note that FIFOs are essentially blocking in nature, and serve as communication and synchronization channels.

- Signal connections: These are flags which can be used to exchange information in a non-blocking (asynchronous) manner.

1.2 Leaf sub-systems

Leaf sub-systems are essentially a collection of threads which are described using **C** and **Aa**. These form the algorithmic heart of the overall system.

1.3 Visualization of a typical system

We show a visualization of a typical system in Figure 2.

The top-level system **top** has three hierarchical sub-system instances **top/i1** **top/i2** and **top/i3**. The hierachical sub-system instance **top/i1** has a single leaf-instance **top/i1/i0**, etc. The white arrows indicate FIFO connections between sub-systems.

Each leaf instance will consist of a collection of threads, which can communicate amongst them-selves using either the storage in the leaf instance or FIFOs.

1.4 Summary

The structure of a system is conventional but there are some important rules and restrictions.

- Hierarchical sub-system instances will interact only through FIFOs and signals.

- A sub-system can be defined once and instantiated multiple times.
- All storage will be declared and accessed from within leaf sub-system instances.

2 Towards an AI/ML system description in the AHIR-V2 environment

We start with a hierarchical description.

```
$system ai_ml_engine $library ai_ml_lib
  $in
    $pipe INPUT_DATA
  $out
    $pipe OUTPUT_DATA
{
  $instance ictrl_inst
      ai_ml_lib:input_controller

  $instance mem_inst
      ai_ml_lib:mem_pool

  $instance processing_inst
      ai_ml_lib: processor_element

  $instance octrl_inst
      ai-ml_lib: output_controller
}
```

The system being describe here is the one shown in Figure 3.
The basic flow of information is as follows:

1. An input image is sent into the system using the FIFO from the left.
2. The input controller requests the memory pool to provide a memory area to store the image.
3. The memory pool provides the storage area if available (as a buffer index).
4. The input controller stores the image into the memory pool, and passes the buffer index to the processing element.
5. The processing element works on the image, using the memory pool and its local storage for the processing.
6. Once done, the processing element passes an output data index to the output controller.

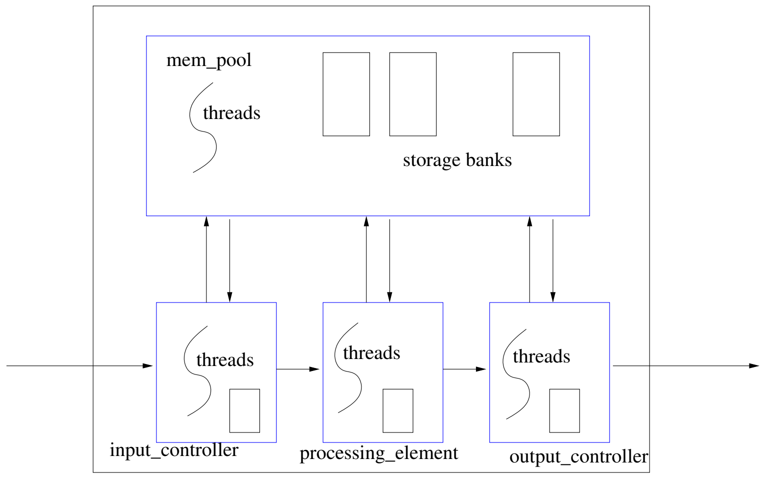

Figure 3: Simple AI/ML example

7. The output controller picks up the output data from the memory pool using this output data index and sends it out on the FIFO to the right.

We will need to describe the threads that constitute the behaviour of the four leaf sub-systems using either **C** or **Aa** or a combination of the two.

3 Towards a templatized AI/ML system description

In this project, we will move a step above the description level described in the previous section.

- We will define a set of leaf sub-system functions that are essential.
- We will develop a parametrized model for a memory pool which can be flexibly used in various configurations (size of memory, parallelism, number of input ports, internal buffer classes).
  - The physical implementation of these memories can be in FPGA block RAM or in off FPGA DRAM.
- We will define a simple language to show the structuring of the AI system using the essential leaf sub-system functions, and memory pools.
- From this description, we will generate a hierachical system description as in the previous section.
- Tools for mapping the hierarchical system description to VHDL are already available as part of the AHIR-V2 tool-set.

3.1 First stage: understand the existing tools, and providing a reference implementation of the leaf functions

Some introduction to the use of **C** and **Aa** for implementing hardware was covered in EE789. However, I will provide a crash training program for familiarization with the AHIR-V2 toolset in the hierarchical system construction context.

We will simultaneously need to have reference implementations for the leaf functions. The milestones are:

- Familiarity with AHIR-v2 tools:
  - Demonstrated on FPGA for an image filter.
- Reference C implementations of leaf functions.
  - Demonstrated on hello_world AI/ML application.

3.2 Second stage: put together a pilot system using the existing AHIR-V2 tools

This will require us to implement the leaf functions in **C** and **Aa** put them together using the AHIR-V2 toolset and validate on FPGA.

We will simultaneously implement a memory pool which is backed by block RAM as well as by DRAM. The milestones are:

- Validated parametrizable memory pool backed by SRAM and DRAM.
- Implementation of leaf functions in **C**/**Aa** demonstrated on hardware.
- Progress towards templatization language.

3.3 Third stage: templatized construction of an AI/ML system

A simple language and assembler to put together an entire system.