Accelerated Database Processor

Codename: lotta-pain

CSEE 4840 Embedded System Design
25 March 2014
Project Design

Andrea Lottarini (al3125)
Timothy Paine (tkp2108)
I. Project Overview
With our project, we plan to implement a complete Q100 Database Processing Unit\(^1\), or DPU, on the SoCKit board. The project can be divided into three major components: a software SQL compiler, a runtime system, and a set of ASIC tiles corresponding to several of the major SQL commands (i.e. Aggregation, Join). The compiler processes the sql query and develops a query plan for how to execute it. Then, based on the configuration of the DPU (i.e. topology, number of tiles, etc), and the data itself, the runtime system determines the pipeline configurations necessary to process the query. Query execution configures the pipeline and loads the correct memory addresses for streaming, and a single operation is executed. This repeats until the full query has been executed.

II. Software Overview
The software is composed of two major components. The first is a compiler, which takes a SQL query and outputs a query plan. This has been in development for some time, and is focused around the TPC-H query suite (may not handle any generic SQL query). For the purposes of this project, the more interesting component is the runtime system. This takes a compiled query plan, information about the DPU (topology, tile counts, etc), and information about the data (range and frequency for partitioning, etc), and configures the pipeline and memory controllers to execute one small portion of the query plan at a time. This system is similar to a just in time (JIT) compilation and we will use these two terms interchangeably. We will be running our JIT on the ARM core, and implement a set of mechanisms for talking to the pipeline peripheral and memory controllers in order to execute these operations. For more info, see section IV.

III. Hardware Overview
The hardware component of our project is composed of two primary structures, a pipeline of SQL-esque tiles, and the streaming memory infrastructure to send data through the pipeline. Each tile is mapped to a SQL operation or an Auxiliary/Utility function. Some examples of SQL op tiles are Joiner, Aggregator, Sorter, Filter. Some utility tiles include Column Concatenation, Column Stitching, Table Partitioning, Boolean generation (for filtering), ALU. In the pipeline, each tile will have a master/slave relationship with the tile before and after it (see signal description in Appendix, based on Avalon ST signals), and the control blocks (everything within the bold line with an arrow pointing to it from the outside) will be configured by the ARM core using the Avalon bus. For the second component of our project, we plan on using the direct memory access

\(^1\): http://arcade.cs.columbia.edu/q100-asplos14.pdf
controller contained in the hard processor system for managing memory transfer, and additional streaming/arbitration stages may be necessary depending on the details of the DMAC.

IV. Hardware/Software Interface

Here is an overview of the system architecture. The “Data” block is presented here as an abstraction (i.e. may be on disk or other medium in the future), but on the SoCKit will be the same as the memory block.
V. Design Goals and Considerations
For the purposes of this class, the goal of the project is to produce a functioning SQL accelerator. However, since this is part of a broader research project, a number of other considerations will be made as the project progresses, such as pipeline topology, tile counts, pipelines with different size operands, etc.

VI. Checkpoints

Milestone 1
HW. All tiles built and translated to system verilog, signal scheme finalized and implemented for all tiles (Avalon ST), general purpose testbench for tile validation, tile-tile communication validated.
SW. Implement a simulator that could be used to validate the software pipeline- this simulator should functionally emulate the DPU tiles and also provide cycle counts.

Milestone 2
HW. DMA integrated, full suite testing (pull data from memory, stream through hard-configured pipeline, stream out to memory)
SW. Use the simulator to test different strategies in the runtime system. Finalize the interface between the software and the hardware.

Milestone 3
HW. Full dynamic configuration and streaming.
SW. Integrate the runtime and the hw.

Final Goals
HW. Optimal pipeline configuration, area/power/timing measurements (rough) for different configurations. Configurations consider number of tiles, number of pipelines, etc.
SW. If time allows implement and test optimizations and/or different strategies within the compiler and the runtime.
Timing

enable indicates that system is running, read indicates whether or not the data coming in (or out) is valid. ready is for backpressure for sorter, and done is for aggregation. This is essentially the avalon standard with a few extra signals added.