Supplementary Material

This package contains:

A) Nine benchmark applications that are used in performing the experiments in the paper listed below.

B) The procedure to prepare these benchmarks, i.e., the procedure to **convert StreamIt benchmark applications to SDF graphs in SDF3 format**. This procedure is general and can be used to construct more benchmarks in the future.

Kamyar Mirzazad Barijough, Matin Hashemi, Volodymyr Khibin, Soheil Ghiasi, "**Implementation-Aware Model Analysis: The Case of Buffer-Throughput Tradeoff in Streaming Applications**", Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES), June 2015.

A) Benchmark Applications

The benchmark applications are beamformer, bitonicsort, dct, des, fft, matmul, mergesort, mpeg and serpent. Each benchmark has its own folder containing three files:

.str:

The benchmarks originally come from StreamIt benchmark suit [1]. StreamIt benchmark application is a textual program with .str extension that describes the application in form of a set of java-like functions, also known as filters.

In order to limit the graph size and hence reduce the time required to perform cycle-accurate simulations, we have decided to modify few constant parameters in some of the original StreamIt benchmarks. The changes are marked by the following comment in the benchmark .str files.

```
// Kamyar: original value was ...
```

.xml:

We process StreamIt textual file (.str file) and extract the corresponding SDF graph topology along with its actor execution times and channel rates. The result is converted into SDF3 graph format [2] which is an XML file representing the graph topology and its parameters. Details of this procedure is described in Section B of this document.

.pdf:

The visual representation of the constructed SDF graph.

B) Conversion Procedure

In order to convert StreamIt benchmark (.str file) to its corresponding SDF graph in SDF3 format, we first execute StreamIt compiler (RAW processor backend) with the following command.

`strc beamformer.str -raw 1` (see the blue box in the figure)

As a result, StreamIt compiler converts the .str file into a single .c file which is ready to be compiled by RAW processor's C compiler and subsequently executed on the RAW processor (one core of the RAW processor). We neither use the produced .c file nor intend to run the benchmark on a single processor platform, however, the StreamIt compiler generates a series of intermediate files which we use to extract the SDF graph.

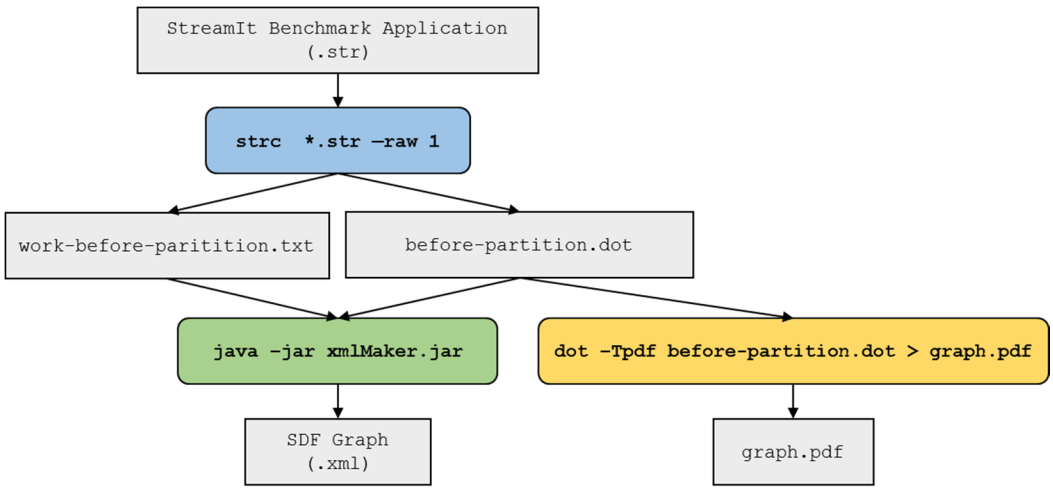
In specific we use the automatically generated "before-partition.dot" and "work-before-partition.txt" files. The first file contains graph topology and channel rates and the second one contains actor execution times. StreamIt estimates actor execution times based on cycle-per-instruction (CPI) counts of the RAW processor. To change the CPI estimates to match another processor, e.g., x86, StreamIt compiler source code needs to be modified, in specific, the "WorkConstants.java" file which contains the list of CPI counts. In fact, this is what we have done because we use Graphite simulator [3] which is based on x86 cores.

We have developed a simple text-processing java program xmlMaker.jar which extracts the information from "before-partition.dot" and "work-before-partition.txt" files and constructs the SDF graph in SDF3 XML format and tag the actors and channels (vertices and edges) with actor execution times and channel data rates.

`java -jar xmlMaker.jar` (see the green box in the figure)

Visual representation of the constructed SDF graph can be obtained by the following command:

`dot -Tpdf before-partition.dot > graph.pdf` (see the yellow box in the figure)



References

[1] http://groups.csail.mit.edu/cag/streamit
[2] http://www.es.ele.tue.nl/sdf3
[3] http://groups.csail.mit.edu/carbon/?page_id=111