Dynamo: A Runtime Codesign Environment

Heather Quinn[1], Miriam Leeser
Dept of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115
{hquinn, mel}@ece.neu.edu

L. A. Smith King
Dept. of Mathematics and Computer Science
College of the Holy Cross
Worcester, MA 01610
LA@cs.holycross.edu

Systems using Field Programmable Gate Array (FPGA) boards have been proven effective for gaining one to three orders of magnitudes speedup over systems based solely on PCs. Signal and image processing applications are especially attractive for implementation on FPGAs as their computationally intensive and massively parallel algorithms can effectively take advantage of the FPGA architecture. In moving part of an algorithm to hardware, one must consider overhead costs as well as the improvement in the computation to determine whether the move will result in overall system speedup. Dynamo is a runtime system for generating hardware/software pipeline implementations. Dynamo balances the benefits of hardware and software implementations and takes overhead costs into account in order to accurately predict runtimes of hardware/software systems.

Currently, the Dynamo system generates hardware/software solutions for image processing applications from a library of predefined component implementations. Many image processing applicationsconsist of a series of algorithms applied to the image, forming a computation *pipeline*. When using Dynamo, an image analyst only needs to specify the pipeline of image processing components to apply and an image or images to process. Image processing components are chosen from a library of predefined components: the *image processing Basic Library of Components* (ipBLOC). Each component in the library has at least one software and one hardware implementation associated with it. From the pipeline specification, Dynamo selects the most efficient combination of of hardware or software component implementations to minimize pipeline runtime (*pipeline assignment*), generates the source code for a HW/SW implementation of the pipeline (*pipeline compilation*), processes the input image(s) using the generated pipeline (*pipeline execution*), and returns the result to the analyst.

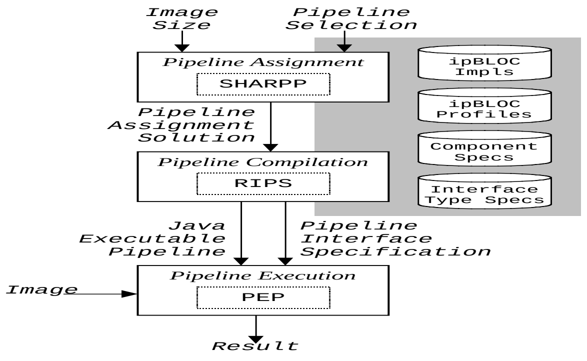

Figure 1: System Overview

The goals of the Dynamo system are to allow an image analyst to focus solely on pipeline selection, create image processing pipelines at runtime, make efficient use of software and FPGA hardware, and build pipelines that maximize performance. Currently, Dynamo is restricted to image processing applications, but is easily extensible to other application domains. Dynamo's design includes three major subsystems which implement pipeline assignment, pipeline compilation and pipeline execution, respectively. These subsystems

[1]Heather Quinn is supported under a Graduate Student Researchers Program Fellowship through NASA's Goddard Space Flight Center.

interact with Dynamo's predefined libraries, including the ipBLOC. Figure 1 shows an overview of the Dynamo system.

At the core of the Dynamo runtime system is *pipeline assignment* (PA). PA assigns hardware or software implementations to each component in the pipeline to minimize total pipeline execution time, while meeting the area requirements for the FPGA device. There are costs associated with passing intermediate images and data between components that differ depending on component types and interfaces. These costs include data movement, communication, reprogramming and hardware initialization. PA must balance the latency and area costs to find the best solution for a given FPGA device. Profiles of the component implementations and of overhead runtimes allow PA to accurately predict pipeline solution runtimes. The pipeline assignment problem is NP-complete, and is similar to the hardware/software codesign partitioning problem.

The Dynamo subsystem responsible for solving PA is named SHARPP (Software/HArdware Runtime Procedural Partitioning). SHARPP solves the pipeline assignment problem quickly at runtime and uses different algorithms for different pipeline sizes to keep execution time short. SHARPP uses dynamic programming for small pipelines (1-7 components) and two variations of tabu search to solve large pipelines (7-20 components). Tabu search is a metahueristic based on local search which provides near-optimal PA solutions. The two variations find solutions that are on average 18% slower than the optimal pipeline latency.

Pipeline compilation and execution occur within the runtime environment. Dynamo's RIPS (Runtime Interfacing for Pipeline Synthesis) subsystem performs *pipeline compilation*. RIPS uses the SHARPP output to compose an executable that calls the appropriate implementations and overhead methods that comprise a pipeline solution. The pipelines are constructed at runtime because pre-constructing PA solutions for all possible combinations of component implementations is not practical. Finally, the pipeline is executed. Dynamo's execution system runs the compiled pipeline on the selected image and outputs the results.

As an illustration of how Dynamo works, assume the image analyst has chosen to run the pipeline "median filter → histogram" on a 40185-pixel image. Table 2 shows the four possible solutions to this pipeline and the SHARPP predicted latencies. The component assignments are shown within parentheses with ":sw" indicating software and ":hw" indicating hardware. These results show that the optimal solution for this image size uses the hardware implementations for both components. Figure 2 shows what the pipeline looks like when the overhead costs are included in the pipeline, where the squares represent the components and the circles represent the overhead methods. RIPS was used to build executable solutions for each of the four pipelines. These pipelines were executed and measured so the actual execution times could be compared with SHARPP's estimates. The relative error for the predicted latency is quite low.

We have tested pipelines that range in size from one to 20 components. Many short pipelines are completely assigned to software since the hardware initialization cost is relatively high. As pipelines included more components, enough speedup will be gained by processing in hardware to mitigate the device initialization cost. Therefore, longer pipelines tend to be mainly assigned to hardware, except when either the image is small or the component is not well matched to the FPGA architecture.

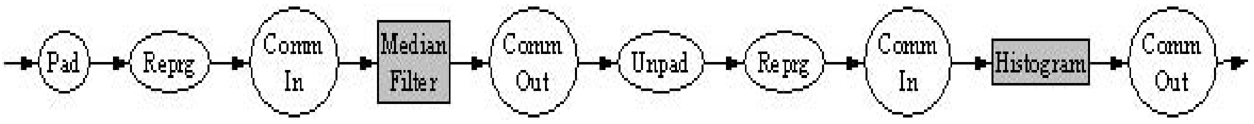

Figure 2: Pipeline Solution with Overhead Methods

Annotated Solution	Predicted Runtime (ms)	Actual Runtime (ms)	Relative Error
(mf:sw) → (histogram:sw)	3766	3801	0.009
(mf:sw) → (histogram:hw)	6076	5844	-0.038
(mf:hw) → (histogram:sw)	2772	2836	0.023
(mf:hw) → (histogram:hw)	2718	2260	-0.169

Table 2: The Four Solutions to median filter → histogram