A Domain-specific Approach for Software Development on Manycore Platforms

Jerker Bengtsson and Bertil Svensson
{Jerker.Bengtsson, Bertil.Svensson}@hh.se

MCC-08 Ronneby, November 27-28, 2008

Centre for Research on Embedded Systems
What Domain?

Ref. H. Sahlin, "Introduction and overview of LTE Baseband Algorithms", Baseband research group, Ericsson AB

- High-performance signal processing (e.g. Radio Basestations or Radar Systems)
  - time constrained processing of infinite streams of data
  - large amounts of heterogeneous parallelism

Problem: How to describe and best map time constrained computation graphs on abstract manycore targets?
What Domain?

Ref. H. Sahlin, "Introduction and overview of LTE Baseband Algorithms", Baseband research group, Ericsson AB

- High-performance signal processing (e.g. Radio Basestations or Radar Systems)
  - time constrained processing of infinite streams of data
  - large amounts of heterogeneous parallelism
- **Problem**: How to describe and best map **time constrained** computation graphs on **abstract** manycore targets?
We argue that *domain-specific* parallel models of computation (MoC) are needed to

- be able to develop efficient parallelization and mapping tools
- enable portability of tools and design methodologies
- raise the parallel programming abstraction level

Dataflow MoC's are *naturally parallel* and provides an excellent match to

- signal processing applications
- manycore targets
We argue that *domain-specific* parallel models of computation (MoC) are needed to
- be able to develop efficient parallelization and mapping tools
- enable portability of tools and design methodologies
- raise the parallel programming abstraction level

Dataflow MoC’s are *naturally parallel* and provides an excellent match to
- signal processing applications
- manycore targets
We are building an iterative mapping and code generation tool. The idea is to iteratively tune parallel mapping by user-specified constraints in the application model and feed back dynamic performance measurements. We use the Ptolemy modeling environment (UC Berkeley).
We describe an application using Synchronous Dataflow (SDF).

With each actor we associate a tuple \(< r_p, r_m, R_r, R_s >\) where

- \(r_p\) is the worst case computation time, in number of operations.
- \(r_m\) is the requirement on data allocation, in words.
- \(R_s = \{ r_{s1}, r_{s2}, \ldots, r_{sn} \}\) and \(r_{si}\) is the number of words produced on channel \(i\) each firing.
- \(R_r = \{ r_{r1}, r_{r2}, \ldots, r_{rm} \}\) and \(r_{rj}\) is the number of words consumed on channel \(j\) each firing.
The Application Model

CERES

- We describe an application using Synchronous Dataflow (SDF)
- With each actor we associate a tuple $< r_p, r_m, R_r, R_s >$ where
  - $r_p$ is the worst case computation time, in number of operations.
  - $r_m$ is the requirement on data allocation, in words.
  - $R_s = \{ r_{s_1}, r_{s_2}, \ldots, r_{s_n} \}$ and $r_{s_i}$ is the number of words produced on channel $i$ each firing.
  - $R_r = \{ r_{r_1}, r_{r_2}, \ldots, r_{r_m} \}$ and $r_{r_j}$ is the number of words consumed on channel $j$ each firing.
What manycore targets?

- We are interested in manycores with:
  - core individual instruction sequencing
  - tightly coupled interconnectivity
  - distributed on-chip memory (core private)
  - software controlled caching

- We have derived a parallel machine model for such targets
A machine is described by two tuples $M$ and $F$

The computational resources are described by

$$M = < (x, y), p, b_g, g_w, g_r, o, s_o, s_l, n_b, c, h_l, r_l, r_o >$$

which are parameters

The computational performance is described by

$$F(M) = < t_p, t_s, t_r, t_c, t_{gw}, t_{gr} >$$

which are functions of $M$ determining the cost for

- process the fire code of an actor ($t_p$)
- core send and receive ($t_s, t_r$)
- core to core propagation time ($t_c$)
- reading and writing to global memory ($t_{gw}, t_{qr}$)
A machine is described by two tuples $M$ and $F$

- The computational resources are described by

$$M = < (x, y), p, b_g, g_w, g_r, o, s_o, s_l, n_b, c, h_l, r_l, r_o >$$

which are parameters

- The computational performance is described by

$$F(M) = < t_p, t_s, t_r, t_c, t_{gw}, t_{qr} >$$

which are functions of $M$ determining the cost for

- process the fire code of an actor ($t_p$)
- core send and receive ($t_s, t_r$)
- core to core propagation time ($t_c$)
- reading and writing to global memory ($t_{gw}, t_{qr}$)
To model a machine we first set parameters of $M$

Then we define how to evaluate the functions in $F$

- $t_p(r_p, p) = \left\lceil \frac{r_p}{p} \right\rceil$
- $t_s(R_s, o, s_o) = \left\lceil \frac{R_s}{\text{framesize}} \right\rceil \times o + R_s \times s_o$
- $t_r(R_r, o, r_o) = \left\lceil \frac{R_r}{\text{framesize}} \right\rceil \times o + R_r \times r_o$
- $t_c(R_s, d, s_l, c, h_l, r_l) = s_l + d \times h_l + \left\lceil (R_s - 1) \times \frac{1}{c} \right\rceil + r_l$
- $t_{gw}(R_s, d, s_l, c, h_l, b_g, g_w) = ...$
- $t_{gr}(R_s, d, s_l, c, h_l, b_g, g_r, r_l) = ...$

With this approach we can tune $F$ in detail for a certain target to obtain higher accuracy
To model a machine we first set parameters of $M$

Then we define how to evaluate the functions in $F$

- $t_p(r_p, p) = \left\lceil \frac{r_p}{p} \right\rceil$
- $t_s(R_s, o, s_o) = \left\lceil \frac{R_s}{\text{framesize}} \right\rceil \times o + R_s \times s_o$
- $t_r(R_r, o, r_o) = \left\lceil \frac{R_r}{\text{framesize}} \right\rceil \times o + R_r \times r_o$
- $t_c(R_s, d, s_l, c, h_l, r_l) = s_l + d \times h_l + \left\lceil (R_s - 1) \times \frac{1}{c} \right\rceil + r_l$
- $t_{gw}(R_s, d, s_l, c, h_l, b_g, g_w) = ...$
- $t_{gr}(R_s, d, s_l, c, h_l, b_g, g_r, r_l) = ...$

With this approach we can tune $F$ in detail for a certain target to obtain higher accuracy
To model a machine we first set parameters of $M$

Then we define how to evaluate the functions in $F$

- $t_p(r_p, p) = \left\lfloor \frac{r_p}{p} \right\rfloor$
- $t_s(R_s, o, s_o) = \left\lfloor \frac{R_s}{\text{framesize}} \right\rfloor \times o + R_s \times s_o$
- $t_r(R_r, o, r_o) = \left\lfloor \frac{R_r}{\text{framesize}} \right\rfloor \times o + R_r \times r_o$
- $t_c(R_s, d, s_l, c, h_l, r_l) = s_l + d \times h_l + \left( (R_s - 1) \times \frac{1}{c} \right) + r_l$
- $t_{gw}(R_s, d, s_l, c, h_l, b_g, g_w) = ...$
- $t_{gr}(R_s, d, s_l, c, h_l, b_g, g_r, r_l) = ...$

With this approach we can tune $F$ in detail for a certain target to obtain higher accuracy
The Configuration Graph is a dataflow process network (PN)

- The PN model is annotated with the functions of $F$
  - edges are weighted with one of $t_c$, $t_{gw}$, $t_{gr}$
  - vertices has a list of operations $t_p$, $t_s$, $t_r$

The usage of the IR is two-fold, we can:
- use it to generate code for cores and the network
- do abstract interpretation to evaluate performance
The Configuration Graph is a dataflow process network (PN)

- The PN model is annotated with the functions of $F$
  - edges are weighted with one of $t_c$, $t_{gw}$, $t_{gr}$
  - vertices has a list of operations $t_p$, $t_s$, $t_r$

The usage of the IR is two-fold, we can:
- use it to generate code for cores and the network
- do abstract interpretation to evaluate performance
The Configuration Graph is a dataflow process network (PN).

- The PN model is annotated with the functions of $F$:
  - Edges are weighted with one of $t_c$, $t_{gw}$, $t_{gr}$.
  - Vertices have a list of operations $t_p$, $t_s$, $t_r$.

- The usage of the IR is two-fold, we can:
  - Use it to generate code for cores and the network.
  - Do abstract interpretation to evaluate performance.
Contributions in the paper

We have

- proposed a fine-grained machine model
- developed a multipurpose intermediate representation
- shown how to do analysis on the IR using abstract interpretation

What we describe in the paper is implemented in Ptolemy
Contributions in the paper

- We have
  - proposed a fine-grained machine model
  - developed a multipurpose intermediate representation
  - shown how to do analysis on the IR using abstract interpretation

- What we describe in the paper is implemented in Ptolemy
We aim for a tool chain offering flexible mapping strategies. The goal is to evaluate and use non-functional properties for iterative tuning of parallel mappings.

To explore how efficiently this can be done, we further need to:

- model and compare with concrete target (e.g., RAW)
- calibrate the model so that IR calculations closely match dynamic measurements on target hardware

To investigate practical usefulness, we plan to:

- model an application (e.g., LTE uplink) and study the consequences of parallel mapping choices
We aim for a tool chain offering flexible mapping strategies. The goal is to evaluate and use non-functional properties for iterative tuning of parallel mappings.

To explore how efficiently this can be done, we further need to:

- model and compare with concrete target (e.g., RAW)
- calibrate the model so that IR calculations closely match dynamic measurements on target hardware

To investigate practical usefulness, we plan to:

- model an application (e.g., LTE uplink) and study the consequences of parallel mapping choices
Thank you!

e-mail: Jerker.Bengtsson@hh.se