Multiscale Dataflow Computing

Uvod

Motiv

The programming language is not simply a tool with which a preconceived task or function can be accomplished; it is an extensive basis of structure with which the imagination can interact. –

John Chowning

Nova paradigma – snabdevanje podacima

Data Flow

- Data flow: actual flow of data values among instructions that produce results and those that consume them
  - branches make flow dynamic, determine which instruction is supplier of data
- Example:

```
DADDU    R1,R2,R3
BEQZ     R4,L
DSUBU    R1,R5,R6
L:  …
OR       R7,R1,R8
```

- OR depends on DADDU or DSUBU?
  Must preserve data flow on execution

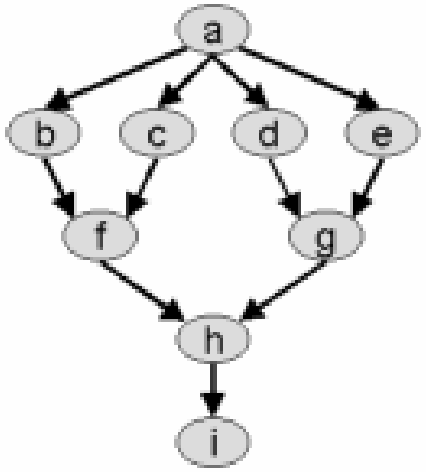
DDG graf (data dependency graph)

Instruction Sequences

B3

```
(a)  t1 := ld(x);
(b)  t2 := t1 + 4;
(c)  t3 := t1 * 8;
(d)  t4 := t1 - 4;
(e)  t5 := t1 / 2;
(f)  t6 := t2 * t3;
(g)  t7 := t4 - t5;
(h)  t8 := t6 * t7;
(i)  st(y,t8);
```

Instr. Sequence 1



B3'

```
(a)  t1 := ld(x);
(d)  t4 := t1 - 4;
(e)  t5 := t1 / 2;
(g)  t7 := t4 - t5;
(b)  t2 := t1 + 4;
(c)  t3 := t1 * 8;
(f)  t6 := t2 * t3;
(h)  t8 := t6 * t7;
(i)  st(y,t8);
```

Instr. Sequence 2

B3''

```
(a)  t1 := ld(x);
(b)  t2 := t1 + 4;
(c)  t3 := t1 * 8;
(f)  t6 := t2 * t3;
(d)  t4 := t1 - 4;
(e)  t5 := t1 / 2;
(g)  t7 := t4 - t5;
(h)  t8 := t6 * t7;
(i)  st(y,t8);
```

Instr. Sequence 3

DFE mašina i programski jezici

- Programski jezici za DFE (data flow) mašine treba da budu prilagođeni za lako generisanje grafa zavisnosti po podacima i izbegavanje svih slučajeva kada u vreme prevođenja ne može da se definiše da li ima ili nema zavisnosti po podacima.
- Osim toga, treba da se eliminišu sve antizavisnosti i izlazne zavisnosti, da one ne bi usporavale izvršavanje operacija.
- Postoji više načina da se to postigne. Osnovno je pravilo da se samo jednom može dodeljivati vrednost promenljivoj (single-assignment rule).

Statičke data flow mašine

Na osnovu te ideje, formirane su prvo statičke DataFlow mašine.

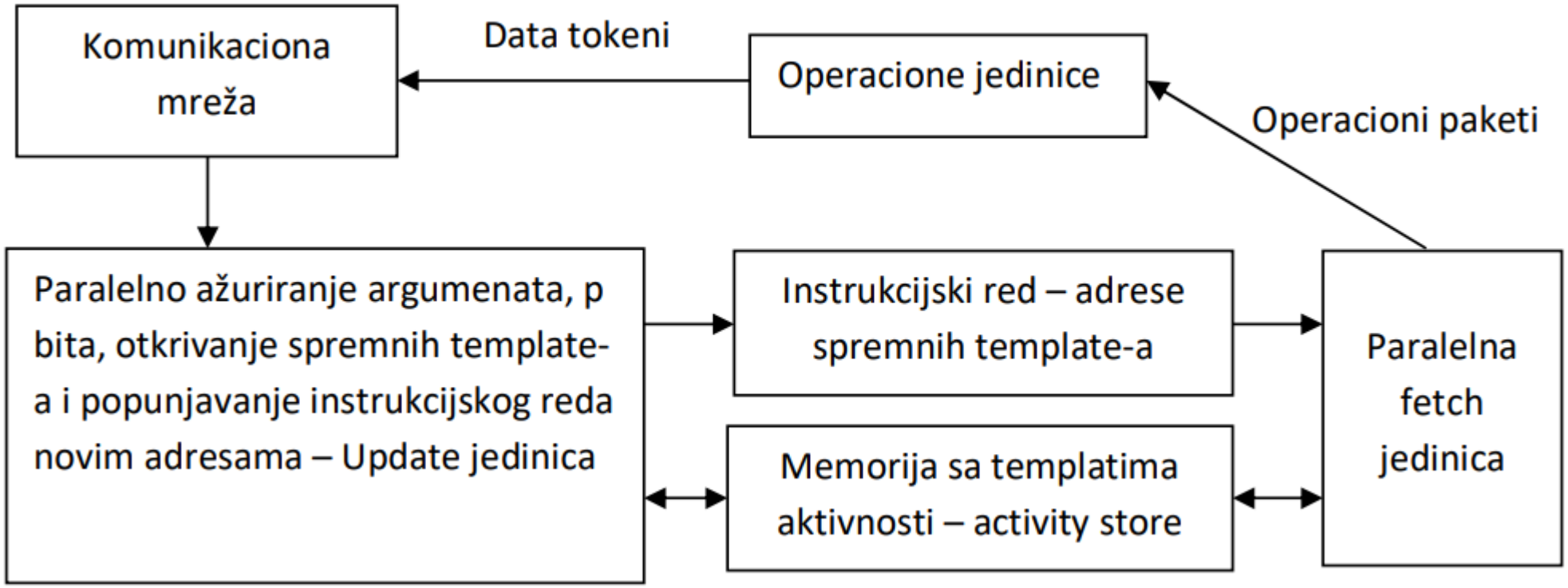
Prvo se postavilo pitanje kako da se graf zavisnosti po podacima nekako pretoči u torke koje će opisivati pojedine operacije. Dakle, već za realizaciju kôda bazičnog bloka je neophodno da se napravi kompleksna instrukcija koja ima jedan ključan problem: u njenom kodu treba nekako da postoji referenciranje svih operacija koje su zavisne po podacima od nje.

To predstavlja problem zato što je, u opštem slučaju, broj operacija koje koriste rezultat neke operacije promenljiv.

Kako sve operacije nisu komutativne, neophodno je i definisati ulaz sledeće operacije (ulaz aritmetičko logičke jedinice) na koji dolazi rezultat prethodne operacije od koje je operacija zavisna. To je jedan od razloga zašto se moraju formirati mali paketi-torke za podatke.

Različite torke se moraju formirati za prosleđivanje istog rezultata operacija do svih drugih zavisnih operacija. **Ti mali paketi se nazivaju data tokeni, a u osnovi predstavlja torku sa svim neophodnim dodatnim informacijama koje prate podatak.**

Statička data flow mašina



Statička data flow mašina - mane

- Osnovni nedostatak statičkih DataFlow mašina potiče od činjenice da su podaci vezani statički za operaciju, pa problem predstavljaju delovi koda koji se ponavljaju. To su pre svega petlje i procedure koje se pozivaju. Problem su npr. DoAll petlje.
- Pretpostavimo zbog jednostavnosti da Doall petlja nema zavisnosti preko granica iteracija. Tada su slobodni skupovi operacija za sve iteracije odjednom data ready, ako se posmatra samo petlja. Kako su mašine podacima pokretane, u delovima koda pre petlji, ti spremni argumenti za iteracije petlji se izračunavaju i iteracije petlje mogu da krenu čim su raspoloživi. Dakle, za istu statičku operaciju iz petlje se može pojaviti poplava argumenata iz različitih iteracija – različitog konteksta.
- Pritom, statička DataFlow mašina nema mogućnost da uparuje levi i desni argument operacije, tako da moraju da budu iz iste iteracije.
- Na osnovu pokretanja podacima, bez restrikcija, moglo bi da se dobije mešanje argumenata za operacije iz razlicitih iteracija i samim tim nekorektno izvršavanje.

Nova paradigma – alternativa paralelnom izračunavanju

- Maxeler Technologies developed an alternative paradigm to parallel computing: Multiscale Dataflow Computing.
- Dataflow computing was popularized by a number of researchers in the 1980's, especially J. B. Dennis. In the dataflow approach an application is considered as a dataflow graph of the executable actions; as soon as the operands for an action are valid, the action is executed and the result is forwarded to the next action in the graph.
- There are no load or store instructions as the operational node contains the relevant data.
- Creating a generalized interconnection among the action nodes proved to be a significant limitation to dataflow realizations in the 1980's.
- Over recent years the extraordinary improvement in transistor array density allowed emulations of the application dataflow graph.
- The Maxeler dataflow implementations are a generalization of the earlier work employing static, synchronous dataflow with an emphasis on data streaming. Indeed "multiscale" dataflow incorporates vector and array processing to offer a multifaceted parallel compute platform

Optimized dataflow programs

At the heart of Multiscale Dataflow Computing is the programming environment, described in this tutorial. While all this is loosely termed the Maxeler compiler the work is much more than a high level translator. Embedded in it is the approach to writing optimized dataflow programs. There are at least three different optimization processes involved.

1. The application actions are written in a dataflow graph type form, unrolling loops, specifying actions processing a data stream.
2. Next, the data flow from memory must be described so that it can be properly scheduled into the dataflow engine.
3. Finally, multiple dataflow engines can be configured together in various ways for maximum application acceleration. All this is done using familiar programming vernacular such as Java type vocabulary.

The essence of the Maxeler programming approach is high performance with high productivity on the part of the programmer.

Predlog ubrzanja: Maxeler

- Reprogramabilne max kartice
- Mogu značajno da ubrzaju algoritam
- Ideja: pronaći paralelne poslove
- Implementirati poslove u max kernelima

MISANU machine

- You can access physical MISANU machine via ssh:

ssh -p 279 -o PubkeyAuthentication=no <username>@maxeler.mi.sanu.ac.rs

Data flow model of computation

- In a dataflow program, we describe the operations and data choreography for a particular algorithm (see Figure 2).

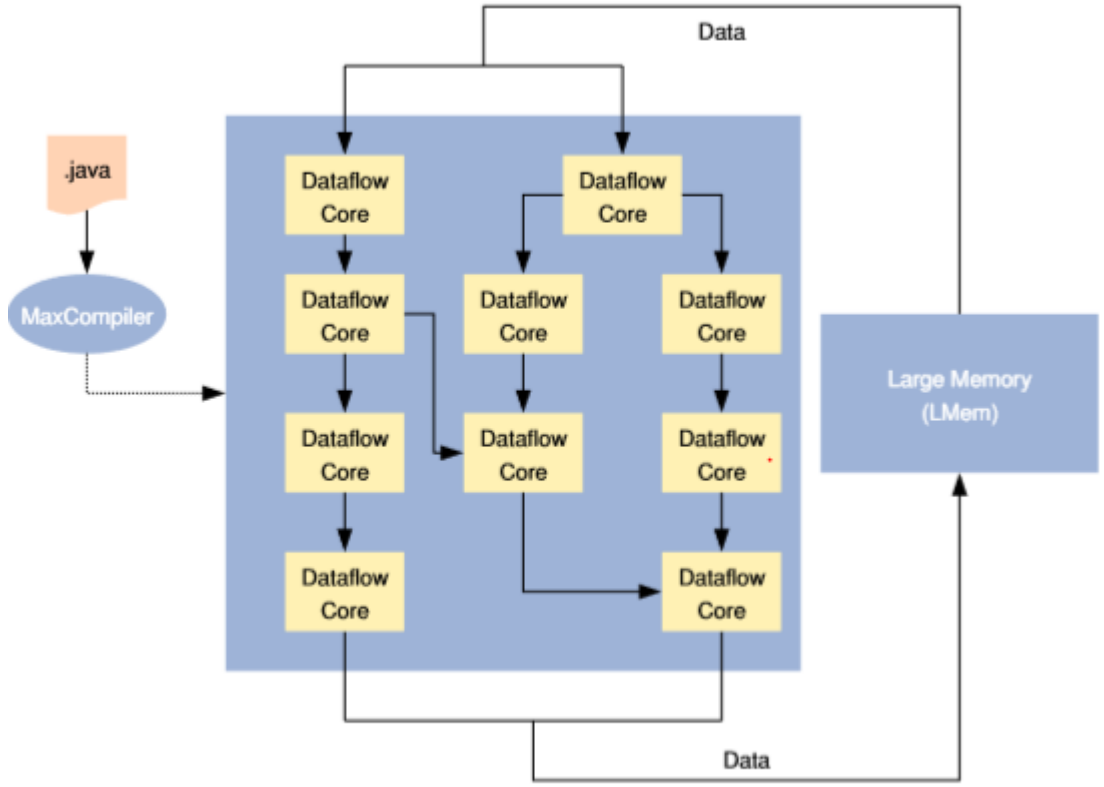


Figure 2: A dataflow program in action.

Data flow model of computation

- In a Dataflow Engine (DFE), data streams from memory into the processing chip where data is forwarded directly from one arithmetic unit (’dataflow core’) to another until the chain of processing is complete. Once a dataflow program has processed its streams of data, the dataflow engine can be reconfigured for a new application in less than a second.

Data flow model of computation – dataflow core

- Each dataflow core computes only a single type of arithmetic operation (for example an addition or multiplication) and is thus simple so thousands can fit on one dataflow engine.
- In a DFE processing pipeline every dataflow core computes simultaneously on neighboring data items in a stream.
- Unlike control flow cores where operations are computed at different points in time on the same functional units (”computing in time”), a dataflow computation is laid out spatially on the chip (”computing in space”).
- **Dependencies in a dataflow program are resolved statically at compile time.**

Data flow engine (DFE) structure

- The data flow engine structure itself represents the computation thus there is no need for instructions per se; instructions are replaced by arithmetic units laid out in space and connected for a particular data processing task.
- Because there are no instructions there is no need for instruction decode logic, instruction caches, branch prediction, or dynamic out-of-order scheduling.
- By eliminating the dynamic control flow overhead, the full resources of the chip are dedicated to performing computation. At a system level, the dataflow engine handles computation of large scale data processing while CPUs running Linux manage irregular and infrequent operations, IO and inter-node communication.