Executable Mathematics

Jan Kuper

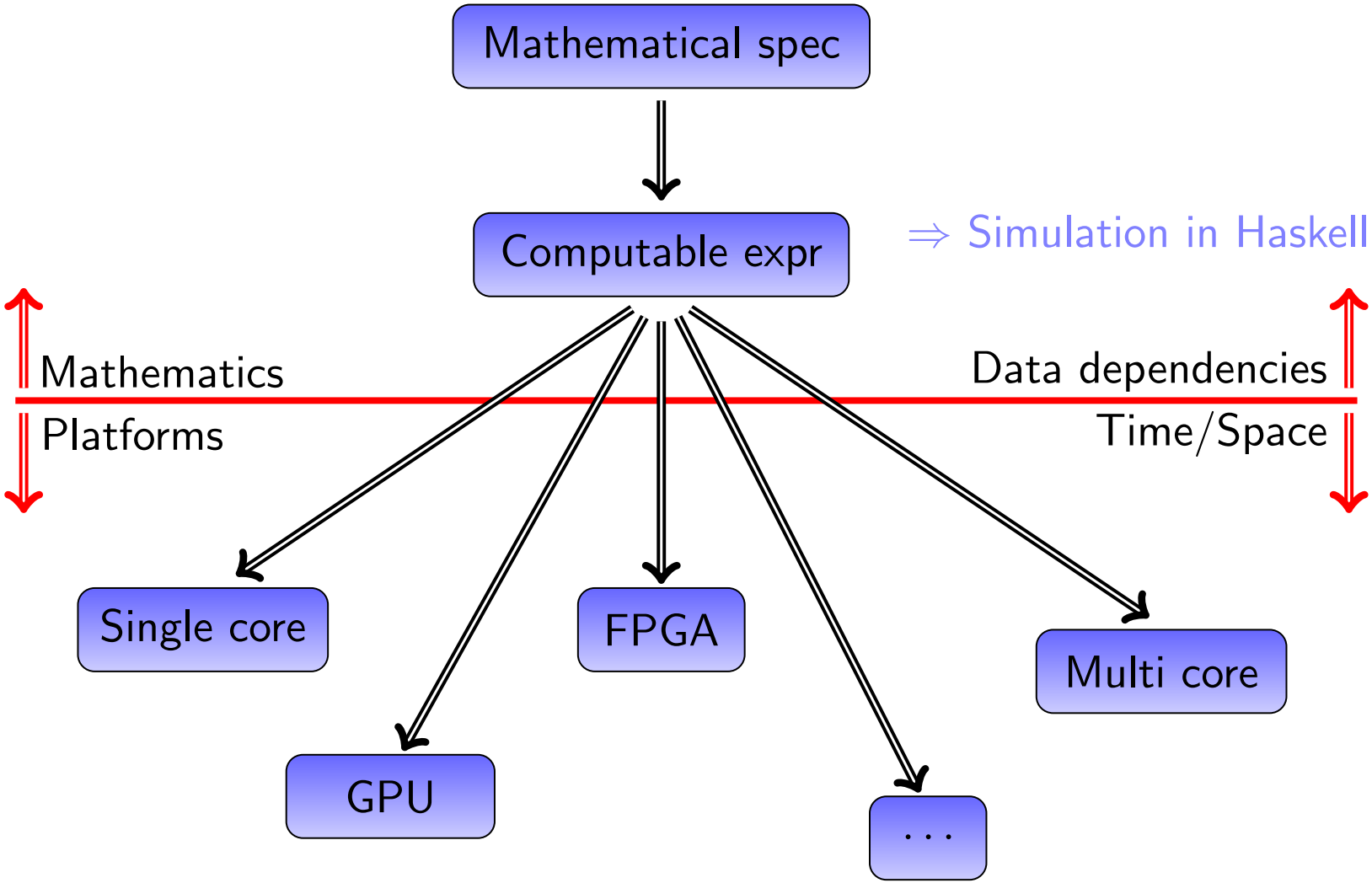
Embedded Systems Group
University of Twente

GPU-NIRICT
Utrecht

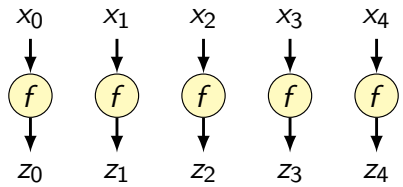
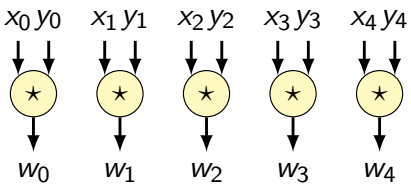
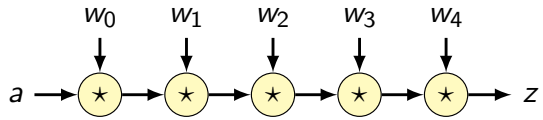
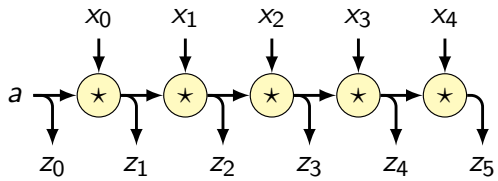
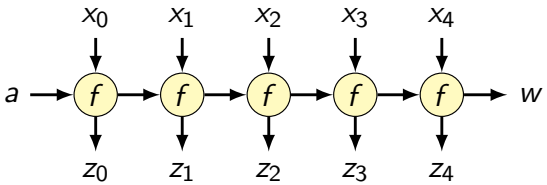
December 3, 2015

Key aspects

- Programming model, one programming environment for various platforms
- Hardware/Software codesign
- Stay within mathematical realm as long as possible (data dependencies)
- Not starting from imperative reference implementation
- Specification, simulation, implementation in one language (Haskell $\approx$ "mathematics in typewriter font")
- ...


Mathematical spec
Computable expr
⇒ Simulation in Haskell
Mathematics
Platforms
Data dependencies
Time/Space
Single core
FPGA
Multi core
GPU
...

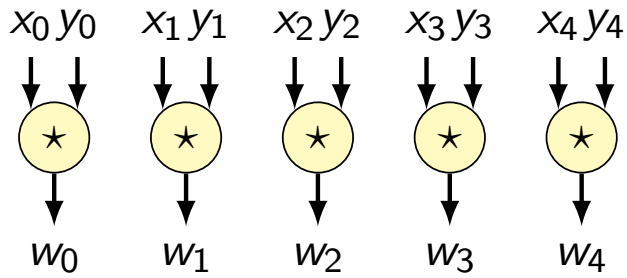
Higher order functions

map		$f\ x \Rightarrow z$	$zs = map\ f\ xs$
zipWith		$x\ \star\ y \Rightarrow z$	$zs = zipWith\ (\star)\ xs\ ys$
foldl		$a\ \star\ x \Rightarrow a'$	$w = foldl\ (\star)\ a\ xs$
scanl		$a\ \star\ x \Rightarrow a'$ $z = a$	$zs = scanl\ (\star)\ a\ xs$
mapAccumL		$f\ a\ x \Rightarrow (a', z)$	$(w, zs) = mapAccumL\ f\ a\ xs$

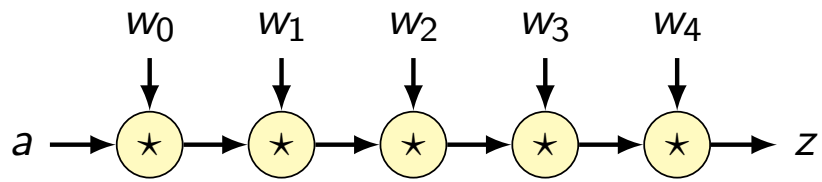
Dot product

$$\vec{x} \bullet \vec{y} \;=\; \sum_{i=0}^{n-1} x_i y_i \;=\; x_0 y_0 + x_1 y_1 + \cdots + x_{n-1} y_{n-1}$$

$ws = \; zipWith\ (*)\ xs\ ys$



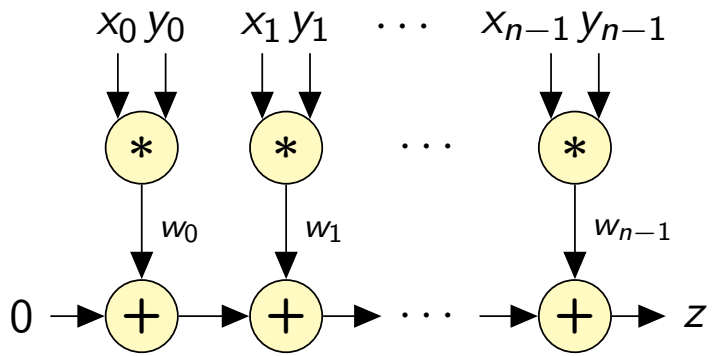
$z = \; foldl\ (+)\ 0\ ws$



$z = foldl\ (+)\ 0\ ws$

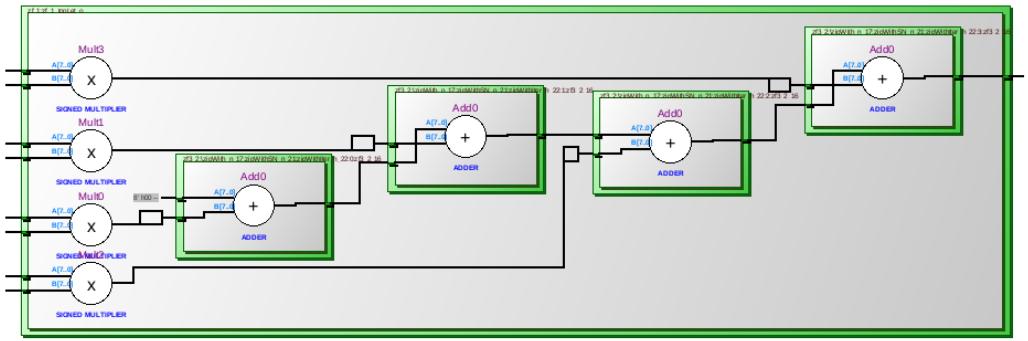
where

$ws = zipWith\ (*)\ xs\ ys$



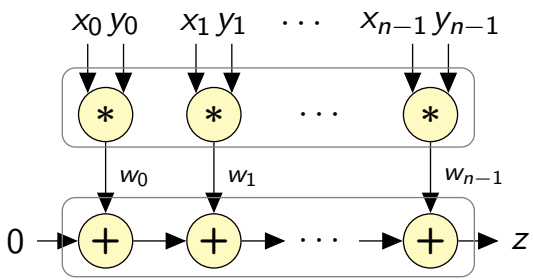
Generated Hardware + Imperative Code

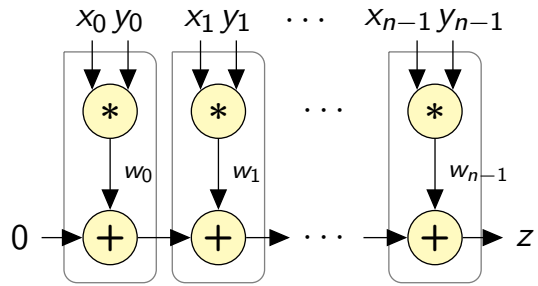
foldl $(+)$ 0 (*zipWith* $(*)$ *xs ys*)




```
for (i0...)  {
 v0[i0] = xs[i0] * ys[i0];
};
Z = 0;
for (i1...)  {
 Z = Z + v0[i1];
};
```

Transformations





$$foldl\ f\ \ a\ (zipWith\ g\ \ xs\ \ ys)$$

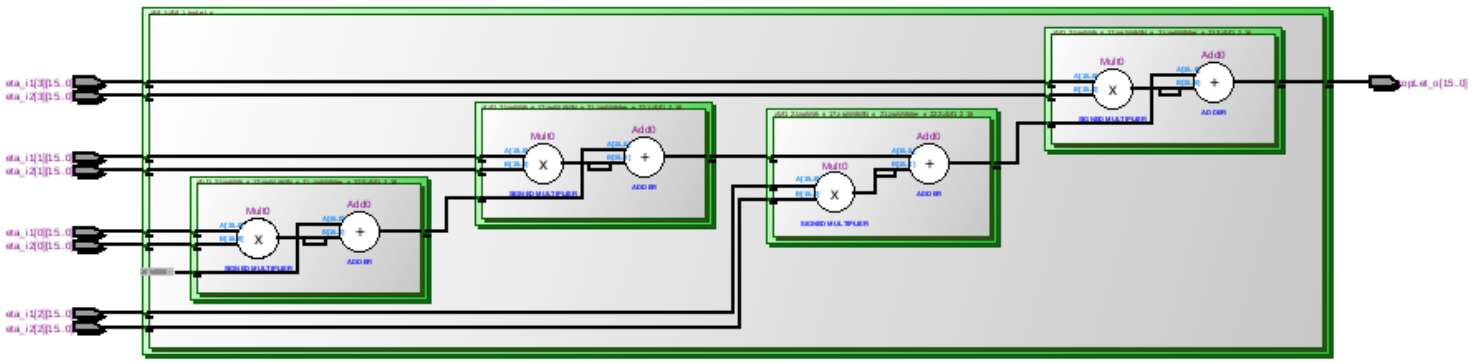
$$\Rightarrow$$

$$foldl\ (f \triangleleft g)\ (zip\ xs\ ys)$$

$$f \triangleleft g\ \ =\ \ \lambda a\ z \rightarrow f\ \ a\ (g\ z)$$

Generated Hardware + Imperative Code

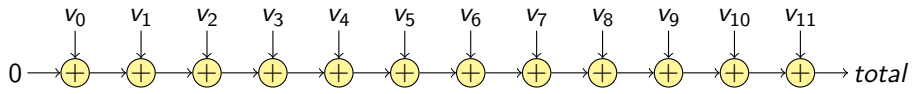
$$foldl\ \ ((+) \triangleleft (*))\ (zip\ xs\ ys)$$



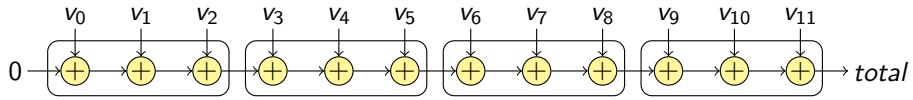
```
Z = 0;
for (i0...)  {
 Z = Z + x * y;
};
```

Some other transformations

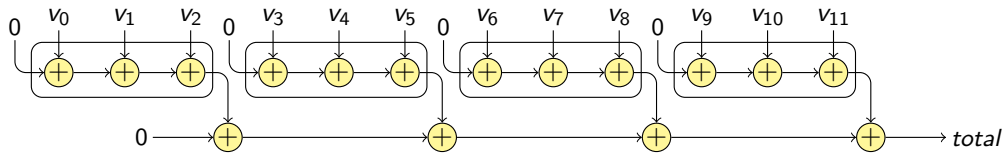
foldl (+) 0 *xs*



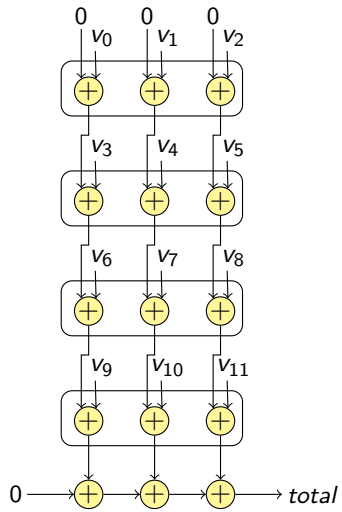
foldl (*foldl* (+)) 0 *xss*



foldl (+) 0 (*map* (*foldl* (+) 0) *xss*)



foldl (+) 0 (*foldl* (*zipWith* (+) 0*s*) *xss*)



Future work

- Extend code generation towards OpenCL, and also towards specific architectures (Xentium, WaveCore, ,. . .)
- Develop more transformations
- Generation of data-dependency graphs, combine with mapping and dataflow analysis for performance
- Case studies: HPC, image processing, signal processing, adaptive cruise control, particle filtering, solving differential equations
- . . .

Thanks