The harsh reality of commercial functional programming

presented at the FP-dag on Jan 9, 2009

Paul Stravers
Vector Fabrics

paul@vectorfabrics.com

FP world domination?

Good idea !

Let's start a company...

...to create a tool that simplifies embedded system design.

1. FP is a Good Thing
2. Create a Haskell-to-FPGA compiler
3. ...
4. Profit?

This does not work!

We need to make a living, investors want ROI

Think again: 

1. FP is a Good Thing
2. Use FP to create a C-to-FPGA compiler
3. ...
4. Profit!

But wait ...

Money is not all that a company is about ...

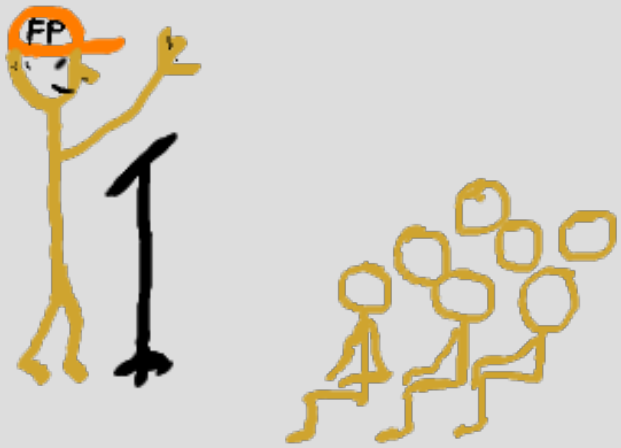
Don't you have a community responsibility?

What do you contribute to the improvement of the human condition?

By setting an example...

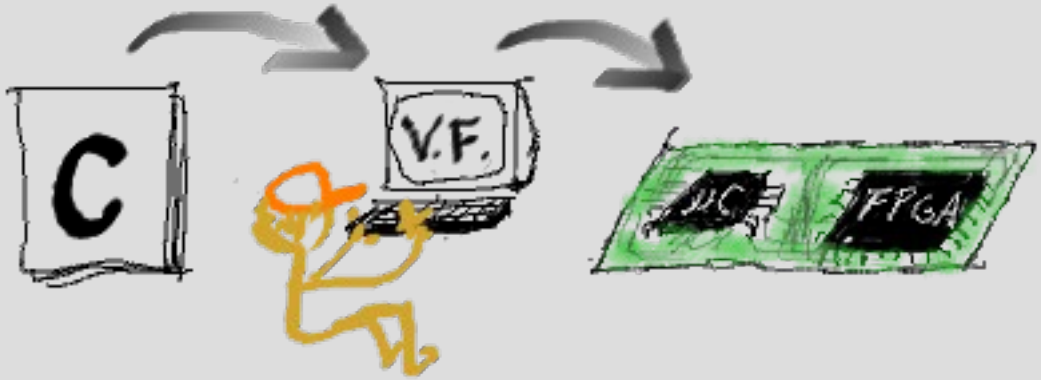
Show the world how to be successful with FP

...and the example will be copied.



About Vector Fabrics

- Founded February 2007
- Makes an embedded systems compiler



- Employs 11 people: programming, marketing
- Funded with venture capital from USA, NL

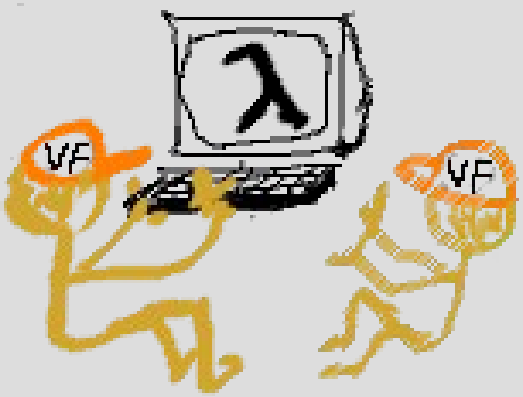
Embedded systems compiler

- program analysis
- concurrency extraction
- multi-core system architectures
- hardware/software assignment
- hardware synthesis and optimization
- multi-threaded run-time system

- ... etc. Hard problems!

How we work

- SVN revision control
- Jira bug and issue tracker linked to SVN
- Hudson continuous integration
  - nightly builds, regression
  - on multiple platforms
  - linked to SVN
- standup meeting every morning
- pair programming
- XP project planning and progress monitoring



Be different!

A startup does not have the burden of legacy
...so we can adopt the best technology

We innovate with our product, and ...
... by deploying the best development tools

How?
We use a modern functional language

Programming in Caml

- Only the functional subset
- Only immutable data structures
- Extensive use of higher order modules
- No objects

Why not Haskell?

- predictable run-time is key: strict evaluation

- escape to mutable data structures

But:

- functional subset is not as elegant and powerful as Haskell

Experience with FP

- programmers quickly become productive
- compiler enforced code robustness
  - static type checking
  - pattern matching
  - immutable data structures
  - pure functions
  - garbage collection
- good performance, no need for hacking
- foreign function interface works well

Experience with FP

- Caml debugger has bugs, no one to fix 'm
- Type inference error messages can be confusing
- Hardest bugs to find: using stale values

```
let node = get_node name graph in
let m = NodeMap.add name node m in
…
let blue_node = change_color Blue node in
let graph = update_node name blue_node graph in
…
let the_node = NodeMap.find name m in
(* at this point the_node is not present in graph *)
```

Conclusions

- FP works well in a commercial setting
- Immediate benefits, expressed in $$
- Tooling can be improved
- Functional programming is FUN !