BAE SYSTEMS

Redesigning the Computer for Security

Using Haskell EDSLs to Bootstrap a New Computing Platform

DARPA CRASH SAFE
BAE Systems, University of Pennsylvania, Harvard University, Northeastern University

Tom Hawkins
tom.hawkins@baesystems.com
22 Sept 2013

The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Approved for Public Release, Distribution Unlimited.
Cleared for Open Publication on 27 August 2013.

BAE SYSTEMS

State of Computer Security

- How secure is our critical infrastructure?

```
Welcome to wellhead7.pipelines-and-things.com
username: admin
password: admin
```

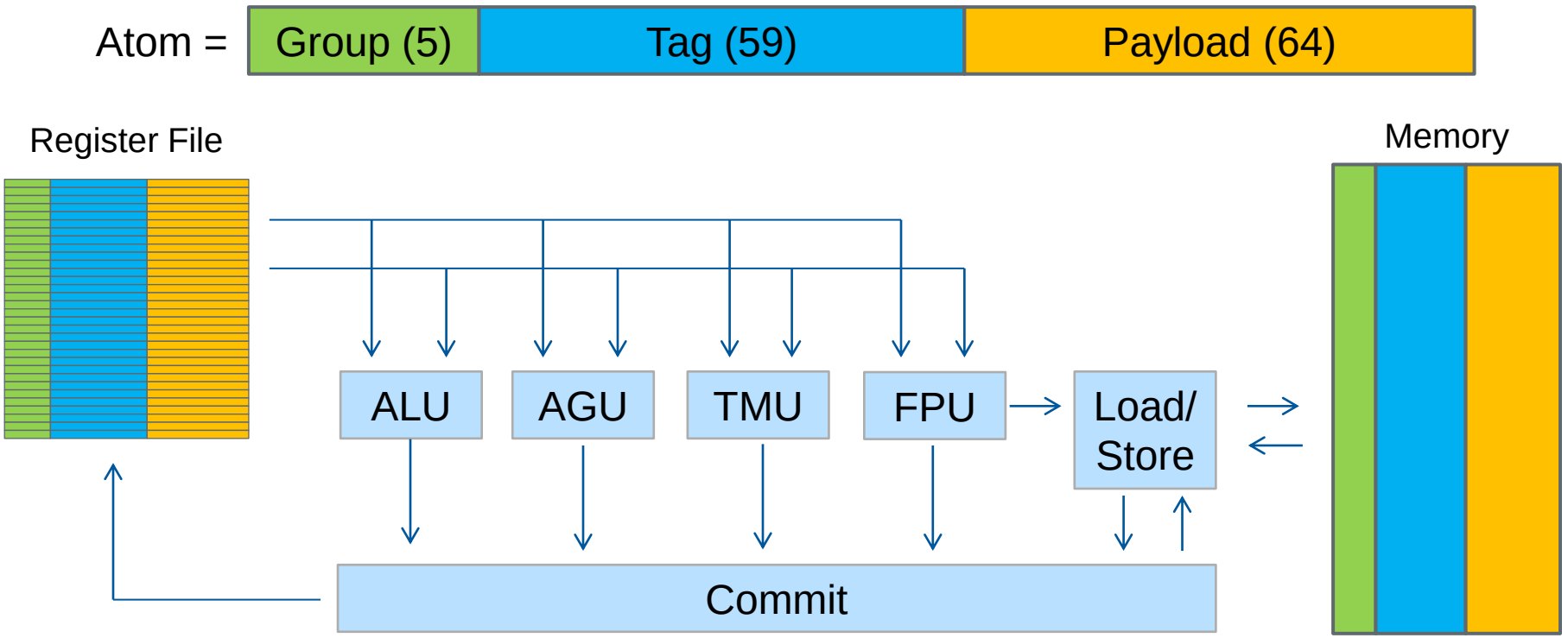
BAE SYSTEMS

The SAFE Solution to Security

- What if we could start from a clean slate?
- SAFE is a Codesign of…
  - A new applications programming language (Breeze).
  - A new system programming languages (Tempest).
  - A new operating system.
  - A new processor .
  - With security at every level for defense in depth.
- Why hardware enforced security?
  - Dynamic security checking is too expensive in software.
    - Fine grained information flow control (IFC).
  - Covers the most general attack model.
    - Scripting attacks down to machine code injection.

BAE SYSTEMS

SAFE Hardware Architecture



- Atomic group unit (AGU) checks atom types, i.e. instructions, data, pointers, streams, etc.
- Fat pointer unit (FPU) check pointer operations.
- Tag management unit (TMU) checks and applies tags.

Starting Project at Day 1

- We have an outline for an ISA, but nothing else.
  - TIARA project as a baseline (Howard Shrobe, Andre DeHon, Thomas Knight).
  - But no languages, no toolchain, no hardware.
- How to proceed?
  - Sketch out an assembly language.
  - Build an instruction set simulator.
  - Start writing and simulating small assembly programs.
  - HW researchers start coding Bluespec.
  - PL researchers start designing Breeze.
    - Plan is to steal Andrew Meyers work on Jif. Port ideas to a dynamic PL.
      - “Breeze should be done in a couple of months.”

BAE SYSTEMS

SAFE Assembly

Frames to manage fat pointer bounds.

Atomic group declarations on data.

Tags on data.

Gate structures for secure closures.

- How long can we keep this up?

Approved for Public Release

BAE SYSTEMS

At Year 1.0

- Assembly is tedious. We need macros.
- Breeze interpreter running. Pressure to start building the compiler.
- Solution: A SAFE assembly DSL embedded in Haskell.
  - Use Haskell is a macro language.
  - Becomes a library for the Breeze compiler.
- Breeze Language, Version 7
  - 4-5 weeks spent on figuring out datatypes for Booleans.
    - “Hmm, this IFC stuff is kind of tricky.”
  - Difficulties arise with access control.
    - Convenience and modularity of lexical authority passing and one-principal-per-module is anything but.

BAE SYSTEMS

SAFE Assembly in Haskell

```
testOffpCode :: Integer -> Program a Label
testOffpCode frameSize = do
  -- frame of values
  valFrame <- frame "Frame of values" $ do
    space frameSize

  -- code frame
  frame "Code frame" $ do

    integer' 0 R0
    integer' 1 R1
    framePointer' valFrame Nothing R2
    testgrp R2 R3 FramePointer
    while R3 $ do
        offp R1 R2 R2
        add R1 R0 R0
        testgrp R2 R3 FramePointer

    integer' frameSize R1
    compareAtoms R5 R1 R0 R6
    ifThenElse R6
      (recordSuccess "done offp")
      (recordFailure "done offp")
    jumpToPassOrFail
```

A Monad to capture programs.

Macros for setting up data.

Macros for better control flow.

BAE SYSTEMS

At Year 1.5

- As a EDSL, Haskell makes for great macros, but it's still assembly.
  - Manual register allocation, calling conventions, and data structures.
- Meanwhile, Breeze compiler inches off ground, but…
  - Awkward transition from high level CPS IR to assembly.
    - We really need an IR somewhere in between.
  - On plus side, SAFE EDSL worked great in code generator.
- Breeze Language, Version 12
  - "What do we do on an access violation?"
  - "Simple. We stop the machine."
  - "But what if I maliciously send you data you can't access?"
  - "Simple, I'll just check the label before I attempt to read it."
  - "But what if the label itself is private?"
  - "Oh..."
    - The Poison Pill problem.

At Year 2.0

- Breeze compiler goes through major overhaul.
  - Some improvement to middle IRs, but still not enough.
  - Breeze compiler is temporarily shelved.
    - Breeze won't come to the rescue of the OS.
  - We REALLY need a higher low-level language.
- Breeze Language, Version 23
  - "We have a solution to poison pills. We'll make all labels public."
    - To label data you must specify the label in advance (brackets).
      - Prevents labels from being information channels.
    - But public labels are not compatible with lexical authority passing.
      - The lexical authority containment problem.
      - Breeze switches to dynamic authority.

BAE SYSTEMS

At Year 2.5

- Tempest is started: The systems programming language for SAFE.
  - Imperative with automatic register allocation and optimizations.
  - Control of assembly with inlining and user specified calling conventions.
  - Uses the SAFE EDSL as a backend.
  - As and EDSL, nicely fills the Breeze compiler IR gap.
- Breeze Language, Version 34
  - Delayed exceptions with not-a-value values (NaVs).
  - Dynamic authority is replaced with clearance.
    - Similar ideas. Both work with public labels.

BAE SYSTEMS

Tempest EDSL with Inline Assembly

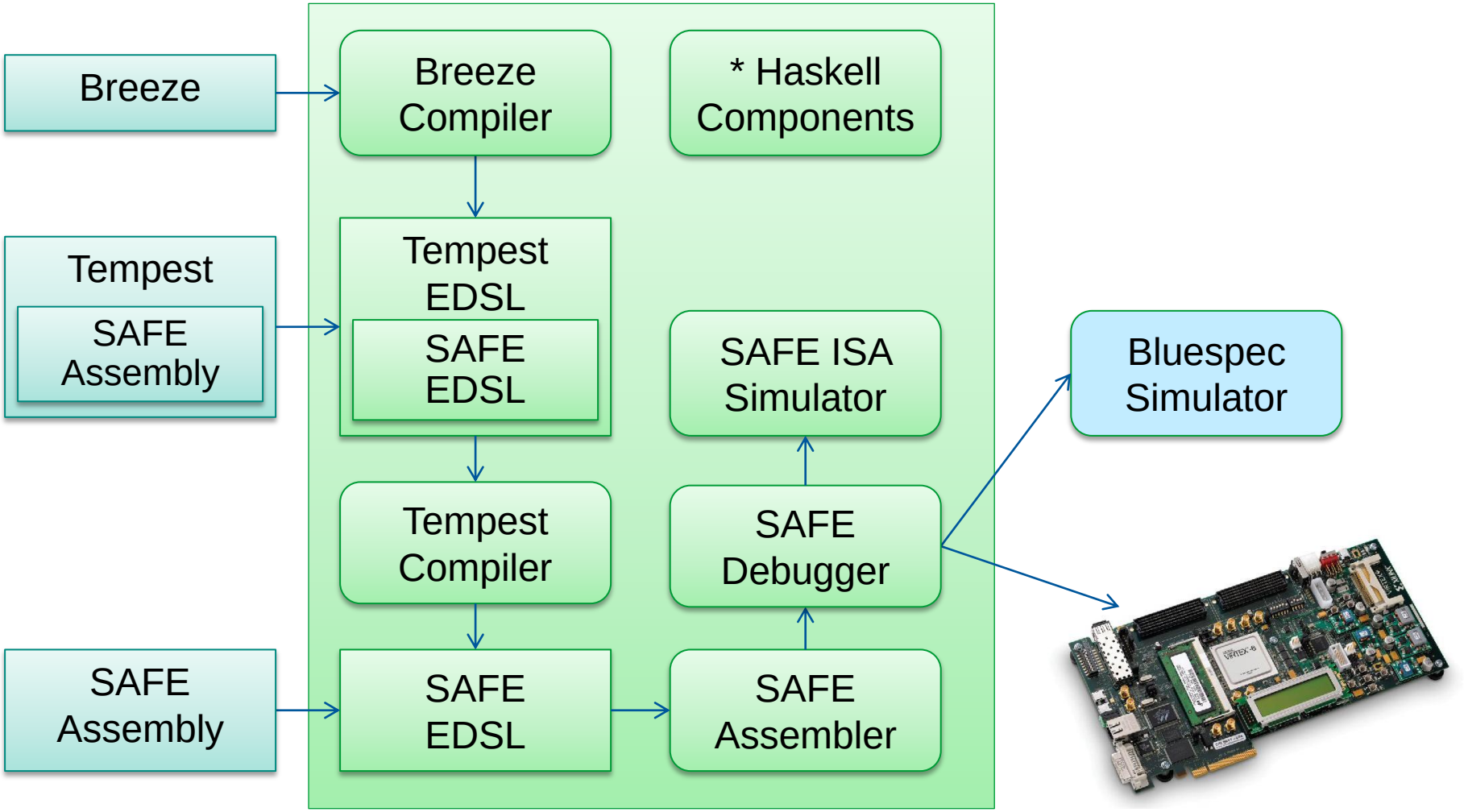
```
(/-/) :: (ToExpr e1, ToExpr e2) => e1 -> e2 -> Expr
a /-/ b = block $ do
  a <- var a
  b <- var b
  return $ asm [intT] $ \ result -> beginAsm $ do
    sub (R a) (R b) (R result)

(/</) :: (ToExpr e1, ToExpr e2) => e1 -> e2 -> Expr
a /</ b = block $ do
  true  <- var 1
  false <- var 0
  diff  <- var $ a /-/ b
  return $ asm [intT] $ \ result -> beginAsm $ do
    trueCase <- label
    end      <- label
    bneg (R diff) trueCase
    mvrr (R false) (R result)
    jmp end
    trueCase -: do
      mvrr (R true) (R result)
    end -: do
      nop
```

SAFE Assembly Sublanguage

The SAFE Flow

BAE SYSTEMS



Lessons Learned (1)

- Designing a higher order IFC language is very hard.
  - Optimal number of PL researchers on a project: 2 to 7
- On day 1, we should have started Tempest, not assembly.
  - Hard to achieve good productivity with assembly code.
  - Tempest is the right level for runtime / processor codesign.
    - The level of indirection provides insulation from a changing ISA.
- EDSLs are great for bootstraping a language.
  - And make excellent backend libraries!

BAE SYSTEMS

Lessons Learned (2)

- EDSLs require that engineers are comfortable with the host language.
- EDSLs are hard to debug.
- Still good reasons for concrete syntax.
  - More relevant for some languages than others.
    - Tempest vs. SAFE assembly.
  - When is the best transition point?
    - Early pressure from developers for modular programming.
    - One language has modularity, the switch can be made.
- Would a DSL have helped hardware design?
  - Forever debugging ISS and FPGA.
  - A DSL describing ISA semantics could keep it synchronized.
    - Generating Bluespec, ISS, SAFE EDSL, Coq, and Documentation.

BAE SYSTEMS

Final Plugs

- SAFE has produced a volume of interesting papers.
  - Private vs. public labels.
  - Lexical authority vs. dynamic authority vs. clearance.
  - Exception handling in IFC.
  - Efficient tag processing in hardware.
  - Efficient fat pointer encoding.
  - See: http://www.crash-safe.org/papers
- At ICFP this week: "Testing Noninterference, Quickly"
  - Catalin Hritcu, John Hughes, Benjamin C. Pierce, Antal Spector-Zabusky, Dimitrios Vytiniotis, Arthur Azevedo de Amorim and Leonidas Lampropoulos.
  - Using QuickCheck to test ISA security.
- We're Hiring!
  - Needed: Functional compiler engineers for Breeze and Tempest.

BAE SYSTEMS

Thanks!

http://crash-safe.org/

The SAFE Team:
Tim Anderson, Arthur Azevedo, Silviu Chiricescu, Nathan Collins, David Darais, Andre DeHon, Delphine Demange, Udit Dhawan, Richard Eisenberg, Mikel Evins, Marty Fahey, Karl Fischer, Greg Frazier, Anna Gommerstadt, Michael Greenberg, Tom Hawkins, Hillary Holloway, Catalin Hritcu, Suraj Iyer, Andrew Kaluzniaki, Ben Karel, Aleksey Kliger, Thomas Knight, Basil Krikeles, Marc Krull, Albert Kwon, Leonidas Lampropoulos, May Leung, Bryan Loyall, Gregory Malecha, Josh McGrath, Benoit Montagu, Greg Morrisett, Luke Palmer, Sam Panzer, Plamena Petrova, Greg Pfeil, David Pichardie, Benjamin Pierce, Randy Pollack, Sumit Ray, Howard Reubenstein, Jothy Rosenberg, Andrea Ruggiero, Olin Shivers, Jonathan Smith, Shannon Spires, Nancy Stafford, Amanda Strnad, Gregory Sullivan, Adrien Suree, Andrew Sutherland, Arun Thomas, Andrew Tolmach, Jesse Tov, Peter Trei, Nick Watson, Chris White, David Wittenberg, Zach Zarrow.

Special Thanks to Howie Shrobe at DARPA