Controlling Hybrid Vehicles with Haskell

Powering Business Worldwide

Overview

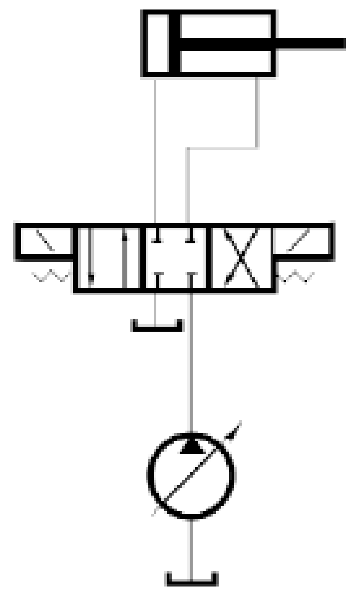
- Eaton
- Hydraulics Hybrid Vehicles
- Haskell @ Eaton
- Atom: A DSL embedded in Haskell
- Functional Programming Challenges

Eaton: Powering Business Worldwide

- Diversified Power Management
- Cleveland, OH; 81,000 Employees; $13 Billon in Sales
- Markets & Products
  - Electrical
    - circuit beakers, power distribution assemblies, uninterruptible power systems
  - Aerospace
    - hydraulics, fuel systems, motion control, circuit protection
  - Truck
    - transmissions, clutches, electric hybrid powertrain systems
    - cruise control, collision warning, traction control systems
  - Automotive
    - air, transmission, and fuel management controls
    - superchargers, differentials
  - Hydraulics
    - valves, pumps, motors, cylinders, fluid conveyance, filtration
    - Golf Grips???
      - Eaton: Powering Business and Golf Balls Worldwide

Hydraulic Control Valves

- Proportional Valves for Directional Control



Hydraulic Accumulators

- Compressed Nitrogen Stores Energy

Hydraulic Pumps and Motors

- Axial Piston Pump
  - Positive Displacement
  - Variable Displacement

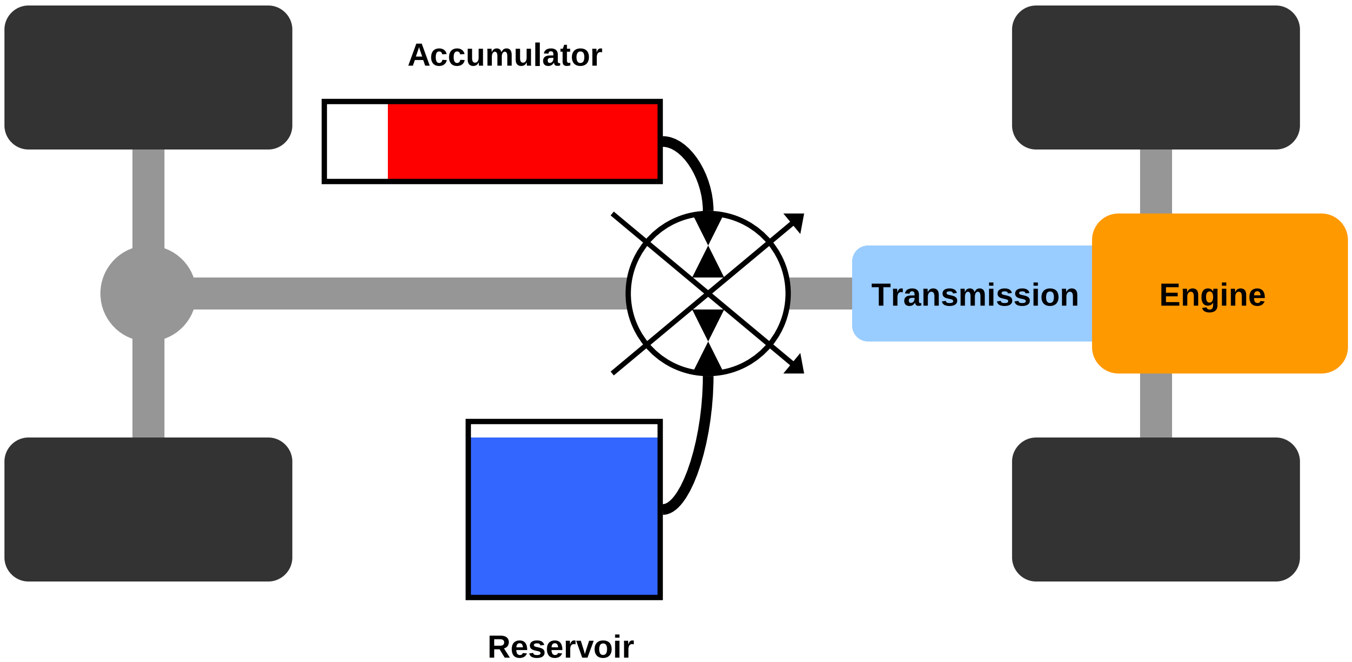
Powering Business Worldwide

Hydraulic Pumps and Motors

- Bent-Axis Piston Pump

EATON
Powering Business Worldwide

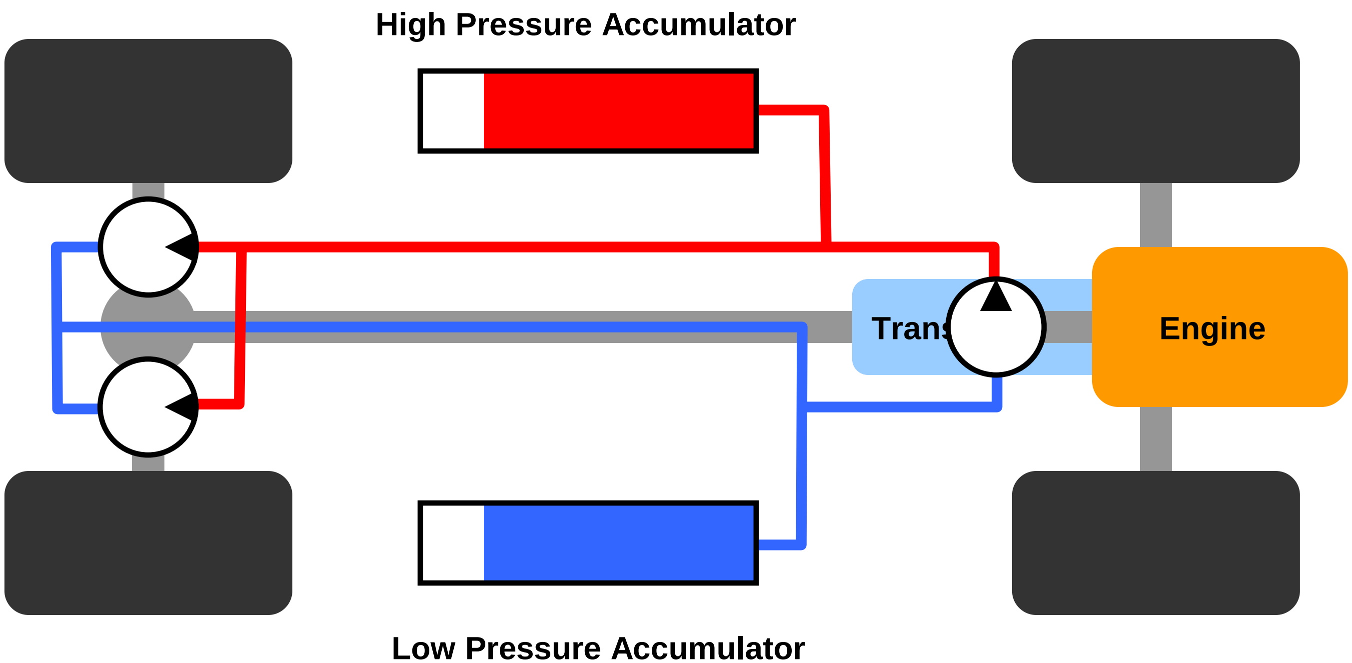
Building a Hydraulic Hybrid



- Parallel Hybrid: Hydraulic Launch Assist (HLA)
  - Augments conventional drivetrain.

Powering Business Worldwide

Building a Better Hydraulic Hybrid



- Series Hybrid: Decouples Engine from Wheels
  - Run engine at optimal RPM. Shut off when not needed.
  - Opens door to alternative engines: Free piston, HCCI.

HLA (Parallel) System

Haskell for Day to Day Stuff

- Scripting, Data Conversion, Field Tools, etc.
- ECU Flash Programming
- Hardware-in-the-Loop Simulation
- Differential Cryptanalysis
- Remote Vehicle Management
  - Data logging, calibration, and re-programming through WiFi and cell modems.
  - Fountain codes for forward error correction.
    - Distributed download, multicast.

Atom DSL

- Inspired by...
  - Bluespec (Arvind and friends)
  - STM
- Atom: Atomic State Transition Rules
  - For embedded hard-real-time control software.
  - Haskell + Atom = Safer Software
    - Concisely express safety related behavior.
- Atom Compiler Automates:
  - Multi-Rate Thread Scheduling
    - No need for RTOS task scheduler.
  - Multi-Rate Thread Synchronization
    - No need to program with locks and semaphores.
  - Multi-ECU Software Partitioning*
    - No need to explicitly program ECUs independently.
    - * not implemented yet, but possible

Atom Semantics

Enabling Conditions

```
balance <- double 0
```

```
system “deposit” $ do
  when depositRequest
  balance <== value balance + amount
```

```
system “withdraw” $ do
  when withdrawRequest
  when $ value balance >=. amount
  balance <== value balance - amount
```

Actions

Rules Execute Atomically

Powering Business Worldwide

Atom Types and Values

```
data System a  -- System monad collects variable and rule definitions.
data Var a     -- State variables.
data Term a    -- Combinational expressions of variables.
               -- Term Bool, Term Word8, Term Int16, Term Double, etc.

-- Variable declarations.
bool   :: Bool   -> System (Var Bool)
word8  :: Word8  -> System (Var Word8)
int32  :: Int32  -> System (Var Int32)
double :: Double -> System (Var Double)

-- Variable reference.
value :: Var a -> Term a

-- Term operations.
inv   :: Term Bool -> Term Bool
(&&.) :: Term Bool -> Term Bool -> Term Bool
(==.) :: Term a -> Term a -> Term Bool
(<.)  :: Term a -> Term a -> Term Bool
mux   :: Term Bool -> Term a -> Term a -> Term a
-- Instances of Num, Fractional, Floating, Bits, etc.
```

Atom Types and Values

```
-- Building system hierarchy.  Each hierarchal node could be rule.
-- Child system inherits parents execution rate.
system :: Name -> System a -> System a

-- Building hierarchy with timing information.
-- Child system executes at a factor of parent's rate.
systemPeriodic :: Name -> Int -> System a -> System a

-- Enabling conditions.
when :: Term Bool -> System ()

-- Variable assignment.
(<==)  :: Var a -> Term a -> System ()

-- Compile an Atom description.
-- Specify top level name and base execution period.
compile :: Name -> Double -> System () -> IO ()
```

EATON Powering Business Worldwide

Atom Example: HLA Disengagement Fault

~/eaton/atom

```haskell
module Faults (failedToDisengage) where

import Atom

-- | Monitor of HLA clutch disengagement.
failedToDisengage :: Term Bool -> Term Bool -> System (Term Bool)
failedToDisengage clutchCommand clutchFeedback = system "failedToDisengage" $ do

  armed <- bool False      -- Fault armed.
  fault <- bool False      -- Fault active.
  timer <- timer "timer"   -- Timer.

  system "armFault" $ do
    when $ inv $ value armed
    when $ inv clutchCommand &&. clutchFeedback
    armed <== true            -- Arm fault.
    startTimerSec timer 0.5   -- Start timer for 1/2 second.

  system "disarmFault" $ do
    when $ value armed
    when $ clutchCommand ||. inv clutchFeedback
    armed <== false
    fault <== false

  system "activateFault" $ do
    when $ value armed
    when $ timerDone timer
    fault <== true

  return $ value fault

"Faults.hs" 31L, 836C written                                              1,1          All
```

Powering Business Worldwide

Compiling Atom: Rule Scheduling

- Each Rule Associated with an Execution Period
  - "Threads" are sets of rules with same period.
- Schedule rules to balance processing.
  - Returns single C function to be called at base rate.

```
rule A (1ms, 10 instructions)

rule B (2ms, 5 instructions)

rule C (2ms, 10 instructions)

rule D (2ms, 5 instructions)
```

```
sample 1
   rule A # 10
   rule B # 10
--rule-C-#-10
   Total D # 20
-------------
   Total    30
```

```
sample 2
   rule A # 10
--rule-B-#--5
   Total D # 10
-------------
   Total    20
```

- Advanced scheduling is possible.
  - eg. Splitting a rule execution across multiple samples.

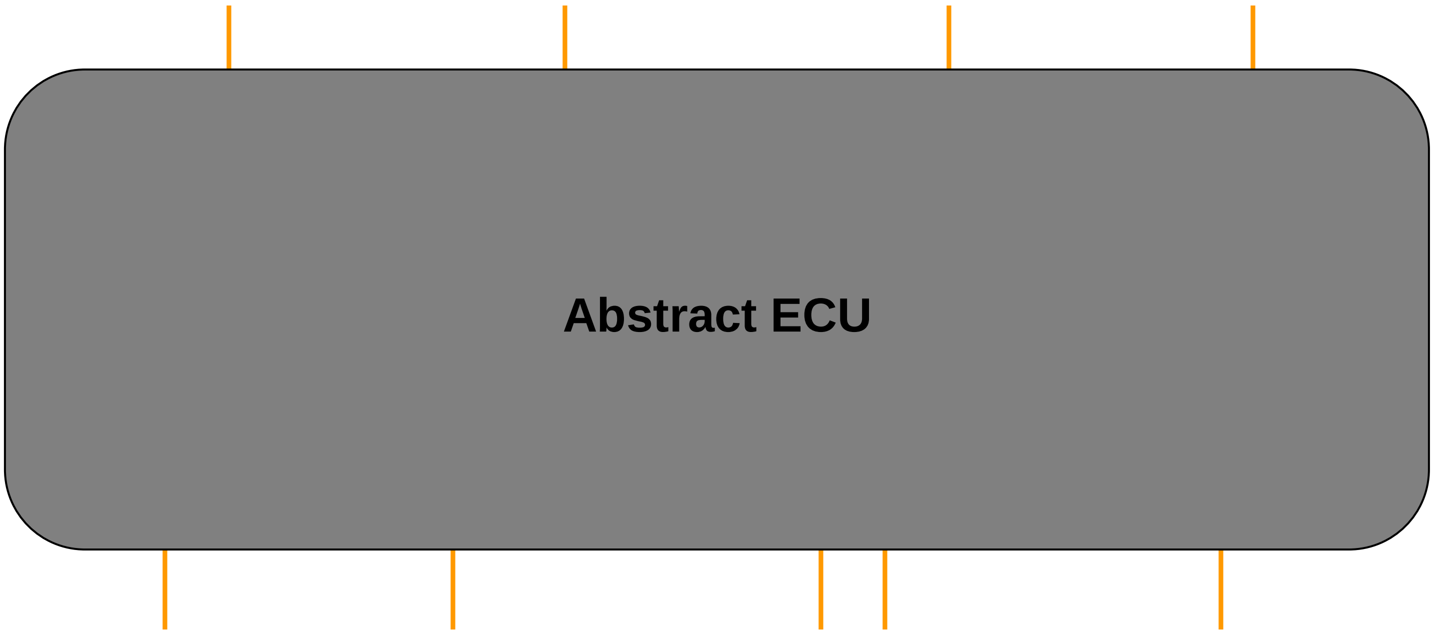
EATON Powering Business Worldwide

Compiling Atom: Rule Scheduling

- Thread Scheduling
  - Compiler does the scheduling, not the OS.
  - Timing semi-verified by compiler.
- Thread Synchronization
  - Compiler adheres to rule atomicity.
  - No need to program with locks and semaphores.
    - Yeah! Life is Good!

EATON
Powering Business Worldwide

Compiling Atom: Multi-ECU Partitioning



- Program the system as a whole. Let the compiler handle...
  - ECU allocation.
  - ECU communication and synchronization.
- Multiple ECUs for redundancy (ie. safety).
  - Requires new compiler constraints: availability and integrity.
  - Which rules are important, and which are less so?

Challenges

- Limitations with Meta Programming
  - instance Eq (Term a).
    - Equality comparison of deep combinational expressions.
  - GADTs only for Meta, not Object Language
  - Considered direct compilation, via YHC.
    - System, not IO, as top monad.
- Functional Programming is a Tough Sell
  - No traction with former 2 employers.
  - Eaton is different.

Succeeding with Functional Programming at Work

- Declare what to compute, not how to compute it.
  - It’s easier to ask forgiveness than to get permission.

Real Haskell Garbage Collection

- Mark and Sweep? No, Clump and Dump!

EATON

Powering Business Worldwide