Accelerating Stateflow With LLVM

By Dale Martin
Dale.Martin@mathworks.com
What is Stateflow?

• A block in Simulink, which is a graphical language for modeling algorithms
What is Stateflow?

• A block in Simulink, which is a graphical language for modeling algorithms

• The Stateflow block models control flow by graphically modeling state transition diagrams, flow charts, and truth tables
What is Stateflow?

- A block in Simulink, which is a graphical language for modeling algorithms
- The Stateflow block models control flow by graphically modeling state transition diagrams, flow charts, and truth tables
Traditional Simulation Approach

Simulink Simulation Engine

Simulink built-in block algorithms

Simulation target DLL

chart2.c  chart1.c  fcn.c

sum  gain  scope
Simulation through Code Generation

- **Pros**
  - Simulation and production code generation based on same technology => less code, fewer bugs
  - Much faster runtime than “interpreted” implementation
  - Easy to call customer-supplied C/C++ code

- **Cons**
  - First time overhead due to generating code and invoking a compiler
  - Needs an external C-compiler
    - (Different levels of difficulty to customers on Mac, Linux, and Windows)
How can we target LLVM instead of C?
How can we target LLVM instead of C?

- A new compiler backend – from our internal IR, to LLVM
About our internal IR

- Represents a high level of abstraction – matrix operations, fixed-point and complex math, structures, complex control flow
- Gets progressively lowered into multiple backend languages – C, VHDL, Verilog, PLC Structured Text
About our internal IR

- Represents a high level of abstraction – matrix operations, fixed-point and complex math, structures, complex control flow
- Gets progressively lowered into multiple backend languages – C, VHDL, Verilog, PLC Structured Text
- And now LLVM!
An observation: we’re really good at working with our own IR

- Many good debugging tools
- Many experts in the building
An observation: we’re really good at working with our own IR

- Many good debugging tools
- Many experts in the building
- Let’s map our semantics onto LLVM in our own IR
What does that mean exactly?

- Like our “normal” compiler flows, we do “lowerings” to go from high-level abstractions to lower level abstractions
  - Lower matrix operations into loops
  - Fixed-point math into integer math, etc
What does that mean exactly?

- Like our “normal” compiler flows, we do “lowerings” to go from high-level abstractions to lower level abstractions
  - Lower matrix operations into loops
  - Fixed-point math into integer math, etc.
- In addition, we go further
  - Booleans become int1 or int8 depending on context (control flow vs. data)
  - Unions get mapped into Structures with one field; accesses get turned into cast operations
  - Many more examples
Where do we end up?

- A syntactically legal version of our own IR
- That maps one-to-one onto LLVM IR
Where do we end up?

- A syntactically legal version of our own IR
- That maps one-to-one onto LLVM IR
- This makes the translation really simple
**R2015a: Just-in-Time (JIT) Compilation**

Simulink Simulation Engine

- `chart2()`, `chart1()`, `fcn()`
- Optimized block functions in memory without C-code

- Simulink built-in block algorithms
  - `sum`
  - `gain`
  - `scope`
JIT-based Simulation in R2015a

- No need for a C compiler 😊
- Fast startup 😊😊

- Transparent to the user
  - No knobs or buttons or options
  - Model compilation speeds up through JIT when it can

- Automatically fall back to codegen modes as needed
  - e.g., Custom code, and step-by-step debugging use code generation
JIT Model Compile Time Improvement

Data from >5000 internal test models
- 99% of the models are 20-50% faster
Challenges

- Supporting Linux, Mac, and Win64
- Our runtime can throw exceptions
  - On win64, LLVM can’t handle exceptions passing through
  - Wrote a pass (in our IR) to wrap every runtime call with error checks/early returns
Challenges

- Discovered the hard way that MC-JIT does not really work on Windows with LLVM 3.5 or 3.6 😞
- Due to release schedule, stuck at 3.5 and legacy JIT for now
Questions?

- Come find me or email Dale.Martin@mathworks.com