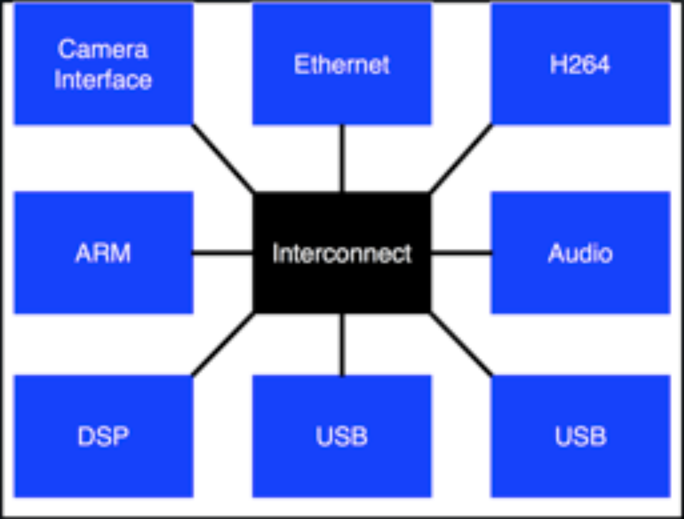
Building SOC’s Efficiently

Derek Pappas

FastpathLogic

How do you build and verify SOC’s efficiently?



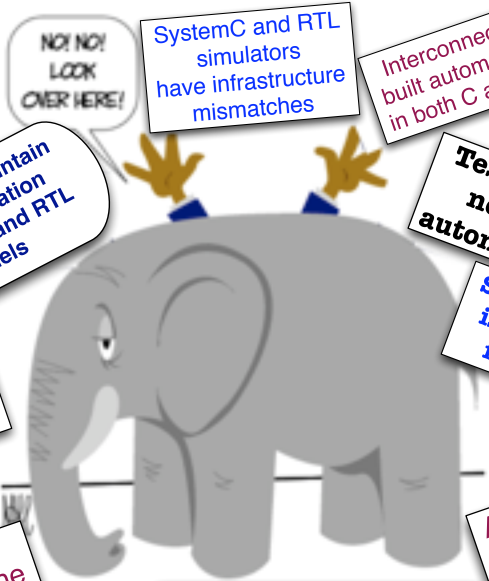
The Challenge

- Reduce the time to build complex SOC's

Data formats

	RDL	Atrenta	IP XACT	Bluespec	FPL
Single spec	🟥	🟥	🟥	🟥	🟩
GUI	🟥	🟥	🟥	🟥	🟩
All Infrastructure	🟥	🟥	🟥	🟥	🟩
Verilog/VHDLSysC	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩
	🟥	🟥	🟥	🟥	🟩

Outline



SystemC and RTL simulators have infrastructure mismatches

Interconnect not built automatically in both C and RTL

Difficult to maintain synchronization between C and RTL models

Testbenches not built automatically

SOC design tool flow is fragmented/not fully automated

Creating custom on chip buses time consuming

Registers do not "talk" to the interconnect

Adding custom instructions to processors has issues

SOC design tool flow is fragmented/not fully automated

How do you reduce the SOC verification and design time

- Find the places that engineers spend time fixing problems and automate those processes

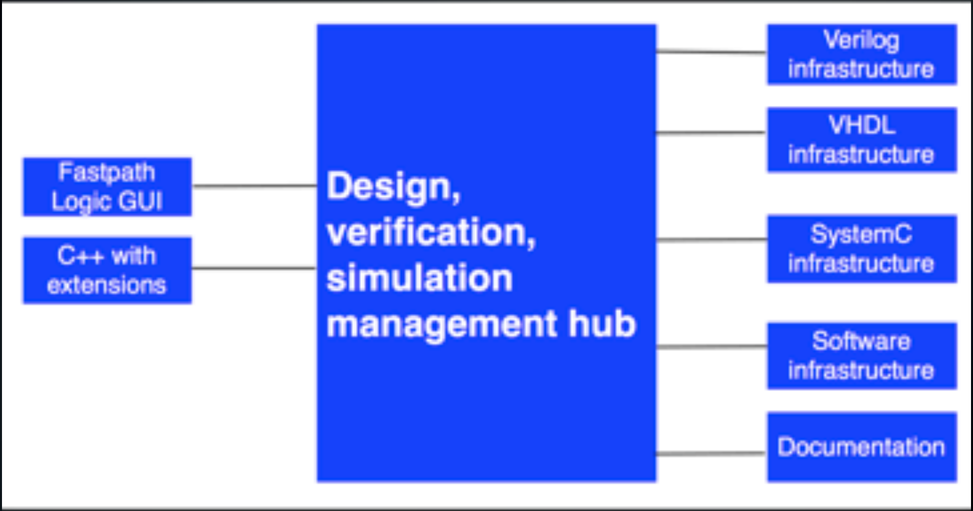
Use a single specification to generate equivalent software and hardware infrastructure code

- Reduce time to working code-not just coded
- Ensure that the software and hardware are “speaking the same language”
- Make changes rapidly in the same project and future proliferation projects

Hand tools or ...

Power tools

Design management hub



Need to build equivalent models automatically

- Algorithmic C/C++ models
- Behavioral SystemC models
- RTL implmentations

Compare the different models

- In order to compare the different models infrastructure components need to be the same in all models
  - Interconnect-interfaces
  - Address map/registers-fields/enums/register files/memories
  - instruction set architecture-formats, fields,

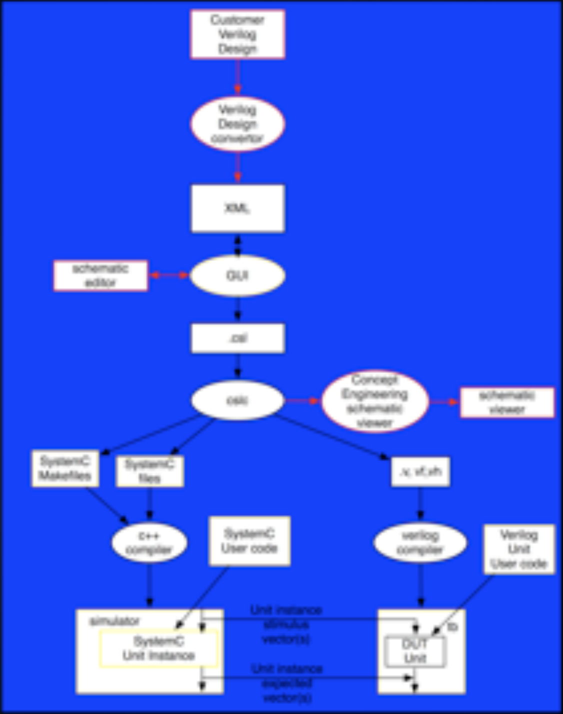
Generate the interconnect automatically

- Specify the following:
  - design hierarchy
  - connection library
  - connections using endpoints
  - auto router creates intermediate connections

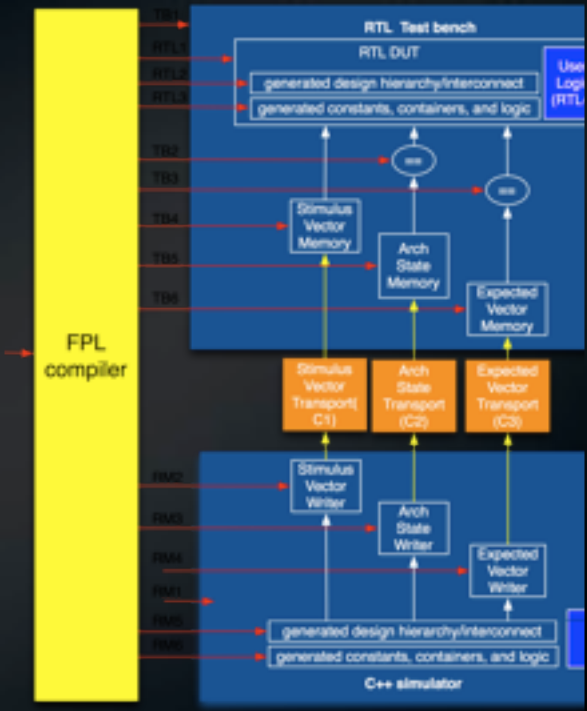
SoC's have many networks

- On chip buses-Amba, OCP, custom (token ring, tree, ...)
- Out of band-control, query, status, performance
- Testability-JTAG talking to MBIST, I/O loopback, self test,...
- Memory controller/client

Generation of shared interfaces: unit to unit unit to vector vector to unit



As a result “connected” components have the same interfaces

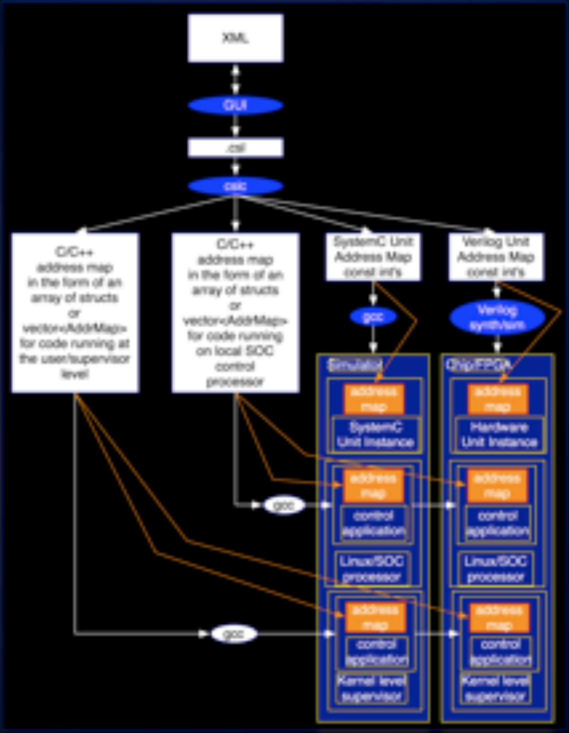

FPL compiler
TB1
RTL1
RTL2
RTL3
TB2
TB3
TB4
TB5
TB6
RM2
RM3
RM4
RM1
RM5
RM6
RTL Test bench
RTL DUT
generated design hierarchy/interconnect
generated constants, containers, and logic
User Logic (RTL
==
==
Stimulus Vector Memory
Arch State Memory
Expected Vector Memory
Stimulus Vector Transport(C1)
Arch State Transport (C2)
Expected Vector Transport (C3)
Stimulus Vector Writer
Arch State Writer
Expected Vector Writer
generated design hierarchy/interconnect
generated constants, containers, and logic
C++ simulator

SOC’s and software use common addressing

- Generate the address map software and hardware code automatically
  - address maps
  - address decoders
  - registers
    - fields
    - enums

The address maps are used in many places in a system.

This is complicated.

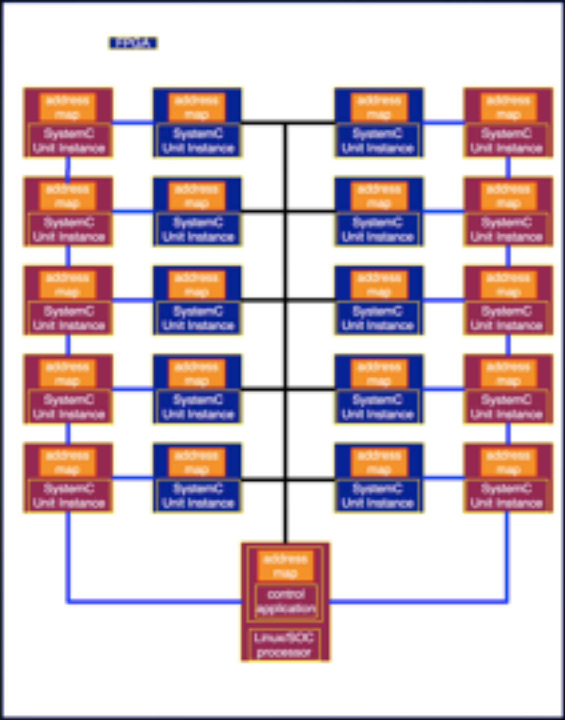


SOC's and software use common addressing

- problem wrt to productizing network IP due to the addressing issues.
- Asking engineers to write new software for each device is time consuming and needs to be automated.
- This is a problem with any network IP product.

The address maps are used in many places on a chip.

This is very complicated.



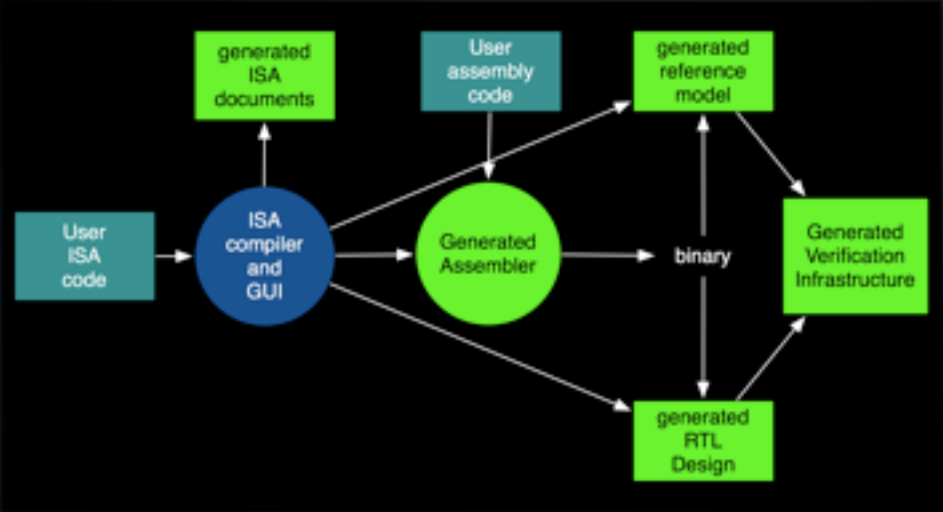
Should the memory map generation process be automated?

- Specify the following:
    - Instruction set architecture
    - pipelines
    - logic units in processor

Generate the assembler and processor automatically

- Specify the following:
  - Instruction set architecture
  - pipelines
  - logic units in processor

Process to build


generated ISA documents
User assembly code
generated reference model
User ISA code
ISA compiler and GUI
Generated Assembler
binary
Generated Verification Infrastructure
generated RTL Design