A Modular Digital VLSI Flow for High-Productivity SoC Design

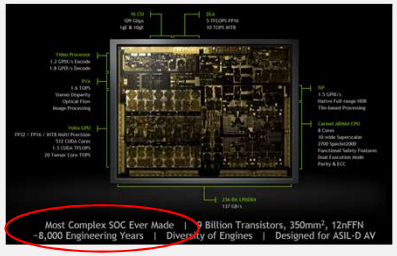
[1]Brucek Khailany
[2]Gu-Yeon Wei, [3]David Brooks

[1]Director of ASIC and VLSI Research at NVIDIA
[2]Proessor of Electrical Engineering and Computer Science at Harvard, [3]Professor of Computer Science at Harvard

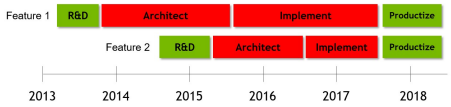
Designs Thrust: Circuit Realization at Faster Timescales (CRAFT)

The Design Complexity Challenge

Example Digital SoC:
NVIDIA Xavier Self-Driving Car SoC [CES 2018]

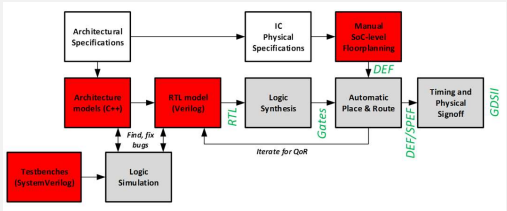


Typical development timeline: 3-5 years from R&D to product:



- RTL design and verification dominates: >70% of IC design effort at NVIDIA
  - Prohibits which features make it into each SoC
- Need 10x lower design and verification effort:
  - Overlap architect & implement phases for faster time-to-market, more features

Problem: RTL Design and Verification Effort



Architects write architectural simulators in C++, verify performance

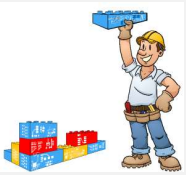
An ASIC design engineer re-implements the architectural spec in Verilog but is only able to test the code using a Verilog simulator that runs 10^6 times slower than the production hardware.

An ASIC verification engineer makes sure that the two pieces of code are 100% functionally equivalent before tapeout using SystemVerilog code only understood by ASIC verification experts

Object-Oriented HLS-based Design

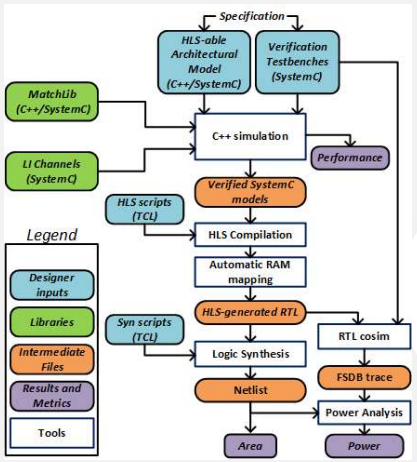
Object-Oriented HLS (OOHLS)

- Leverage HLS tools to design with C++ and SystemC models
- MatchLib: Library of commonly-used micro-architectural components in HLS-able C++
- All communication between SystemC modules through Latency-Insensitive Channels and/or on-chip networks
- Target: 10x productivity of manual RTL coding



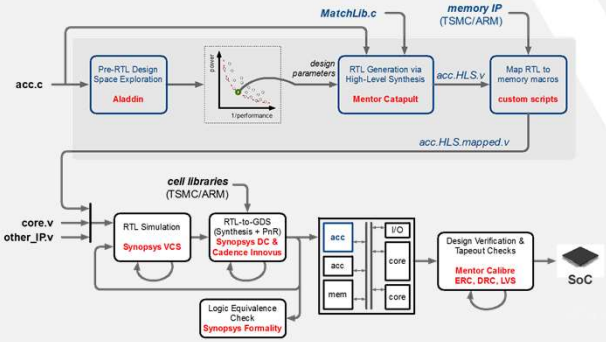
MatchLib: Modular Approach To Circuits and Hardware LIBrary

- Goal: "STL/Boost" for Hardware
- Untimed C++ and loosely timed SystemC models
- Compatible with HLS for mapping to RTL
- Addresses a usability and Quality-of-Results gap with existing HLS toolflows
- Open source release planned in 2H 2018

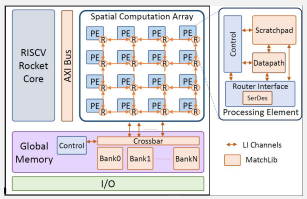


	Component	Description
functions	nv_float	Floating-point arithmetic functions (mul, add, mul-add)
	Crossbar	N-to-N configurable bit width cross-bar switch
	Encoder/Decoder	1-hot encoders and decoders
	FIFO	Configurable FIFO
classes	Arbiter	1-out-of-N round-robin selector
	Mem_array	Abstract memory class
	nvhls_vector	Vector helper container w/ vector operations
	nvhls_array	Array helper container to work around Catapult issues
	Connections	Modular IO supporting latency-insensitive channels
	Arbitrated Crossbar	Crossbar with conflict arbitration and queuing
	Arbitrated Scratchpad	Scratchpad memories with arbitration and queueing
	Reorder Buffer	Out-of-order writes into queue, in-order reads
modules	Serializer/Deserializer	N-bit packets to/from M cycles of (N/M)-bit packets
	Cache	Parameterized by linesize, capacity, associativity
	Scratchpad	Banked memory array with crossbar
	DataStreamer	DMA-like functionality
	SFRouter	Store-and-Forward NoC router
	WHVCRouter	Wormhole NoC router with virtual channels
	AXI Components	Master/Slave Interfaces & bridges for AXI interconnect

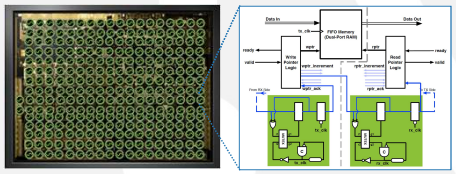
Example C++-to-layout Design Flow



ML Accelerator Design



High-Productivity Clocking Methodology



Pausible Bisynchronous FIFO for low-latency error-free interfaces *[Keller et al., ASYNC 2015]*

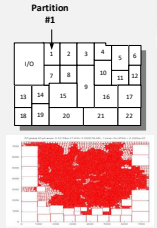
Fine-grained Globally Asynchronous Locally Synchronous Adaptive Clocking

- 100s-1000s of on-chip per-partition adaptive clock generators
- Low-latency error-free clock domain crossings
- "Correct by construction" top-level timing closure
- Reduce voltage margin needed for power-supply noise

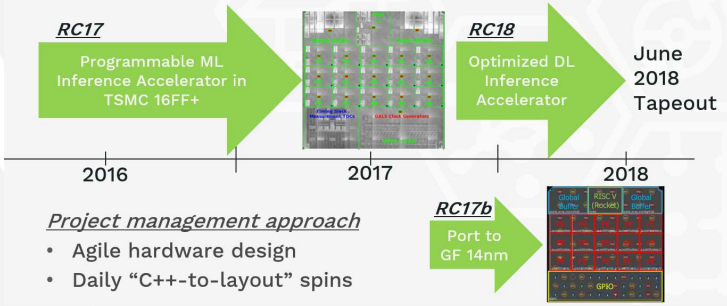
Physical Design and Floorplanning

Small Abutting Auto-floorplanned Partitions

- <24 hour iteration time through place-and-route tools
- Auto-floorplanning algorithms for bounding box, pins, macros
- Minimize wirelengths and overlaps via gradient descent and multilevel clustering
- RAM legalization via simulated annealing



SoC Testchip Demonstrations



Project management approach

- Agile hardware design
- Daily "C++-to-layout" spins

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA)
The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government

THE ELECTRONICS RESURGENCE INITIATIVE