FPGA Overview
And
FPGAs vs. Processors/ASICs/GPUs

Scott Borisch
R&D IC (FPGA) Engineer, Keysight Technologies
February 10, 2020

Background

- 23 years FPGA experience ...
  - 8 years National Instruments
  - 7 years Anue Systems (acquired by =>)
  - 5 years Ixia (acquired by =>)
  - 3 years Keysight Technologies
- … focused on test equipment
  - High precision electronic measurement
  - Ethernet networking test equipment

Slides will be available for download

FPGA Applications

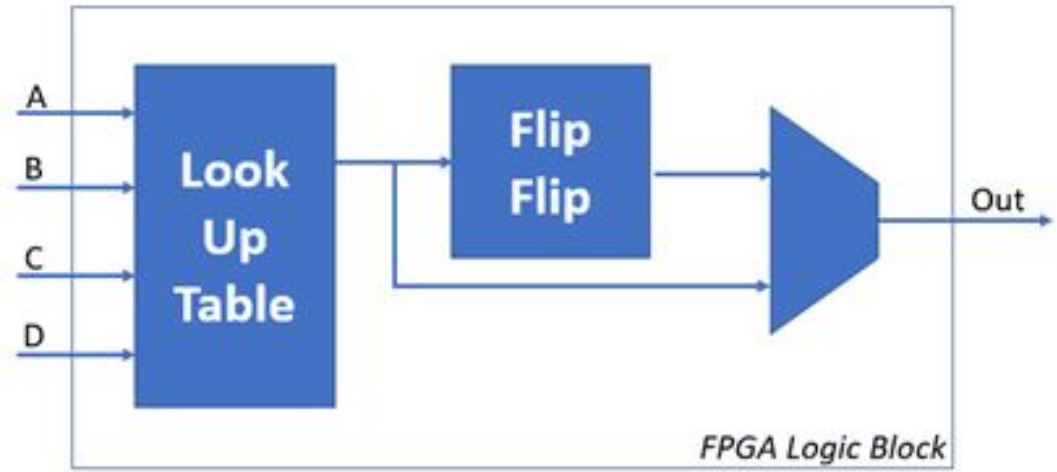
- Aerospace/Defence
- Automotive
- Broadcast TV/Professional A/V
- Consumer Electronics
- Industrial (Robotics, Motion Control, Machine Vision, 3D Printing)
- Medical (Ultrasound, MRI, Robot Assisted Surgery)
- Test and Measurement
- Mobile Telephony (5G wireless)
- Wired Communications (Ethernet)
- Database and Data Analytics
- Finance
- High Performance Computing/Artificial Intelligence/Machine Learning
- Network Acceleration
- ASIC Prototyping

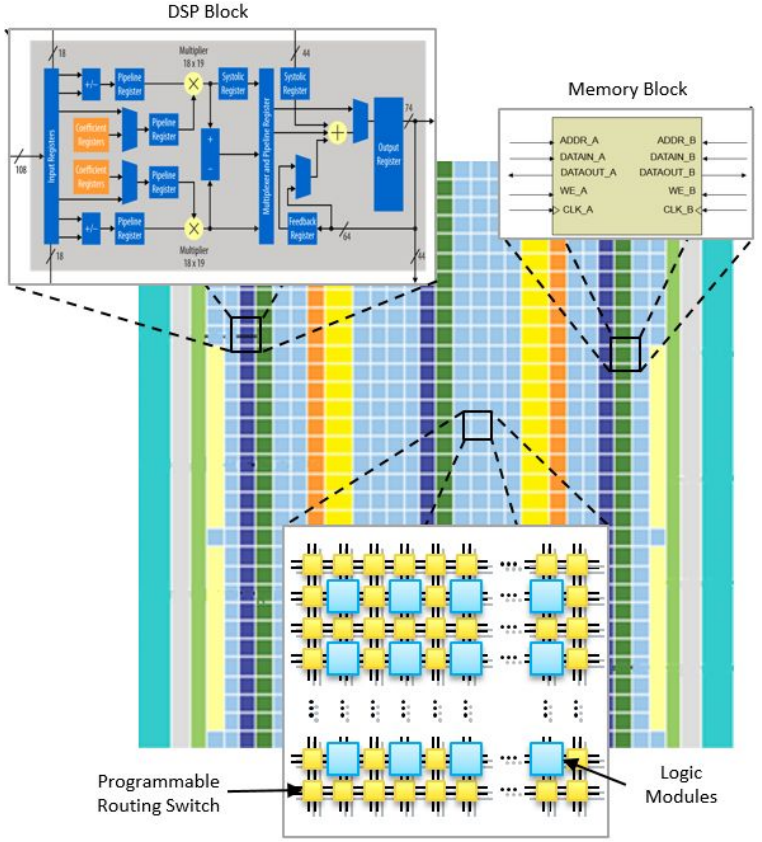
(from Xilinx website)

FPGA Architecture Overview

- Look-up tables + Registers (Flip-Flops)
  - Use these to make adders, counters, shift registers, multiplexers, state machines, etc.
- On-chip memories
- Dedicated fixed point (and floating point) multipliers
  - Fixed point multipliers in FPGAs have been around for awhile
  - Floating point multipliers are finally present in top of the line families (Xilinx Versal, Intel Stratix 10)
  - No FPGA (I am aware of) has native division support
- **The above three items, combined, are often referred to as “FPGA Fabric”**
- Chip-to-Chip and Board-to-Board Communication
  - Ethernet, PCIe, SATA, etc.
  - Uses “SERDES” (serializer/de-serializer) and PLLs (Phase Locked Loops -- Clock Multiplier/Divider/Clock Recovery)
- Newer features
  - Embedded processors (typically ARM)
    - You can also have a (simple) processor in your FPGA fabric! Intel NIOS/Xilinx Microblaze
  - “Network on a chip” (NOC)
  - Embedded DRAM Memory (HBM)
  - “AI Engines” (Simple RISC processors with Fixed/Floating Point ALUs and memories)

FPGA Architecture Overview





Comparison 1 of 2

- Processors
  - Best choice for high complexity tasks
  - Often not a good choice for simpler tasks with very high performance requirements (unless supported by dedicated co-processor)
    - Example: encrypting/decrypting massive amounts of data
  - Lots of software programmers available
  - Faster to implement features in C/Python/Java/etc. than in RTL languages (System Verilog/VHDL) for FPGAs.
- ASICs (Application Specific Integrated Circuit)
  - Will always have best performance
  - Least flexibility to design changes
  - Extremely high development cost -- only practical for high volume products
    - IC Mask NRE costs are extremely high
    - RTL developers are harder to find than SW developers

Comparison 2 of 2

- FPGAs (Field Programmable Gate Array)
  - Excellent for high performance applications at low to medium product volumes
    - Processors often unable to FPGA match processing rate -- unless you go with extremely expensive multi-core processors (high end Intel Xeon, etc.)
    - Very good for deterministic response times
    - Very good for low latency task
  - Per unit cost often higher than processors, but no development NRE like ASICs
  - Not as good as ASICs for raw performance (or performance per watt)
  - Development cost lower than ASICs, but higher than processors
- GPUs (Graphics Processing Unit)
  - Very high performance -- IF your problem matches GPU architecture
  - Very good for high performance math that can tolerate deep pipelines
    - Not good for control logic (conditional execution)

Xilinx Versal Example: “AI Engine”

https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf

SIMD: Single Instruction Multiple Data

VLIW: Very Long Instruction Word

DSP: Digital Signal Processing

FIR: Finite Impulse Response (Digital Filter)

LTE20: Part of the 5G wireless spec

Baseband: a communication signal, like a digitized human voice or music

Heterodyne: To shift a baseband signal to a higher frequency for radio Transmission

Crest Factor: Ratio of average to Peak signal power

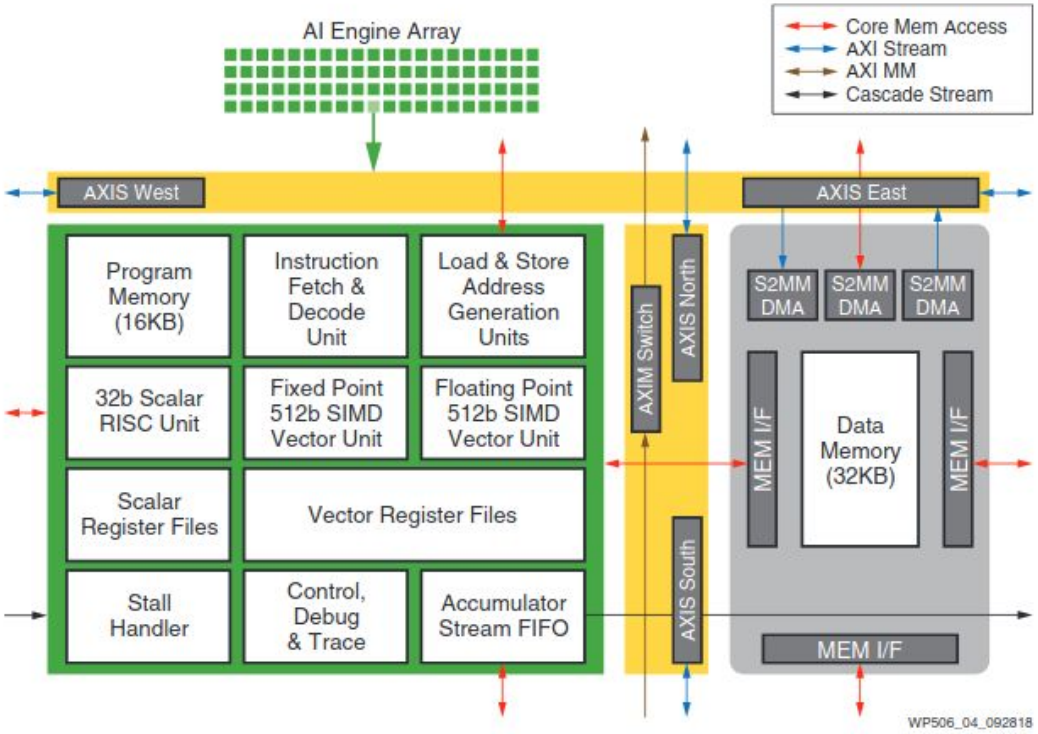


Figure 4: **Detail of AI Engine Tile**

Xilinx Versal Example: Processor + FPGA + “AI Engine”

https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf

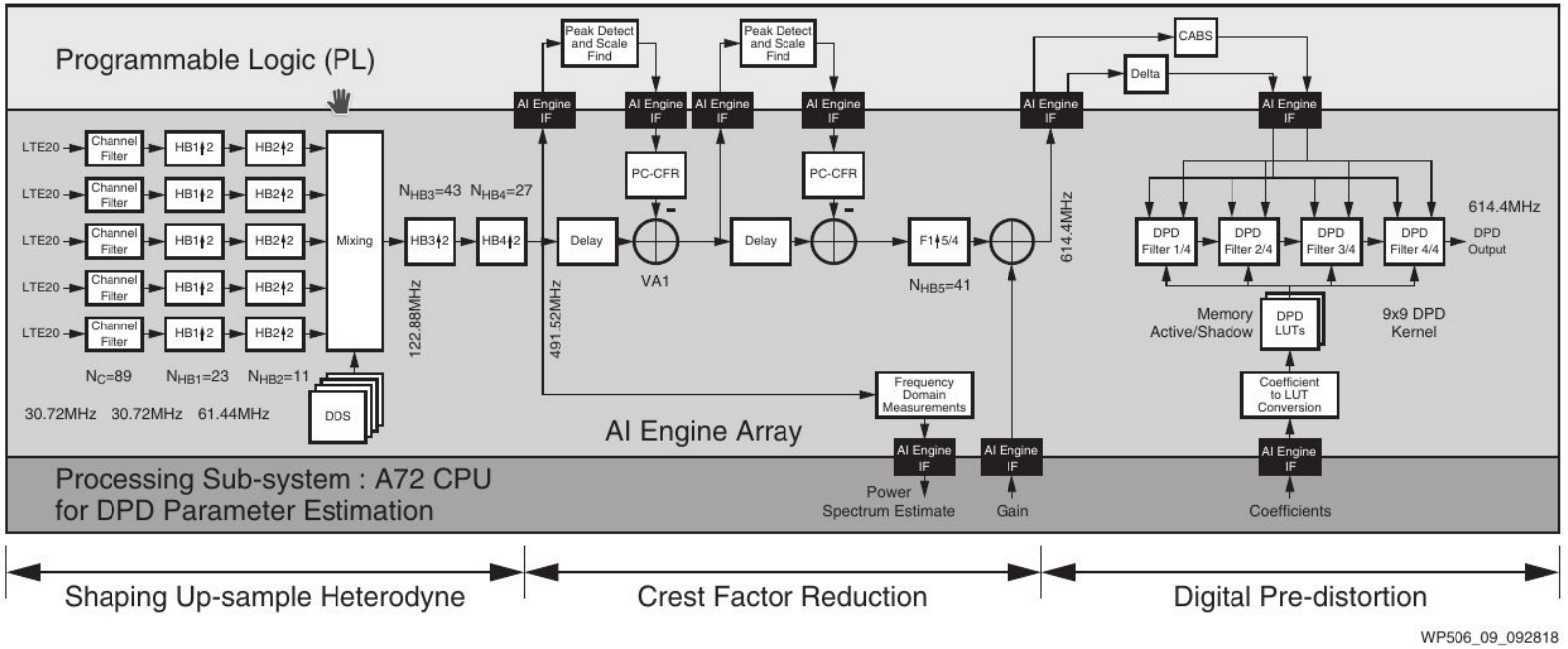


Figure 9: **Block Diagram: 100MHz 5-Channel LTE20 Wireless Solution with DSP**

VHDL vs. System Verilog (1 of 3)

- The only RTLs/HDLs of any relevance: VHDL and System Verilog -- -- from the 1980s until now
- RTL => Register Transfer Language
- VHDL and System Verilog are used for two tasks:
  - Synthesizable Code
  - Testbench Code

VHDL vs. System Verilog (2 of 3)

- Synthesizable Code is the code that gets converted into your FPGA “executable”
  - VHDL/System Verilog => synthesis process=> Netlist (logic functions and interconnect)
  - Netlist => Placement and routing (PAR) process => FPGA “executable” (see FPGA PAR slide)
  - Tools for Synthesis + Place and Route:
    - Intel (Altera): Quartus Prime/Quartus Pro
    - Xilinx: Vivado
- Testbench Code is used to test your Synthesizable code before you run the FPGA “executable” in your FPGA (and have to find your bugs the hard way)
  - Simulators: Mentor Graphics Questa, Synopsys VCS, etc.
  - Full-chip testbench => simulates the board/ICs connected to your FPGA
  - Module testbench => simulates the inputs and verifies outputs of a given RTL module
    - Full-chip testbench is only for sanity checking in large designs. The real work is done in the module level testbenches.

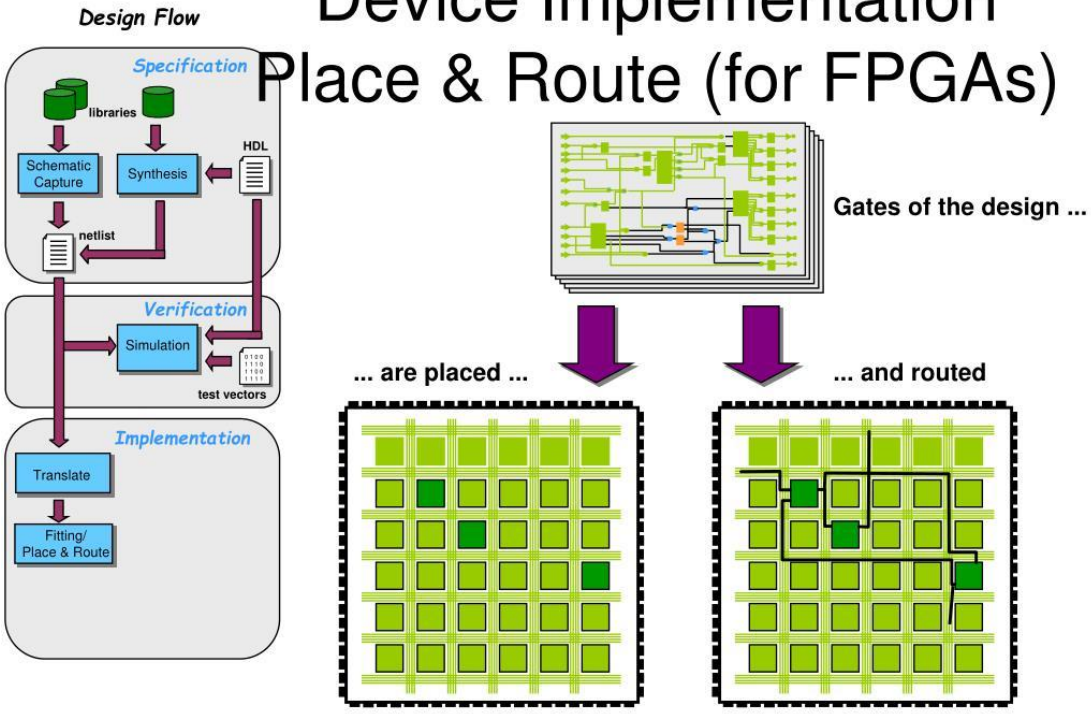
VHDL vs. System Verilog (3 of 3)

- VHDL was developed by the Department of Defense. The syntax is similar to Ada.
- Verilog (predecessor to System Verilog) was developed in academia and has syntax similar to C.
- VHDL has very rigid syntax and steeper learning curve.
- System Verilog has easier learning curve, but some subtle issues can slip through that are caught (as errors) in VHDL. In System Verilog, a comprehensive testbench is often required to catch such issues.
    - Fortunately, most of these issues can be caught by being rigorous about checking and fixing the warnings from your simulator (e.g., Mentor Graphics Questa) and synthesis/place and route tool (e.g., Vivado/Quartus).

FPGA Place and Route

Device Implementation

Place & Route (for FPGAs)



UNM XILINX Los Alamos NATIONAL LABORATORY Sandia National Laboratories SES Consultants, Inc.

FPGA Vendors

- Xilinx and Intel (formerly Altera) are the two “eight hundred pound gorillas” of the FPGA market.
- Intel bought Altera (2015) to develop/promote hybrid processor/FPGA applications
  - Hence integrating Intel FPGAs with Xeon Processors using CCIX cache-coherent interconnect and Intel Optane memory is highly promoted by Intel.
- Xilinx invented FPGA
- Xilinx is emphasizing using FPGAs in cloud computing/machine learning/AI with the Alveo series of plug-in FPGA cards, the Versal FPGA (ACAP) family, and the Vitis software platform
  - Vitis allows for programming FPGAs using C instead of VHDL/System Verilog -- but not as high performance as well written VHDL/System Verilog

Alternatives to RTLs

- Allows for software programmers to harness the power of FPGAs without learning RTLs (somewhat…)
  - LabVIEW FPGA has been around for 17 years -- but these approaches typically give less performance for a given amount of FPGA resources utilized
- Xilinx Vitis
  - https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html
- OpenCL (Intel FPGA supports)
  - https://www.intel.com/content/www/us/en/software/programmable/sdk-for-opencl/overview.html?wapkw=OpenCL
- LabVIEW FPGA
  - https://www.ni.com/en-us/shop/electronic-test-instrumentation/add-ons-for-electronic-test-and-instrumentation/what-is-labview-fpga-module.html

Getting Started

- Recommended books/resources
  - http://ati.ttu.ee/~alsu/fpgas_for_dummies_ebook.pdf
  - https://www.oreilly.com/library/view/design-recipes-for/9780080971360/
  - https://www.amazon.com/Rapid-Prototyping-Digital-Systems-SOPC/dp/0387726705
  - Coursera
    - https://www.coursera.org/courses?query=fpga
    - https://www.coursera.org/learn/intro-fpga-design-embedded-systems
- Cheap(ish) FPGA Eval Boards
  - https://www.terasic.com.tw/en/
    - https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=163#Category165
    - https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=204#Category205

Thank you

Questions?