REAL-TIME DIGITAL SYSTEMS DESIGN AND VERIFICATION WITH FPGAS
ECE 387 – LECTURE 1

PROF. DAVID ZARETSKY

DAVID.ZARETSKY@NORTHWESTERN.EDU

AGENDA

- Introduction
- Administrative
- High-level Design Approach

COURSE OBJECTIVE

WHY TAKE THIS CLASS?

- Hands-on experience with advanced hardware design
  - Learn how to design, test, and implement complex digital systems using FPGAs, preparing you for real-world applications.
- Industry-standard verification tools
  - Gain expertise in SystemVerilog + UVM methodologies widely adopted in the semiconductor industry for design & verification.
- In-depth understanding of cutting-edge technologies
  - Explore how FPGAs are used in emerging fields like AI, IoT, computer vision, communications, and automotive systems.
- Critical skills for high-demand careers
  - Master skills essential for high-demand roles in hardware design, verification engineering, and embedded systems development.
- Accelerated learning with practical, project-based coursework
  - Work on projects that simulate real-life design and verification challenges, equipping you with the practical skills employers seek.

INTRODUCTION

- This class will take you through the full design cycle of a commercial FPGA design project.
- The emphasis is on top-down design approach: Software Model → HDL Model
- You will learn the keys to building real-world FPGA designs from a commercial perspective.
  - Architecture & planning
  - Design strategy & optimizations
  - Simulation & benchmarking
  - Synthesis & performance analysis
- The course will focus on design for high-performance computing applications using streaming architectures.
- Students will gain experience writing hardware designs in RTL and optimizing them through various techniques.

COURSE OUTCOMES

- When a student completes this course, s/he should be able to:
  - Understand the basic strategies for hardware design using SystemVerilog
  - Differentiate SystemVerilog code for hardware simulation and synthesis
  - Build a complete testbench to simulate and validate their SystemVerilog designs
  - Apply Universal Verification Methodology (UVM) for digital hardware verification
  - Use commercial CAD tools to simulate and synthesize SystemVerilog designs
  - Apply parallel processing, pipelining and streaming design methodologies to drive high-throughput performance
  - Analyze and solve timing related problems based on synthesis timing reports
  - Demonstrate a working RTL design with all aspects in the projects

CLASS INFO

- Prerequisites
  - CE 303, 355, or equivalent HDL experience
  - Or Graduate standing
- Class Meeting
  - Mon / Wed 10:00-11:20am, Tech L251
- Canvas
  - Class info / documents
  - Communications
  - Assignment upload
- Contact me
  - david.zaretsky@northwestern.edu
- Office Hours
  - Mon / Wed
  - By appointment only via Calendly
  - L360 or Zoom

COURSE SCHEDULE

Week	Topic	Lab
Week 1	Introduction to FPGA Design & System Verilog	Fibonacci
Week 2	Introduction to Processes & FSMs, Hierarchical designs, and Verification	Matrix Multiplication
Week 3	Streaming Architectures	Motion Detection
Week 4	Universal Verification Methodology (UVM)	Edge Detection
Week 5	Hardware Optimizations	UDP
Week 6	Fixed-point Architectures & Quantization	Cordic
Week 7	Digital Signal Processing	FFT
Week 8-10	Final Project	FM Radio

GRADING

- Weekly Programming Assignments
  - Weekly Programming Assignments: 7 x 10%
  - Class Participation: 10%
  - Final Project: 20%
- Assignments due by Wed @ start of class.
- Late submissions subject to 10% deducted points per day, up to 3 days

ASSIGNMENTS

- Labs
  - Submit code on Canvas
  - Simulate design in Questa
  - Synthesize design in Synplify Premier
  - We will not be implementing designs on FPGA boards
- Final project
  - A comprehensive streaming architecture utilizing all of the techniques and optimizations covered in class
  - Simulation of major code modules
  - Synthesis and performance analysis
  - Project report

FPGA DESIGN TOOLS

- ECE lab:
  - Simulation: Mentor (Siemens EDA) Questa
  - Synthesis: Synopsys Synplify Premier
- Other Development Tools
  - Visual Studio Code
  - Cygwin (Linux emulator for Windows)
  - GCC or G++

REMOTE ACCESS TO WILKINSON SERVERS

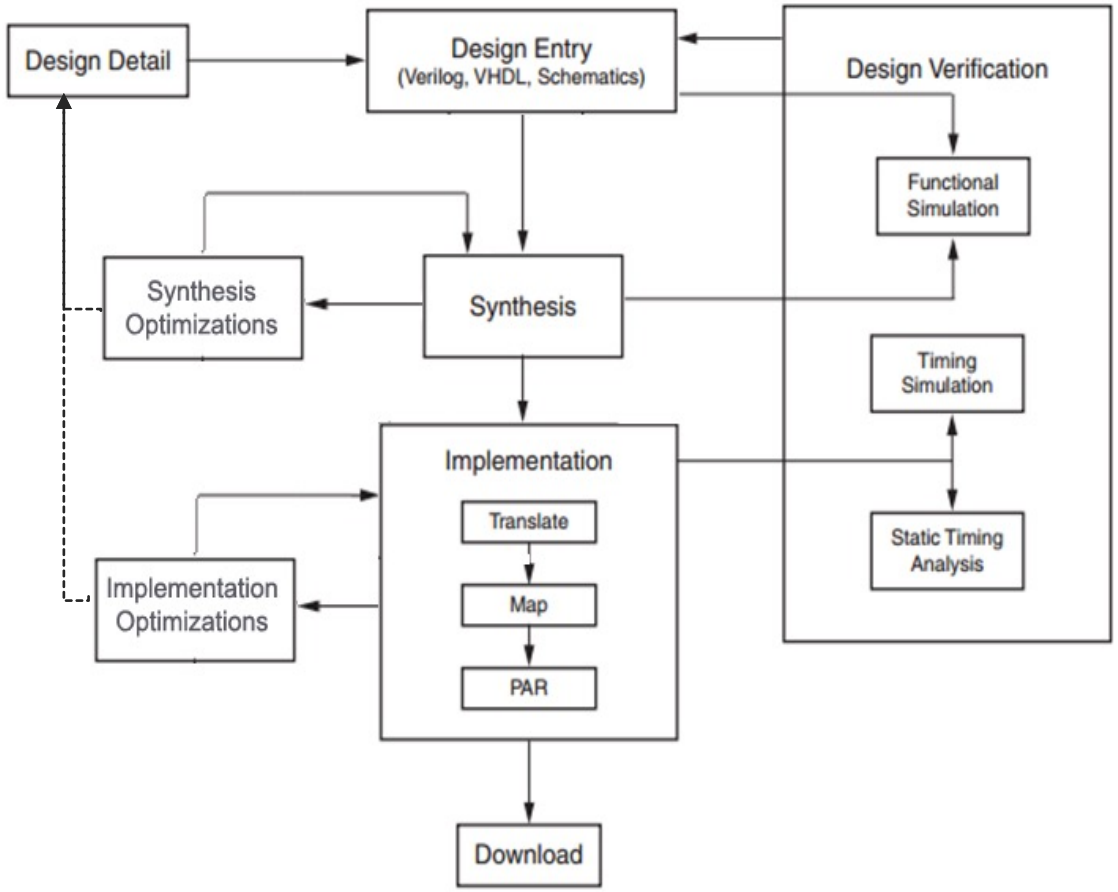
- If you are off-campus, you will need to VPN to access the servers
  - Setting up and using GlobalProtect VPN
- A list of the ECE servers can be found here:
  - http://it.eecs.northwestern.edu/info/2015/11/03/info-labs.html
  - Note some of the high-end server may be used by other classes
  - Always save your work and log out. DO NOT assume that your session will be saved when you return.
- Recommended: Access with FastX via web browser
  - In your browser, enter: https://[server].ece.northwestern.edu:3300/
  - For example, try: https://joker.ece.northwestern.edu:3300
  - Log in with your ECE credentials
  - Click Launch Session
  - Click on Gnome and then click Launch

ACCESSING ECE SOFTWARE

- To access Mentor Modelsim, type:
  - source /vol/eecs392/env/modelsim.env
  - vsim &
- To access Mentor Questa, type:
  - source /vol/eecs392/env/questasim.env
  - vsim &
- To access Synopsys Synplify Premier, type:
  - source /vol/eecs392/env/synplify.env
  - synplify_premier &

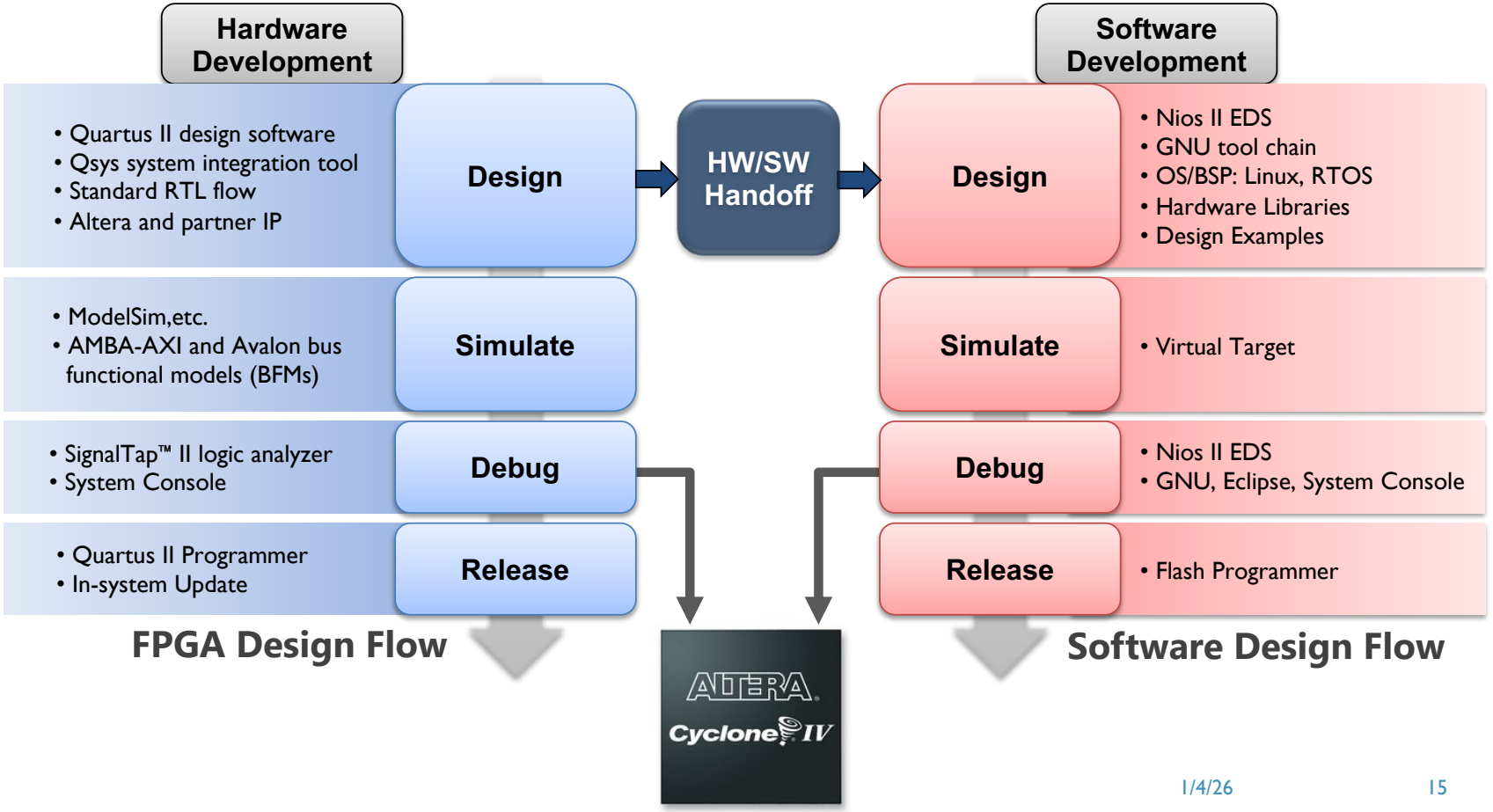
RELIABLE FPGA DESIGN PROCESS

- Reliable FPGA design require simulation at each phase of the design process.
- RTL Functional Simulation
  - Cycle-accurate simulation using RTL HDL source code.
- Gate-Level Functional Simulation
  - Simulation using a post-synthesis netlist to test the functional netlist.
- Gate-Level Timing Simulation
  - Simulation using a post-fit timing netlist, testing functional and timing performance.



SOC FPGA DESIGN PROCESS

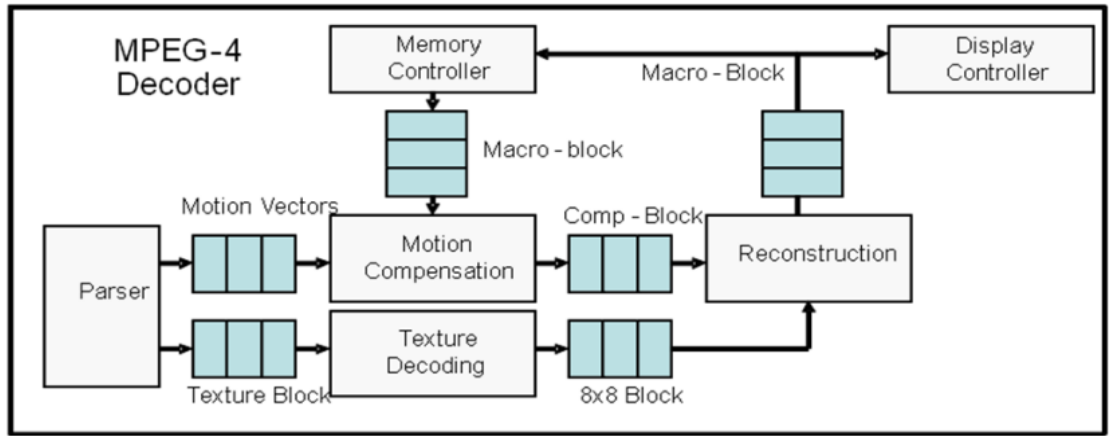
- Hardware / Software co-designs are even more complicated to verify
- Require specialized hardware-software co-simulation tools
- Mentor QuestaSim was developed to provide co-simulation.



MODELING THE HARDWARE APPLICATION

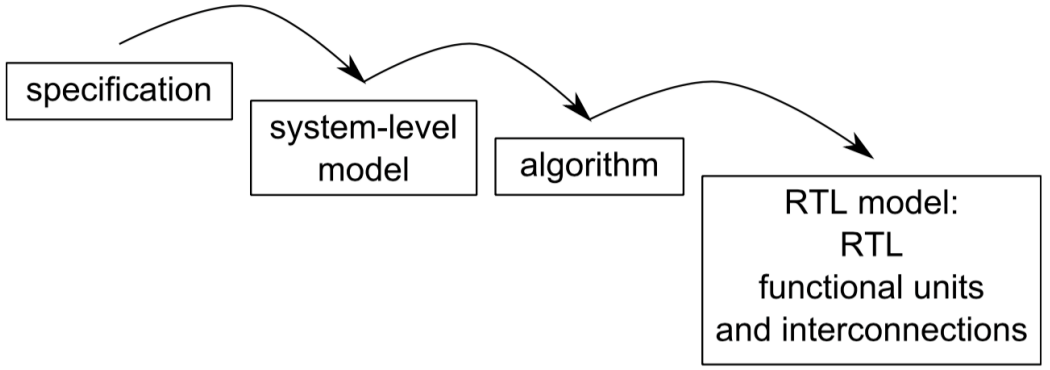
- FPGA designs are typically modeled in software first (Matlab, C/C++) to identify data paths, inputs, bottlenecks, and memory requirements.
- Profiling the software architecture allows us to model the hardware, determine user interfaces, peripherals, bandwidth and memory requirements.

```
int MPEG4_Decoder()
{
    ParserVLD( n_streams, ... );
    CopyControl( perform_cc, ... );
    MotionCompensation( perform_mc, ... );
    TextureDecoder( perform_idct, ... );
    Reconstruction( perform_rec, ... );
    DisplayControl( stnum, ... );
}
```



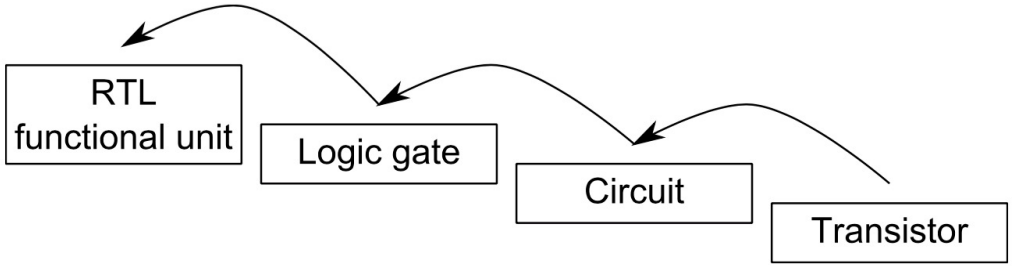
TOP-DOWN DESIGN APPROACH

- In the top-down approach the design process starts with a high-level representation of the system.
- The high-level model includes partitions (subsystems) with a specific task.
- During the design process the implementations of the subsystems are elaborated; they are split into components with more specific sub-tasks and more detailed implementations.
- The process stops when the components of the refined design are simple enough to substitute them with an existing model (practically with an RTL functional unit).



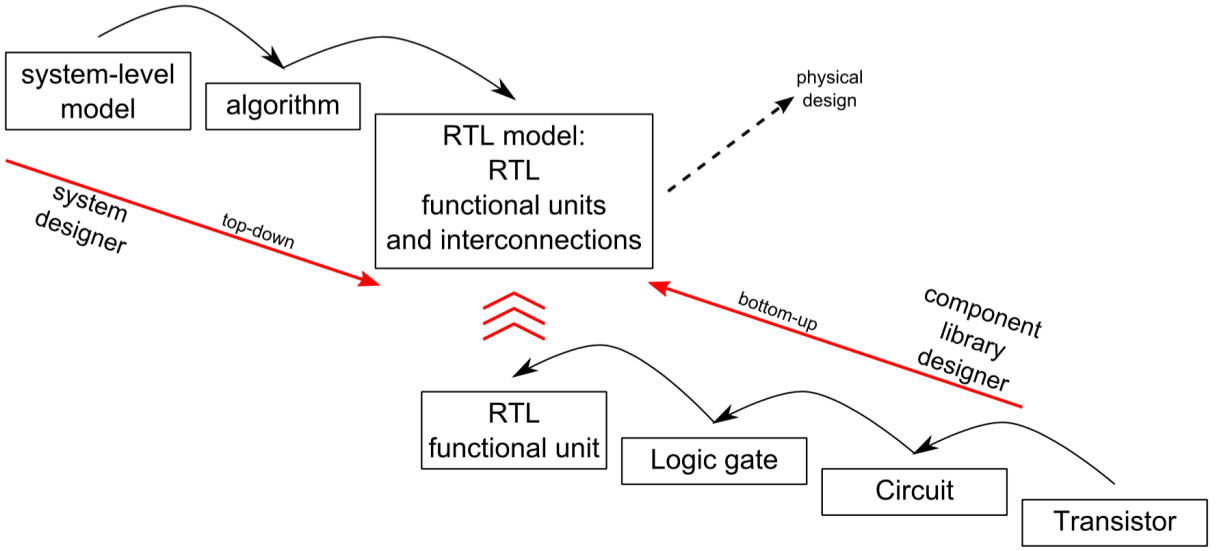
BOTTOM-UP DESIGN APPROACH

- In the bottom-up approach the designer creates basic functional units with very simple tasks.
- Once a sufficient set of elementary functionalities is constructed, a more complex model can be prepared with the combination of the simple ones.
- The design process stops when the increasingly complex model is able to implement the desired functionality defined in the specification.



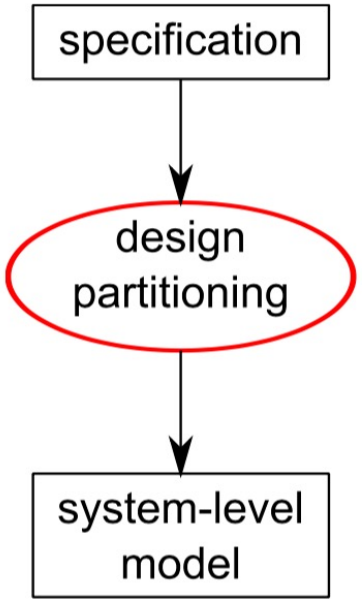
TOP-DOWN AND BOTTOM-UP METHODS IN DIGITAL DESIGN

- In the digital design the top-down and the bottom-up methods are both applied.
- The system designer creates RTL models from the high-level specification with top-down method but the standard cells are constructed from circuit-level by the component library designer with bottom-up approach.



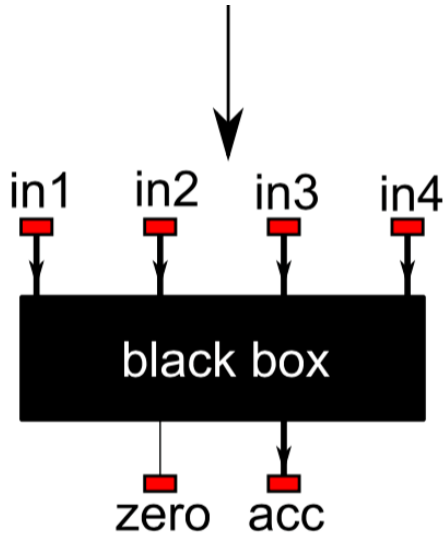
STEP #1. DESIGN PARTITIONING

- The subsystems, their relations and interfaces have to be outlined. The subsystems are represented as "black boxes".



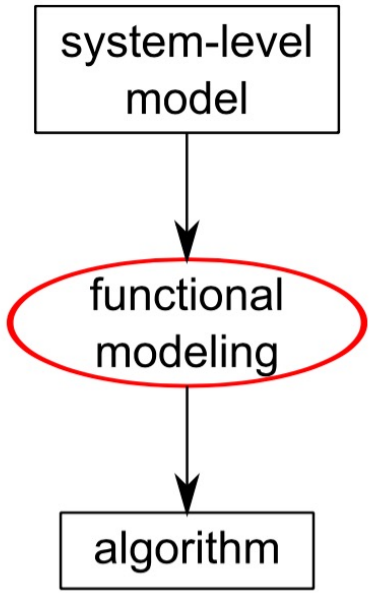
specification

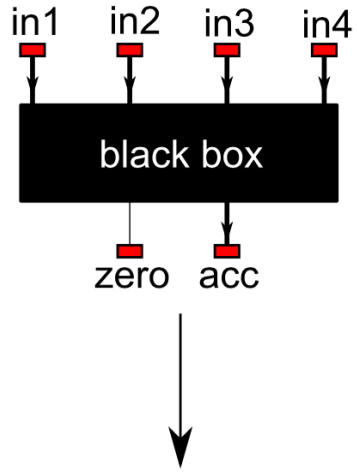
"A functional unit, which is able to calculate the sum of four integers, and to generate a zero flag"



STEP #2. FUNCTIONAL MODELING

- The behavior of the subsystems have to be formulated using software tools and high-level programming languages (C, C++, SystemC)



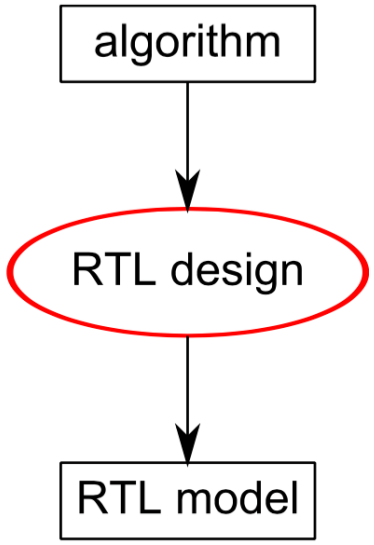


functional model (C++)

```
bool SUM(const int in1,const int in2,const int in3,const int in4,
         int& acc) {
  return !(acc = in1 + in2 + in3 + in4);
}
```

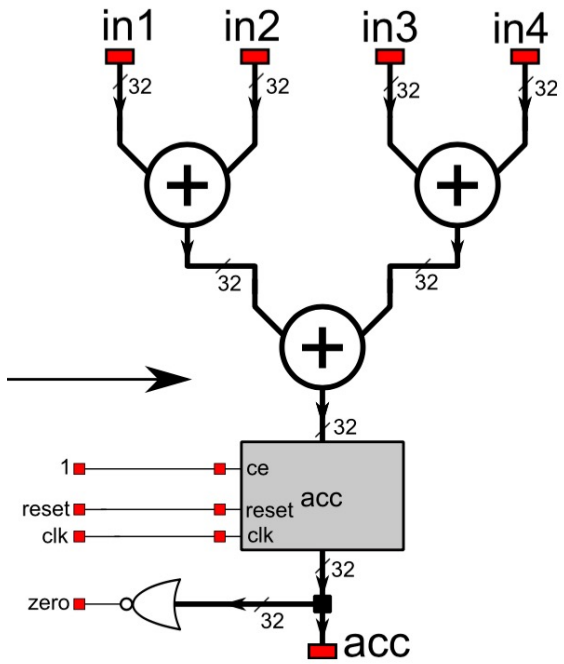
STEP #3. RTL DESIGN

- A microarchitecture consisting of simple functional units (registers, arithmetic units, etc.) has to be constructed using hardware description languages (VHDL, Verilog, SystemC)



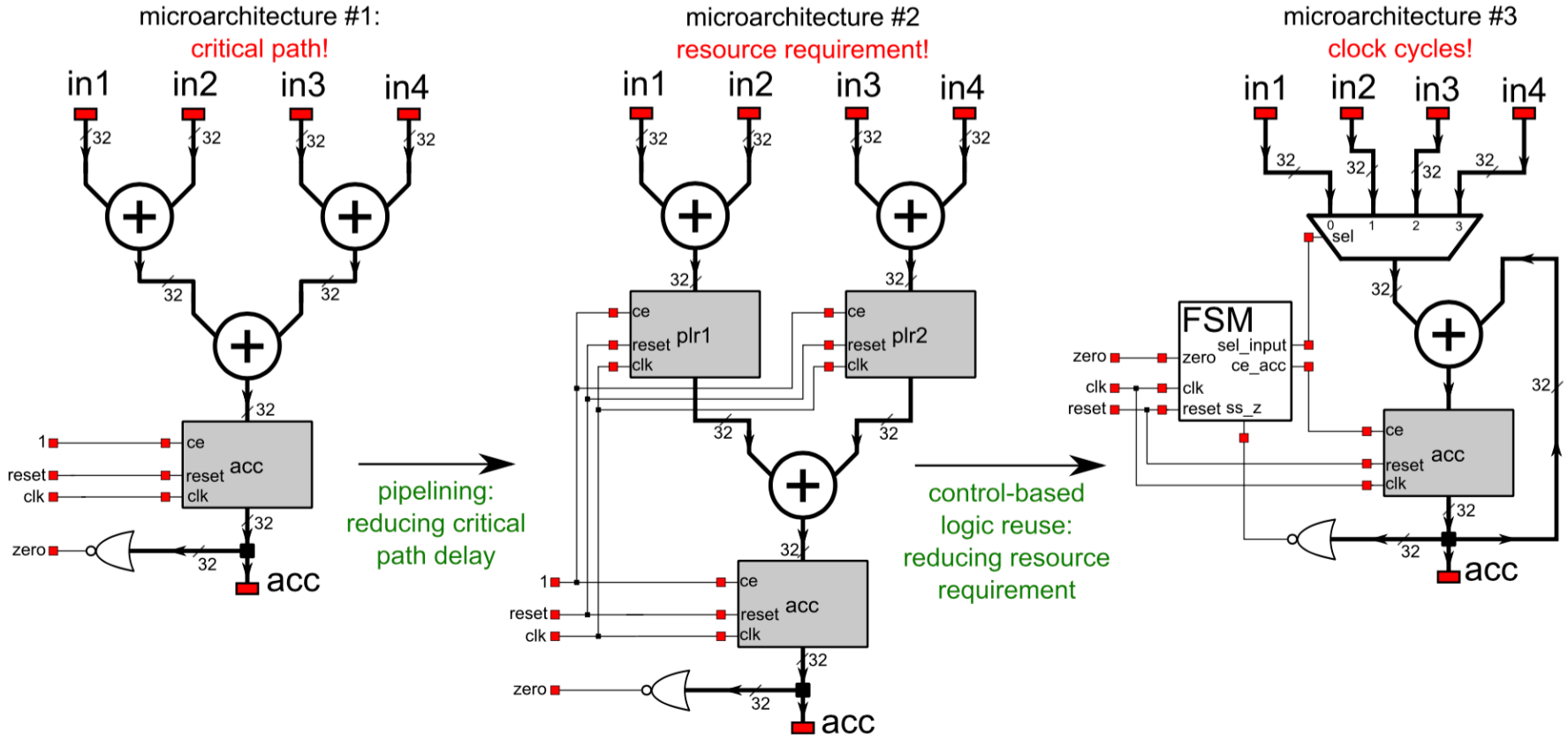
functional model

```
bool SUM(const int in1,
         const int in2,
         const int in3,
         const int in4,
         int& acc) {
  return !(acc = in1 + in2
                 + in3 + in4);
}
```



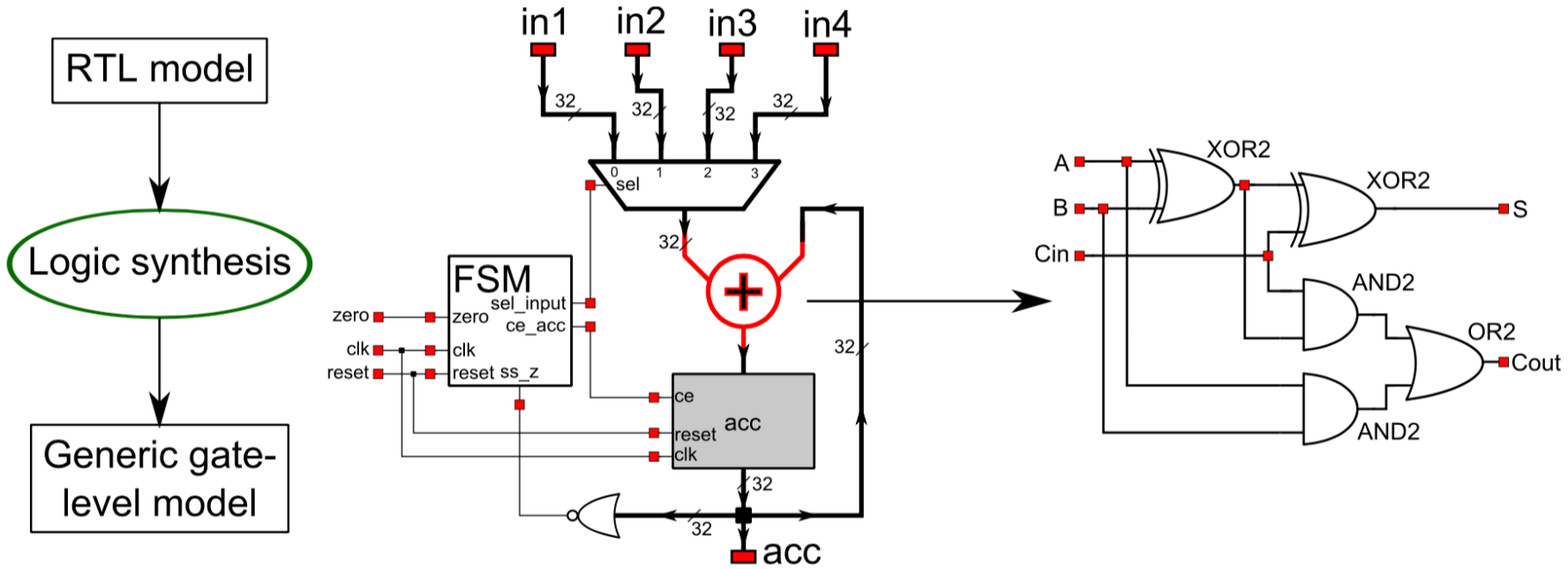
STEP #4. RTL OPTIMIZATIONS

- There are many ways of implementing microarchitectures with the same functionality. We have to choose one with parameters optimal for the application.



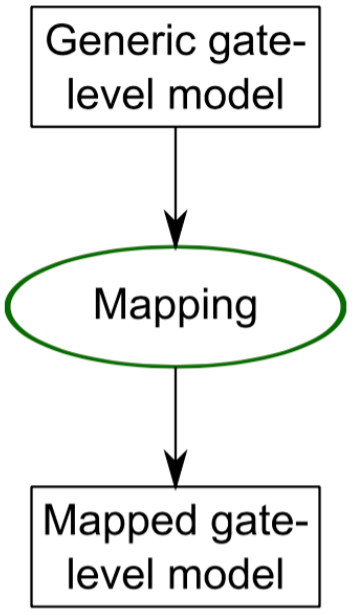
STEP #5. LOGIC SYNTHESIS

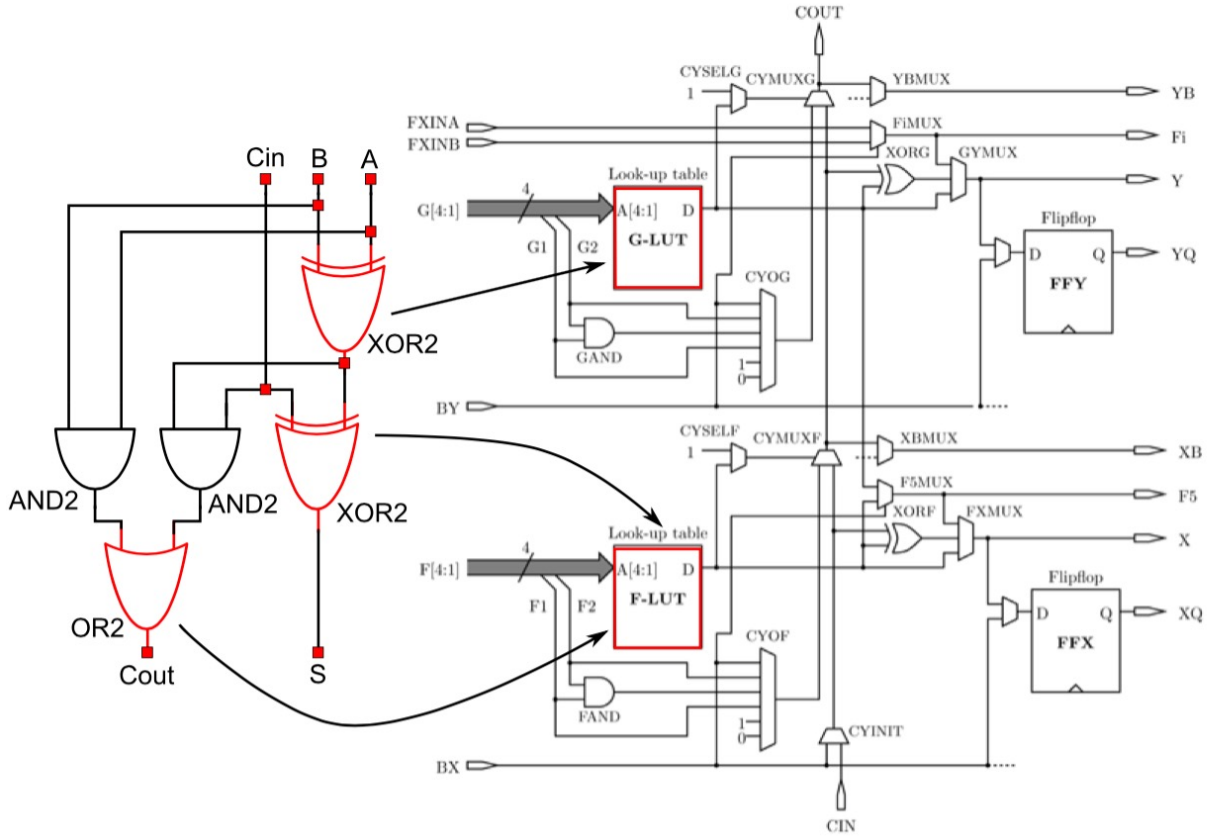
- The RTL description is automatically transformed into a technology and vendor-independent gate-level model.



STEP #6. MAPPING

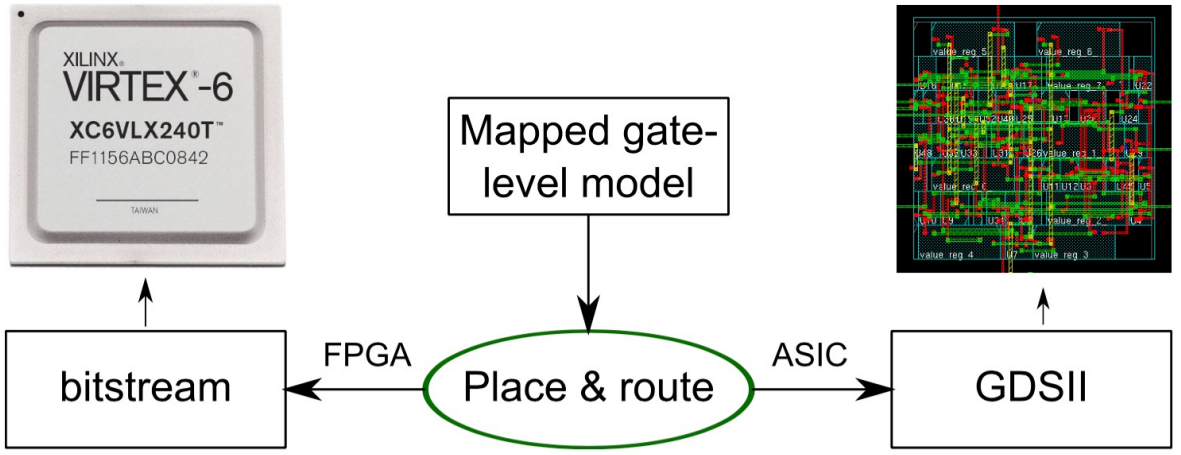
- Mapping assigns specific component library primitives to the generic resources of the gate-level model.
- These library primitives are the basic elements of an ASIC technology or an FPGA device family.





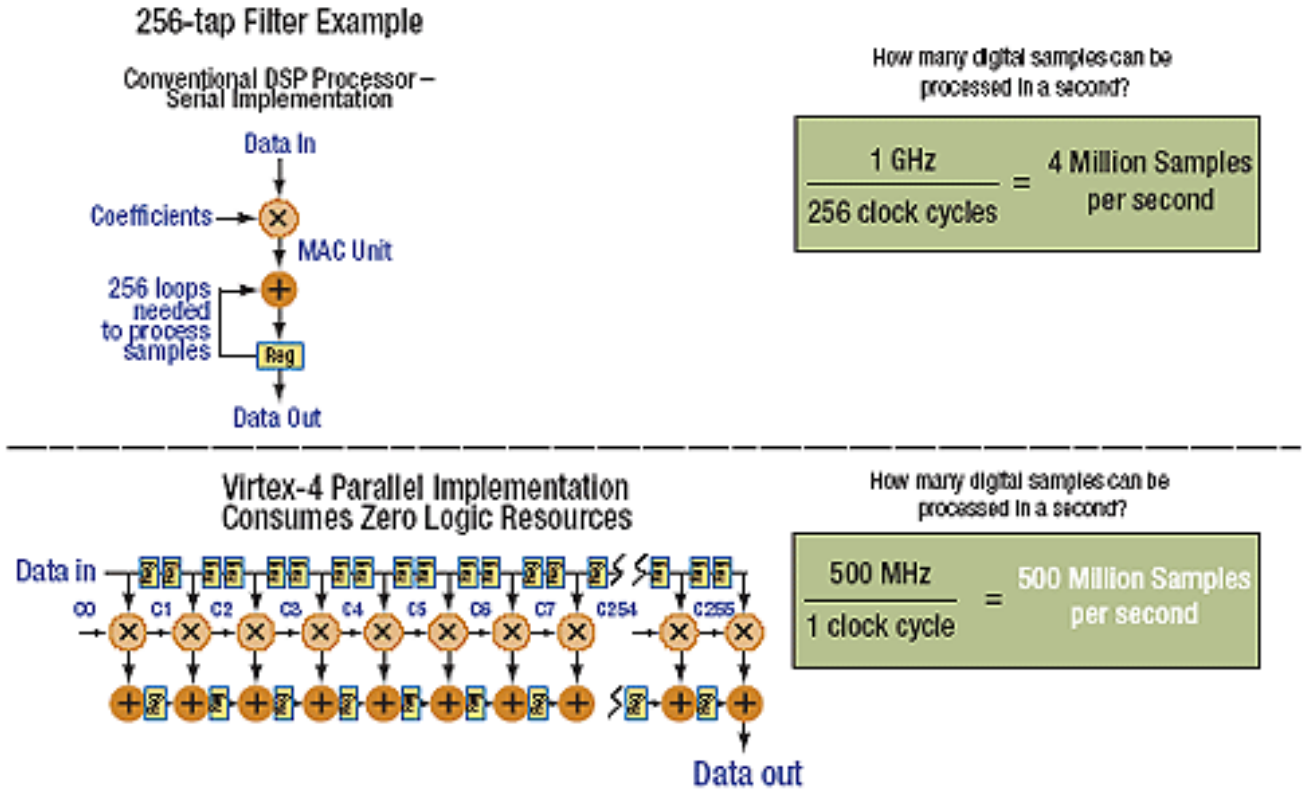
STEP #7. PLACE & ROUTE

- Place: The mapped primitives are assigned to a specific resource in a device (FPGA) or they are placed into a specific position of the chip layout (ASIC).
- Route: The interconnections of the placed primitives are constructed.
- The output of the design flow is a GDSII file (ASIC) including the information required for manufacturing or a bitstream file (FPGA) including the configuration memory content of the FPGA device.



OPTIMIZING FOR DIFFERENT PERFORMANCE GOALS

- Exploiting parallelism in FPGAs allows you to achieve performance at orders of magnitude greater than software
- For example, using hardware replication, we can increase throughput by orders of magnitude.
- A hardware designer must optimize for different performance goals, such as timing, area, and power, generally at the cost of one over others.



IMPACT OF DESIGN PERFORMANCE

- Performance and resoures affect the type, size and cost of FPGA devices.

Design Requirements

Device Resources	FPGA
LEs (K)	34
Adaptive logic modules (ALMs)	12,600
M10K memory blocks	90
M10K memory (Kb)	900
MLABs (Kb)	
18-bit x 19-bit multipliers	74
Variable-precision DSP blocks (1)	37
FPGA PLLs	2
HPS PLLs	2
Maximum FPGA user I/Os	140
Maximum HPS I/Os	130
FPGA hard memory controllers	1
HPS hard memory controllers	1
Processor cores	Single

SoC FPGA Device Selection Options

Device Resources	5CSEA2	5CSEA4	5CSEA5	5CSEA6
LEs (K)	25	40	85	110
Adaptive logic modules (ALMs)	9,434	15,094	32,075	41,509
M10K memory blocks	140	224	397	514
M10K memory (Kb)	1,400	2,240	3,972	5,140
MLABs (Kb)	138	220	480	621
18-bit x 19-bit multipliers	72	116	174	224
Variable-precision DSP blocks (1)	36	58	87	112
FPGA PLLs	4	5	6	6
HPS PLLs	3	3	3	3
Maximum FPGA user I/Os	145	145	288	288
Maximum HPS I/Os	188	188	188	188
FPGA hard memory controllers	1	1	1	1
HPS hard memory controllers	1	1	1	1
Processor cores	Single/Dual	Single/Dual	Single/Dual	Single/Dual

NEXT...

- Review key concepts in SystemVerilog programming