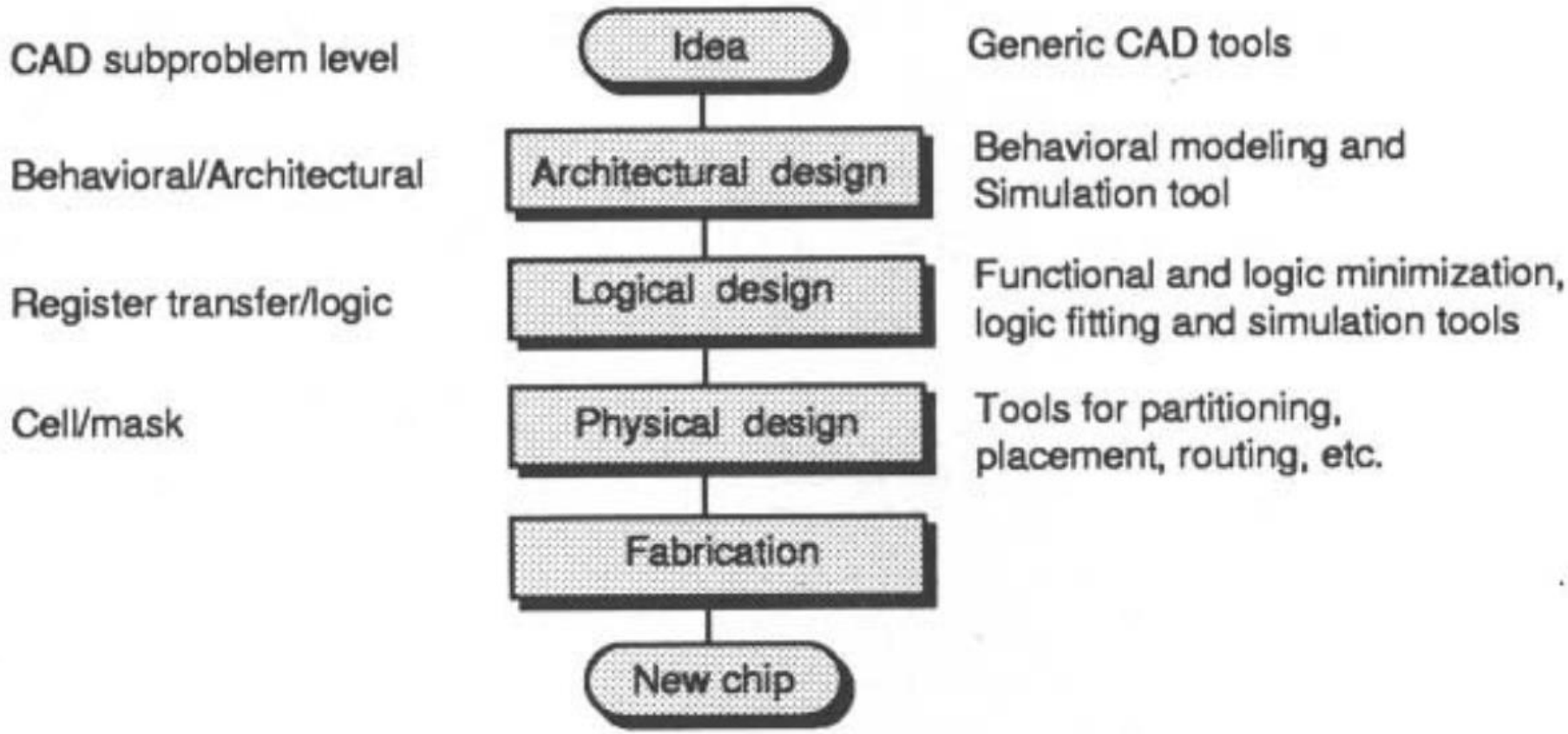
ASIC Design using Verilog/VHDL

(EC9036)

Agenda

- Welcome & Course Identity
- VLSI Design Flow
- Front-end vs Back-end Design
- System Design Concepts
- Our Journey……
- Verification scope
- Books

VLSI Design Process



Front-end (Logical Design):

- RTL coding (Verilog/VHDL/SystemVerilog)
- Functional verification (simulation)
- Synthesis (mapping RTL → gates)
- Pre-layout STA

Back-end (Physical Design):

- Floorplanning
- Placement & Routing
- Clock tree synthesis
- Post-layout STA
- Tape-out (GDSII)

Key point: In this course, we focus **mainly on front-end** but will connect concepts to back-end.

- **Meaning of "System Design"**:

  Breaking a digital system into:

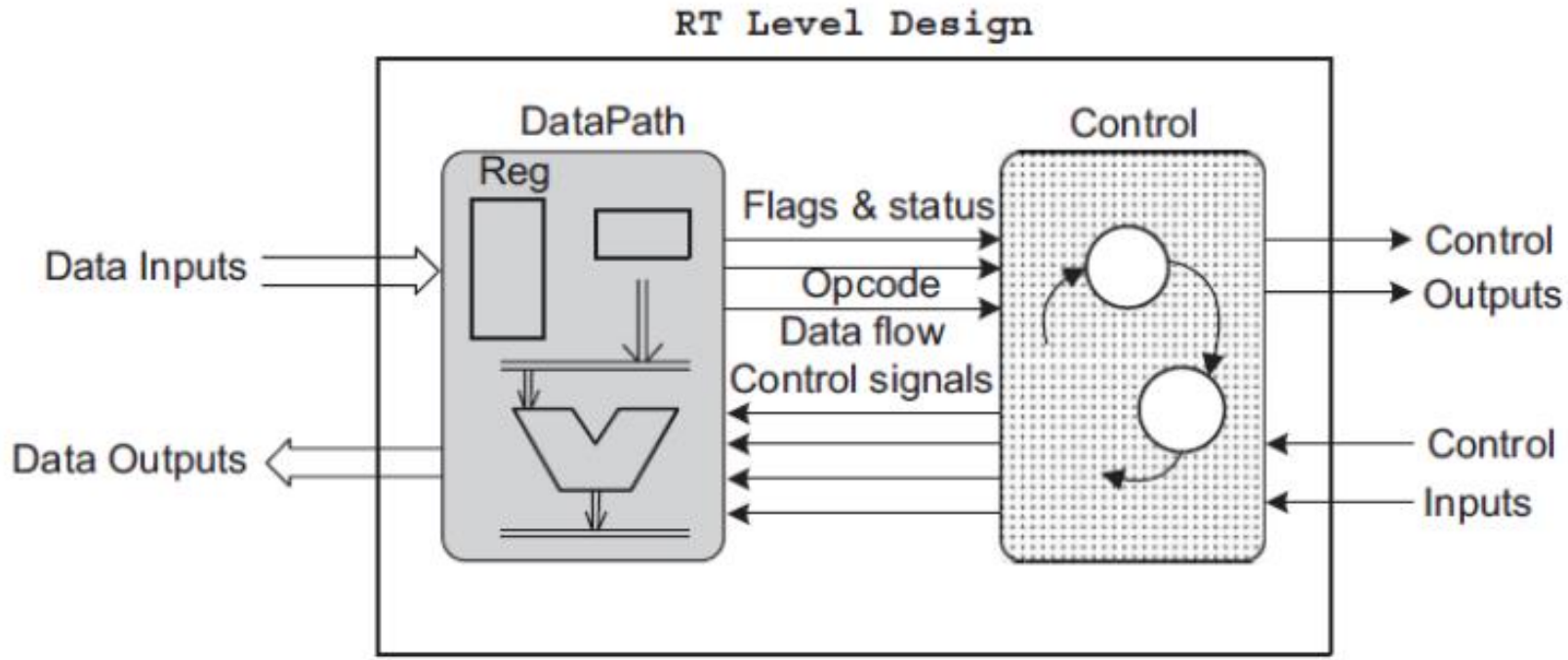
  1. **Datapath** → Operations, storage, computation

     (e.g., ALUs, registers, multipliers)

  2. **Controller** → Decision-making logic

     (e.g., FSM controlling when to load, compute, or output data)

- **Example:** FIR filter =
  - Datapath: multipliers + adders + registers
  - Controller: sequencing coefficients,

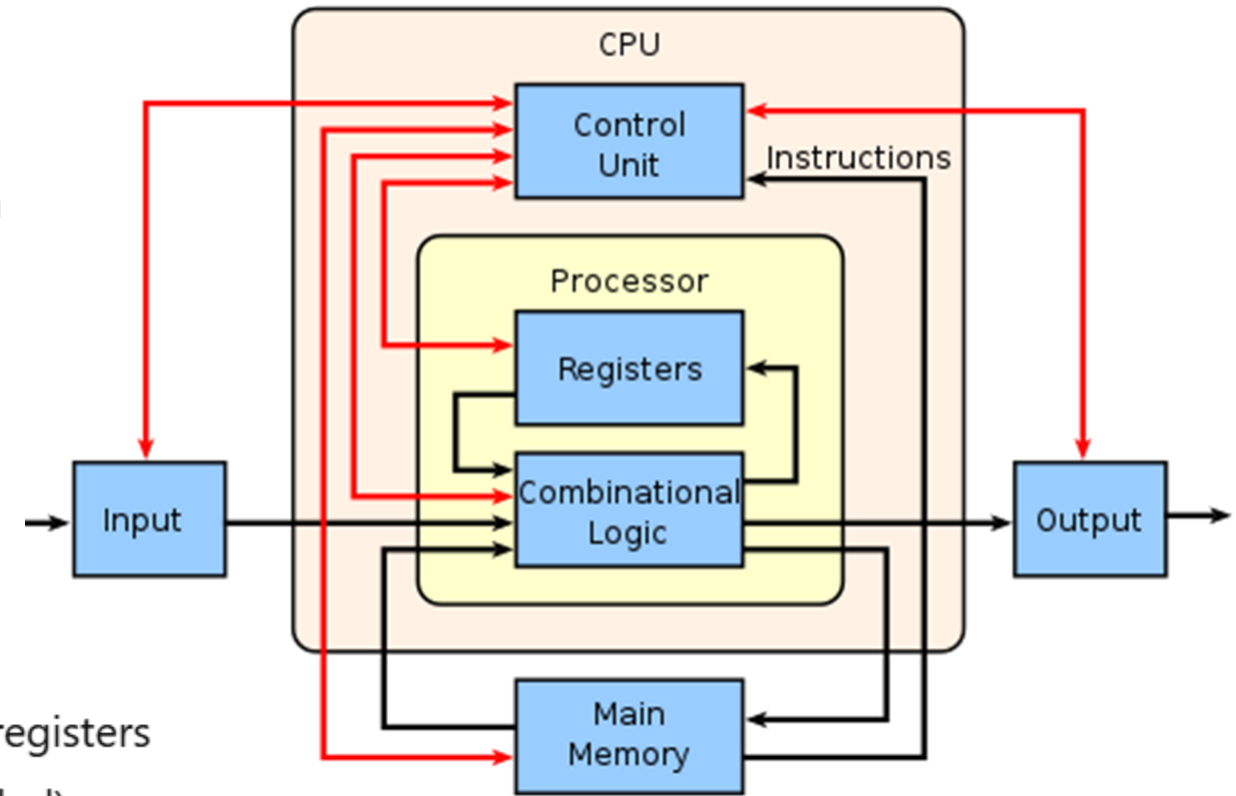


Example: system design

You will build a tiny 8-bit computer that can run a few basic instructions — load numbers, add or subtract them, store results, and even jump to different parts of a program. Your processor will have its own memory, calculator (ALU), and a brain (control unit) that fetches, decodes, and executes each instruction step-by-step, just like a real CPU but on a smaller scale.

Datapath Components:

- **PC (Program Counter):** Holds address of current instruction
- **Instruction Register (IR):** Holds fetched instruction
- **Accumulator (ACC):** Stores computation results
- **ALU:** Performs ADD/SUB and passes through data
- **Memory Interface:** ROM for program, RAM for data
- **MUXes:** For selecting ALU inputs and next PC



Controller (FSM):

States:

1. **FETCH**: Load instruction from ROM into IR
2. **DECODE**: Decode opcode, set control signals
3. **EXECUTE**: Perform ALU/memory operation, update registers
4. **WRITEBACK**: Store result to ACC or memory (if needed)
5. **PC_UPDATE**: Increment PC or jump

Our Journey Will Be Project-Based

We will learn through **five progressively complex projects**:

1. **8-bit MAC system** (with and without memory interface)
2. **Cryptographic system** (e.g., AES, simplified DES)
3. **FIR filter**
4. **Euclidean GCD algorithm**
5. **Mini Processor design**

HDL → RTL level description

- RTL (Register Transfer Level) describes hardware **cycle-by-cycle**:
  - How data flows between registers
  - How combinational logic transforms it in between
- It's **synthesizable** — can be turned into actual gates, LUTs, FFs, DSP blocks.
- Writing RTL ensures **precise control** over hardware behavior, timing, and resource usage.
- Industry-standard HDLs for RTL: **Verilog, VHDL, SystemVerilog**.

Verification

- **SystemVerilog** = Enhanced Verilog with:
  - Better RTL features (interfaces, always_comb, always_ff)
  - Object-oriented verification features
- **UVM (Universal Verification Methodology)**:
  - Industry-standard for ASIC functional verification
  - Covers constrained random testing, functional coverage, reusable testbenches
- Even if this course is RTL-focused, knowing UVM scope prepares you for **verification engineer** roles.

Prerequisites for the Course

Digital Logic Fundamentals

- **Combinational logic**: AND, OR, NOT, XOR, multiplexers, decoders, encoders
- **Sequential logic**: Flip-flops, latches, registers, counters
- **Finite State Machines (FSM)**: Mealy vs Moore, state diagrams, state encoding
- **Number systems**: Binary, hexadecimal, 2’s complement, signed/unsigned arithmetic

ASIC vs. FPGA

- **Front-end stages (Spec → RTL → Verification → Synthesis)** are **almost identical** for ASIC and FPGA.
- **Back-end stages** differ:
  - ASIC: deep physical design, fabrication, mask generation, DRC/LVS, long lead time.
  - FPGA: placement/routing in fixed fabric, bitstream generation, instant re-programming.
- FPGA flow is often used as a **prototype stage** before committing to ASIC.
- Concepts like **timing closure**, **power vs area trade-offs**, and **resource constraints** exist in both — just different resource types (LUTs/DSPs vs std cells).

Books

1. **Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog**
   M. Morris Mano & Michael D. Ciletti — **Fundamental digital logic concepts + HDL basics**
2. **Verilog HDL: A Guide to Digital Design and Synthesis**
   Samir Palnitkar — **Clear Verilog explanations and examples**
3. **SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling**
   Stuart Sutherland, Simon Davidmann, Peter Flake — **Best practices for RTL in SystemVerilog**
4. **Circuit Design with VHDL** *(formerly "VHDL by Example")*
   Volnei A. Pedroni — **Practical, example-driven VHDL design**
5. **Embedded Core Design with FPGAs** *(or "Embedded Core Design and Verification")*
   Zainalabedin Navabi — **Bridges embedded system design concepts with FPGA/ASIC implementation**