VLSI DESIGN FLOW: RTL TO GDS

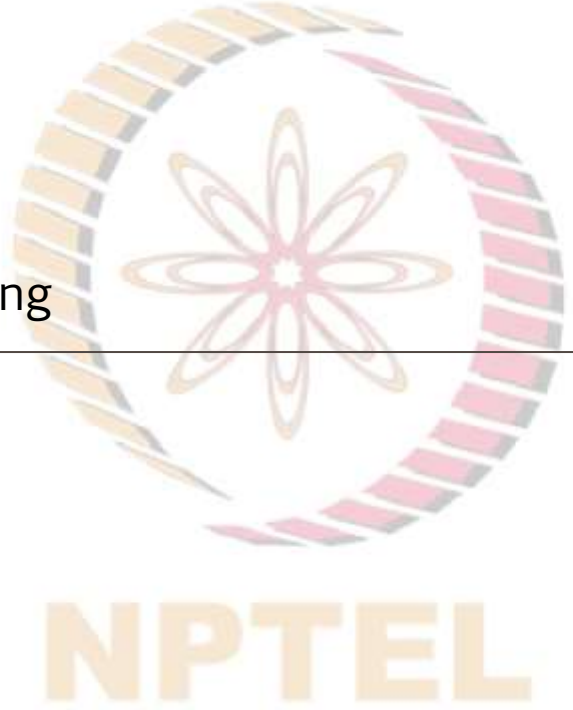
Lecture 3
Overview of VLSI Design Flow: I

Sneh Saurabh
Electronics and Communications Engineering
IIIT Delhi

Lecture Plan

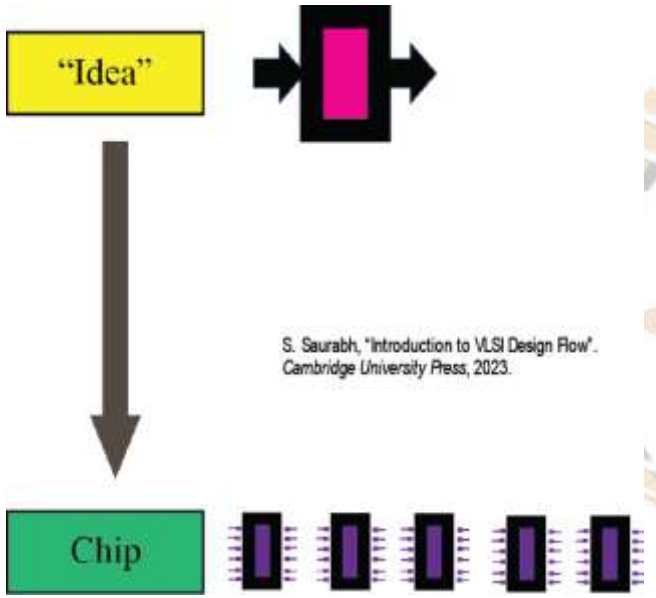
Overview of VLSI Design Flow

- Design Flows
- Abstraction
- Pre-RTL Methodologies
- Hardware—software partitioning



VLSI Design Flow: A top-level perspective

Chip Designing: Input and Output

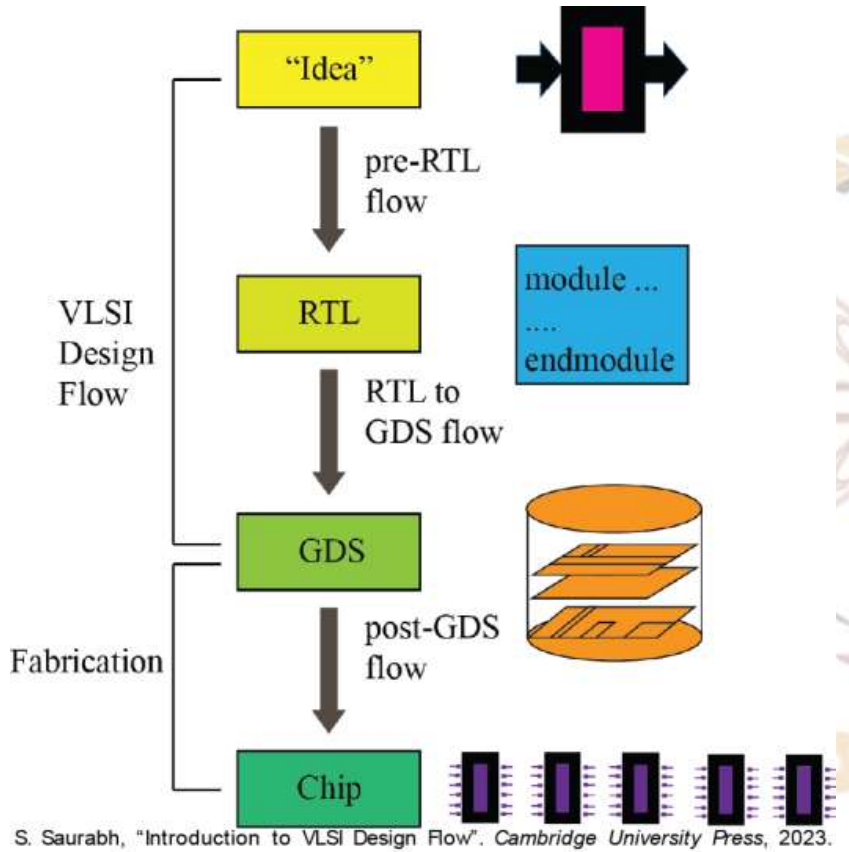


S. Saurabh, "Introduction to VLSI Design Flow", *Cambridge University Press*, 2023.

How to tackle this problem?

Divide and conquer

VLSI Design Flow: Divide and Conquer



RTL: Register Transfer Level (Verilog, VHDL)

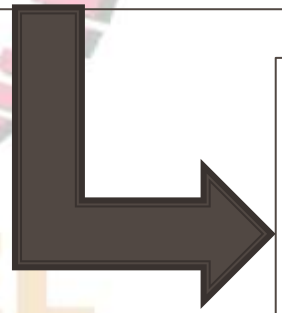
GDS: Graphical Database System (Layout)

Idea to RTL Flow: takes a high-level idea/concept of a product and represents the hardware portion of the implementation in RTL.

RTL to GDS flow: takes an RTL through various stages of logical and physical design steps and finally represents the design as GDS.

GDS to Chip Processes: takes a GDS, prepares masks for a given GDS and fabricates/tests/packages chips

VLSI Design Flow: A top-level perspective



Abstraction

Abstraction: Basic Concept

Abstraction:

- Hiding lower level details in a description
- As a design moves through VLSI design flow:
  - Details are added
  - Abstraction decreases

	Level of Abstraction	Representation
Idea to RTL flow	Very high	System, Behavior
RTL to GDS flow	Decreases subsequently down the flow	RTL, Gate, Transistor, Layout
GDS to chip	No abstraction, actual implementation	Mask, Integrated Circuit.

Why to Abstract?

Considerations for design tasks:

- **Optimization**: Choosing right combination of design parameters to obtain desired QoR by trading-off some of them.
- **Turn-around Time**: Time taken to make changes in a design

Impact of Abstraction:

- At higher level of abstraction large number of solutions can be analyzed in less amount of time.
- Result of optimization at the higher level of abstraction is expected to be better.

	Scope of Optimization	Turn-around time
Idea to RTL Flow	Very High	Low
RTL to GDS flow	Decreases subsequently down the flow	Increases subsequently down the flow
GDS to Chip flow	No optimization. Some corrections.	Very costly re-spin

Abstraction: Illustration

Consider that the functionality is represented in two ways:

A. Logic Formula: $F = (A + B)'$

B. Using a standard cell delivering NOR function placed and connected on the layout.

Which of the above representations:

1. Has greater abstraction?

Ans: A

2. Smaller turn-around time in evaluating different implementations?

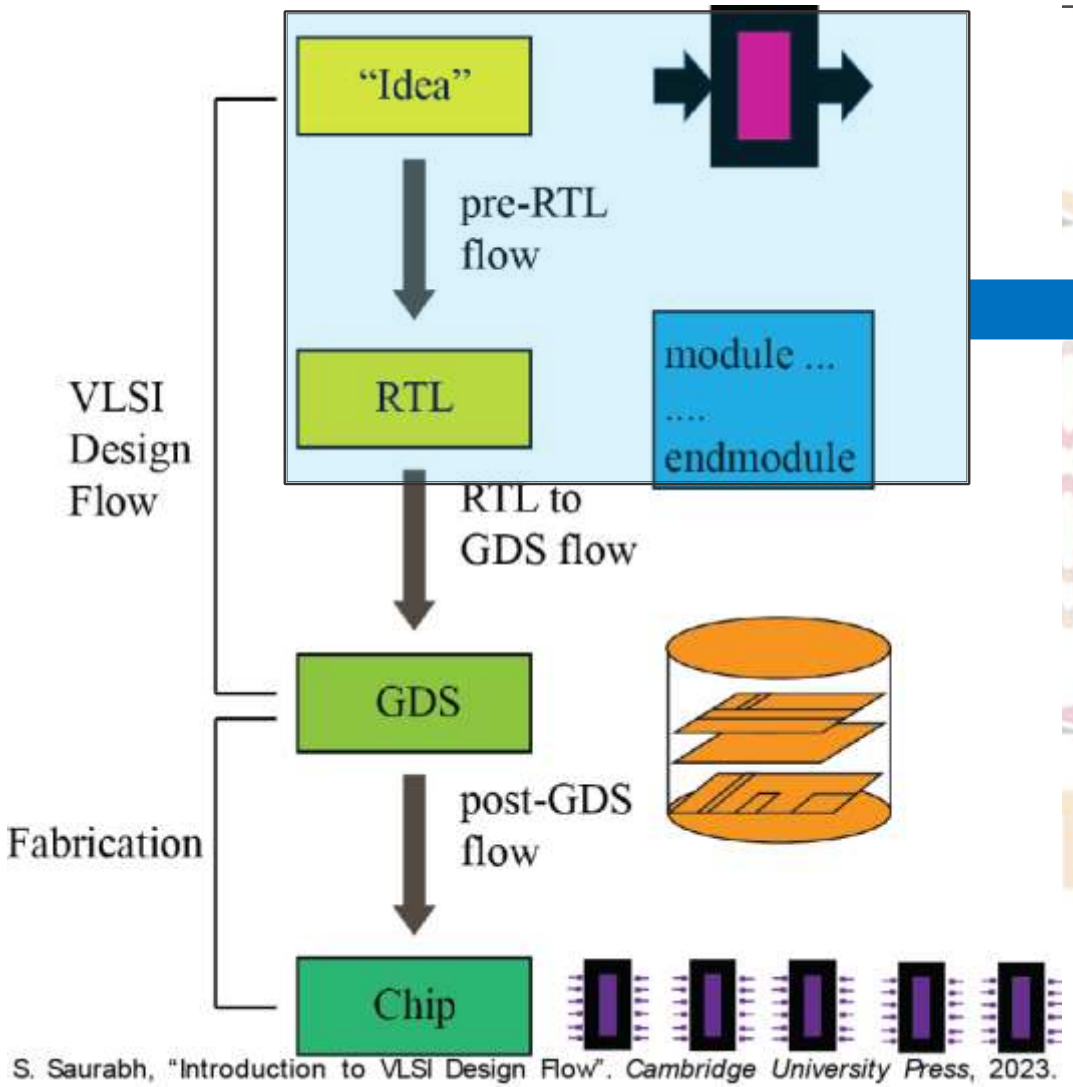
Ans: A

3. Greater accuracy in evaluation?

Ans: B

Pre-RTL Methodologies

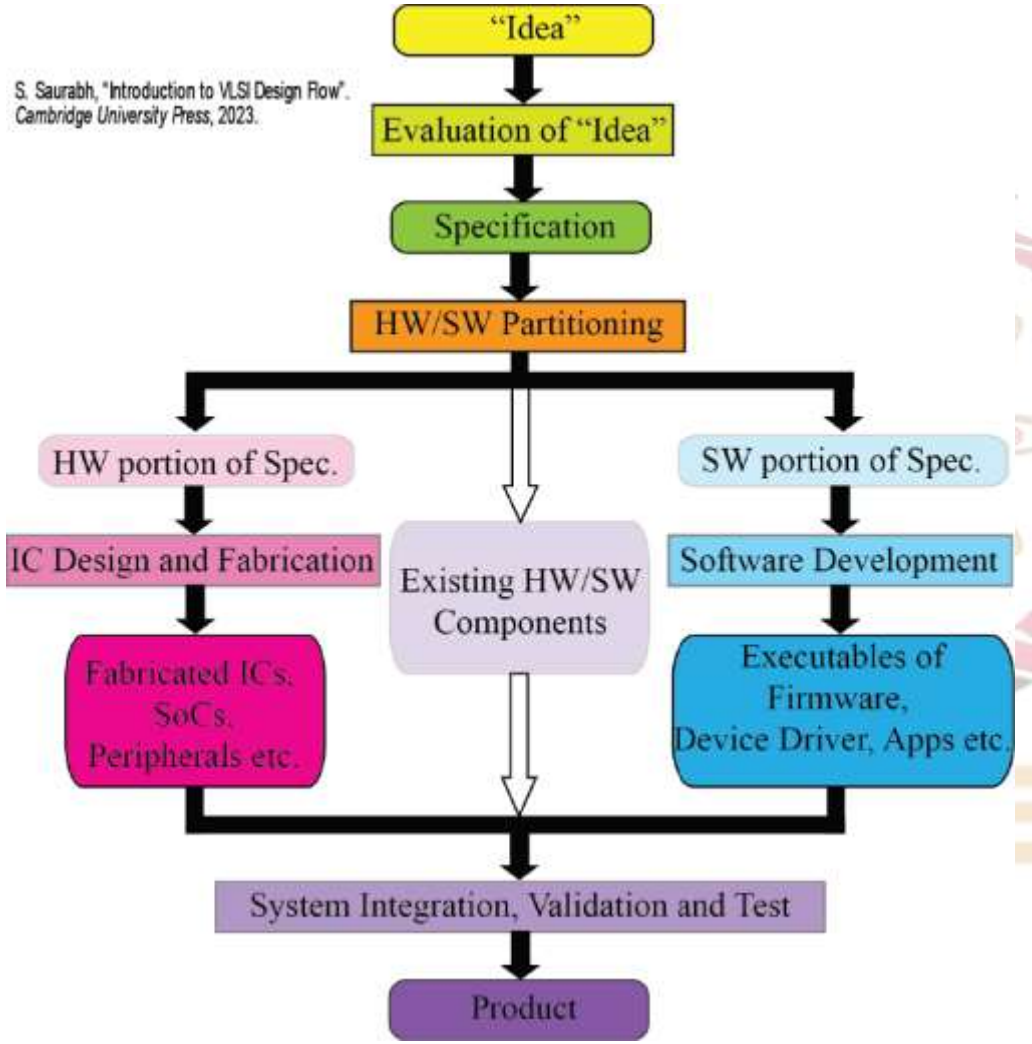
Pre-RTL Methodologies



System-level Design

S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.

System-level Design: Top View



Evaluation of "idea":

- Market requirement
- Financial viability
- Technical feasibility

Preparing specifications:

- Features (functionality)
- PPA
- Time to market (TTM)

HW—SW Partitioning:

- Identify components
- Determine which components to implement in HW/SW

- HW/SW Development (separately)
- System Integration, Validation, Test
- Final Product

Pre-RTL Methodologies

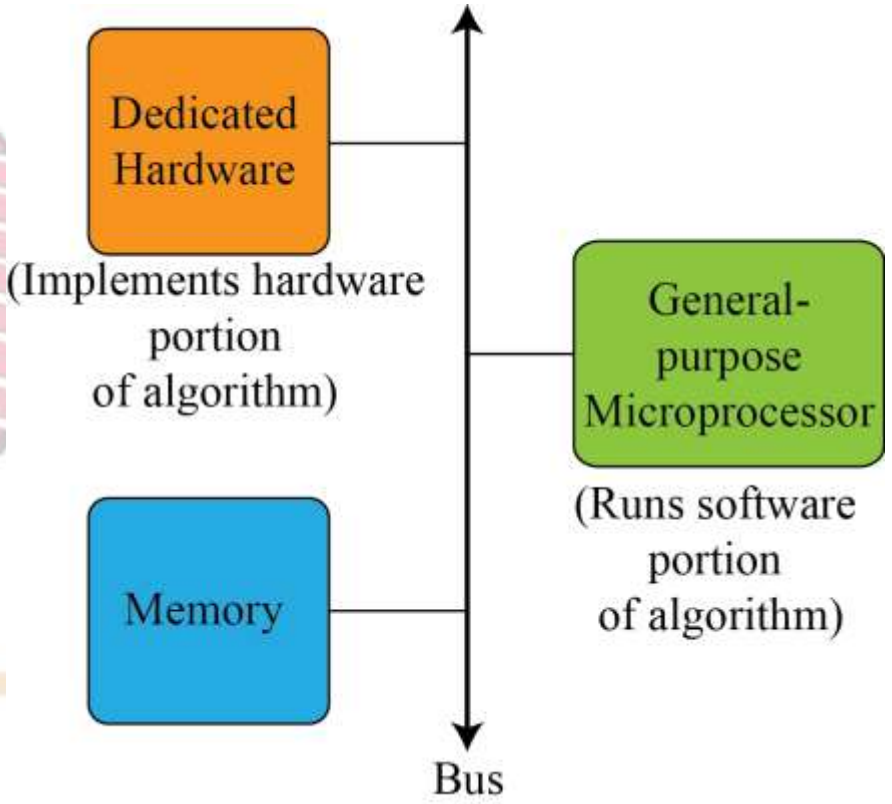
Hardware—software Partitioning

Hardware—Software Partitioning: Motivation

Motivation: Exploit the merits of both hardware and software by choosing right ***combination of hardware and software*** to implement a given function

	Hardware	Software
Performance	High	Low
Cost	High	Low
Risk due to bug	High	Low
Customization	Low	High
Development Time	High	Low

- Hardware: usually runs as parallel circuits and can have very good PPA
  - ➢ Can be implemented in full custom IC, ASIC or FPGA
- Software: usually run sequentially on a general purpose processor



Hardware/Software Partitioning: Example

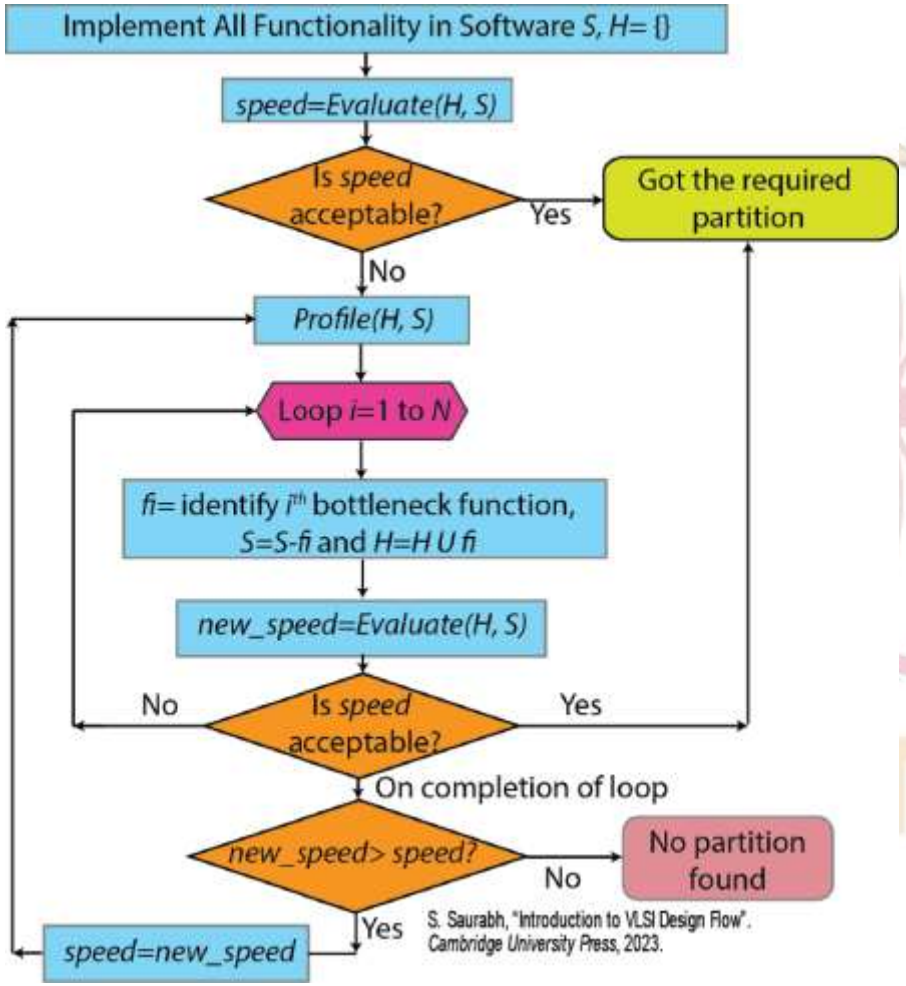
Video Compression

- Algorithm can be divided in two main parts:
  1. Computing Discrete Cosine Transform (DCT): Performed multiple times, bottleneck
  2. Frame Handling and other computation

- **DCT**
  - Hardware for computing DCT
  - Can be computed using parallel circuits
  - Several orders of magnitude faster and more energy efficient implementation in hardware

- **Frame Handling and other computation**
  - Software on a general purpose microprocessor
  - Provides Flexibility

Hardware/Software Partitioning: Methodology(1)



Objective: Finding a minimum set of functions that need to be implemented in hardware to achieve the desired performance

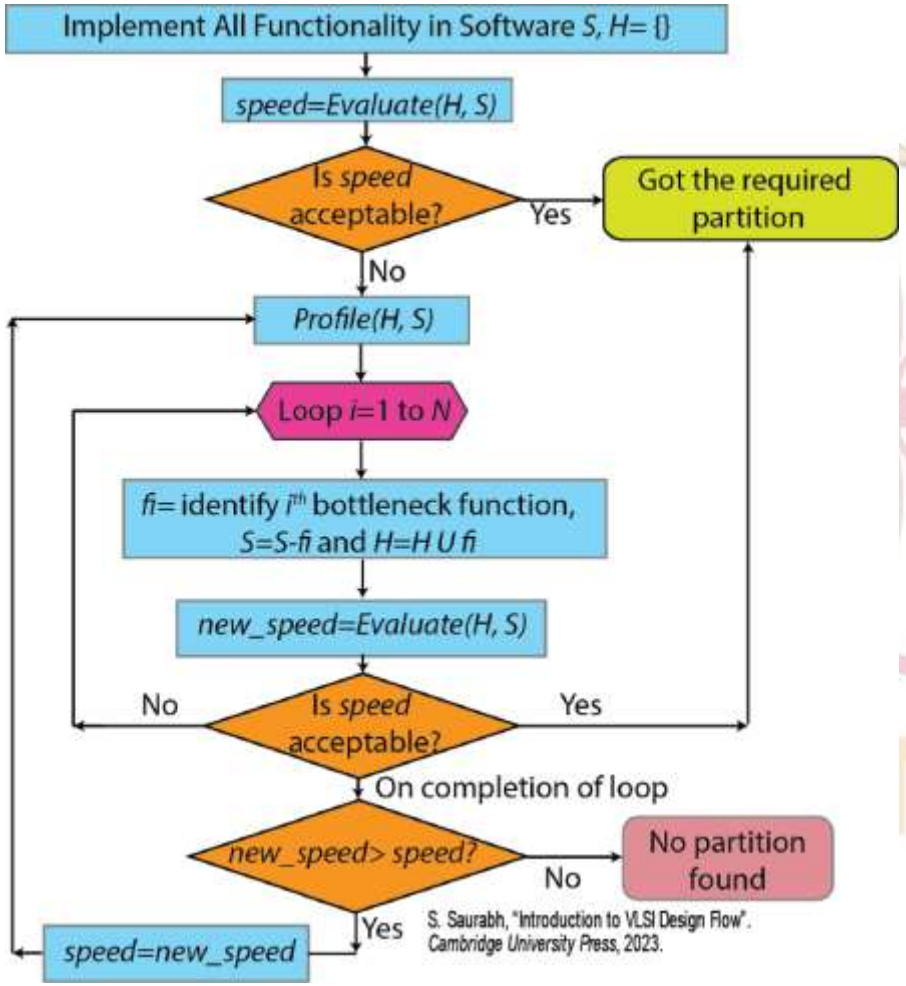
Inputs:

- Given algorithm that is implemented entirely in software
  - S contains set of functions implemented in software
- Acceptable performance P
- Parameter for the algorithm N: maximum number of functions to be move to hardware in each iteration

Output:

- Set of function H to be implemented in hardware (initially H is empty)

Hardware/Software Partitioning: Methodology (2)



Measure performance: $Evaluate(H, S)$

Profiling: Measures frequency or duration of each function calls $[Profile(H, S)]$

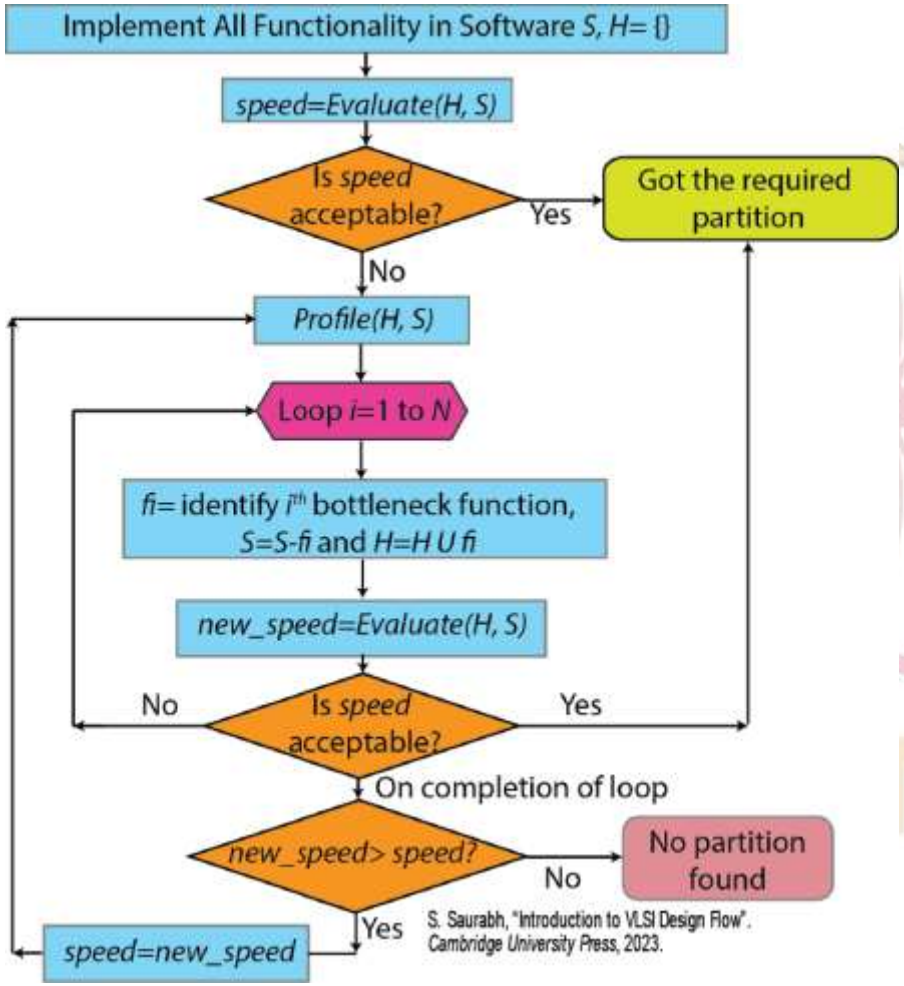
In each iteration:

- Identify $i-th$ most severe bottleneck function fi
  - Assume that fi is implemented in the hardware
  - Measure performance and check whether target performance P is met
- Moves maximum of N most critical bottleneck functions to hardware

Termination criteria:

- Success: Performance target P is met
- Failure: No improvement even after moving N functions to hardware

Hardware/Software Partitioning: Methodology (3)



- Takes a greedy approach
- Very simplistic

Challenges:

- Performance estimation
- Verification: hardware-software co-simulation

References

- S. Saurabh, “Introduction to VLSI Design Flow”. Cambridge: *Cambridge University Press*, 2023.