Five Ways to Design Future SoC

Kazuaki J. Murakami

Director

Computing & Communications Center

Kyushu University

E-mail: murakami@cc.kyushu-u.ac.jp

Messages in This Talk

- Platform-based design will dominate the future SoC designs
- Among many platform architectures, “multiple reconfigurable processor” seems promising to me
- Anyway, we need a comprehensive quantitative analysis on the cost^3/performance of these architectures
  - This talk will give just some taxonomies and a qualitative comparison among them

$\text{Cost}^3 = \text{Design Cost} \times \text{Production Cost} \times \text{Running Cost}$

SoC Design... What Are the Essentials?

- How do we design and implement custom logic?
  - Custom logic is a logic, or function (e.g., MPEG4's ME/MC), which cannot or shall not be implemented on general-purpose processor (e.g., ARM) for some performance or power reason.
- There are at least five ways to implement such custom logic...

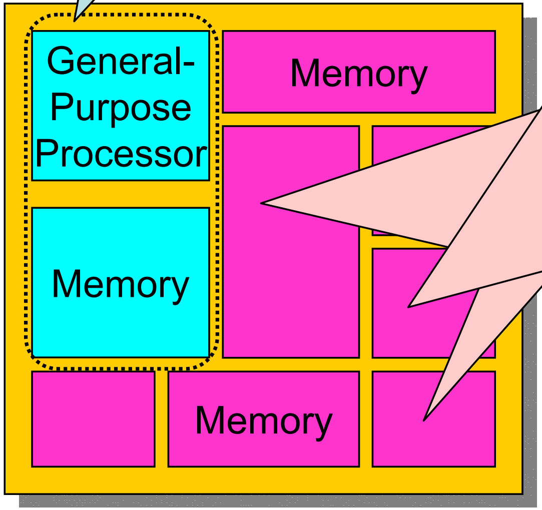
Five Ways to Design & Implement Custom Logic

How to Implement General Logic

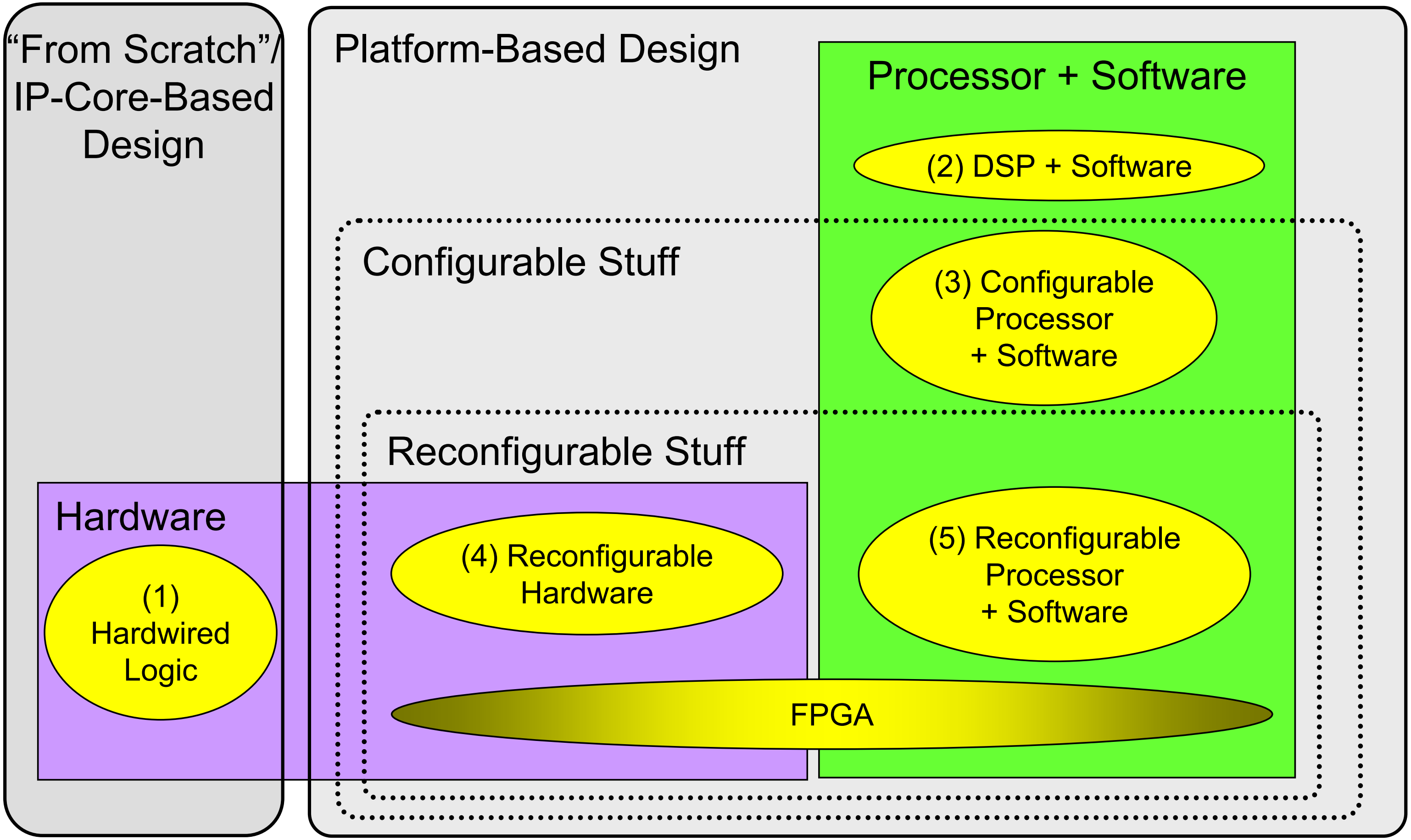
- General-Purpose Processor + Software

How to Implement Custom Logic

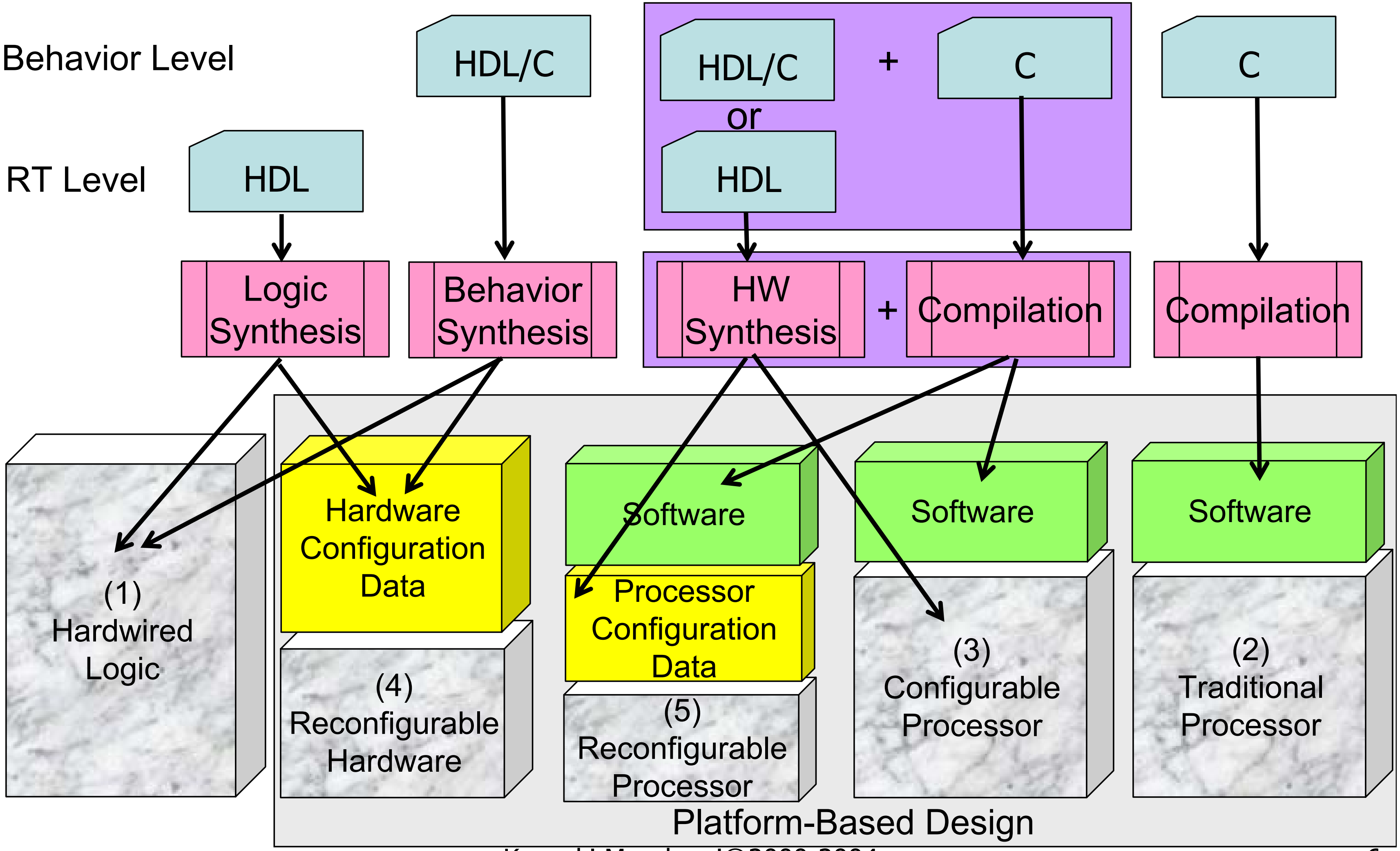
- (1) **"From Scratch" Approach**
  - Design new hardware every time
- (1) **IP-Core-Based Approach**
  - Reuse existing hardware designs, or IP cores
- Platform-Based Approaches
  - (2) **Processor + Software**
  - (3) **Configurable Processor + Software**
  - Reconfigurable Stuff
    - (4) **Reconfigurable Hardware**
    - (5) **Reconfigurable Processor + Software**



Five Ways to Design & Implement Custom Logic

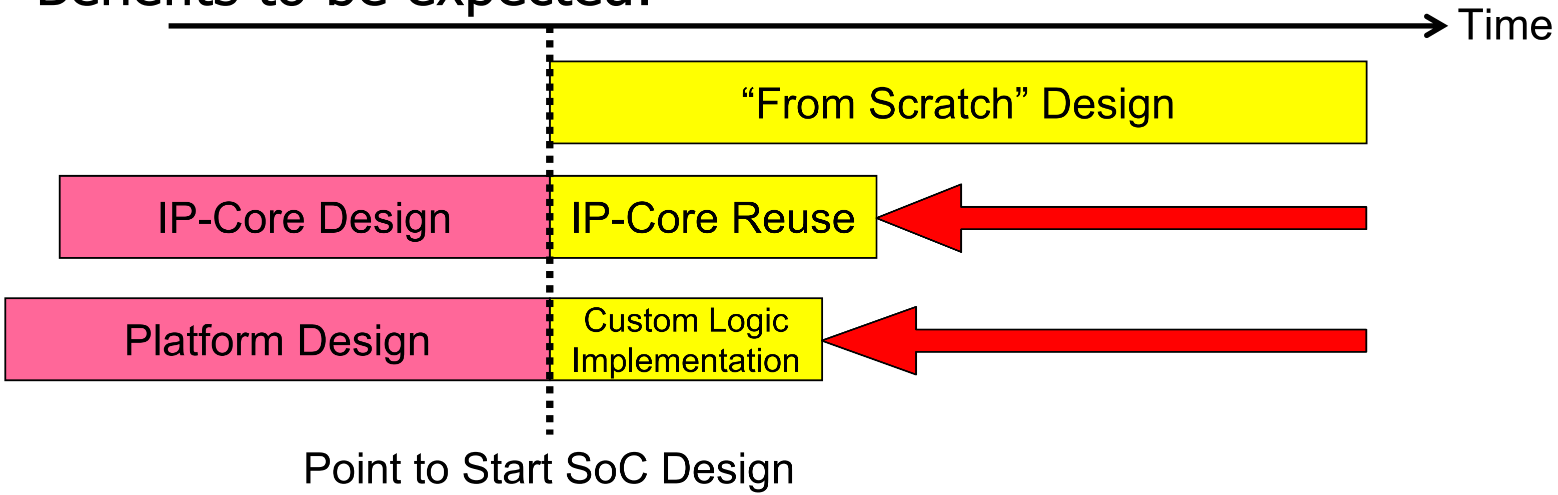

"From Scratch"/ IP-Core-Based Design
Platform-Based Design
Processor + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

Five SoC Design Flows

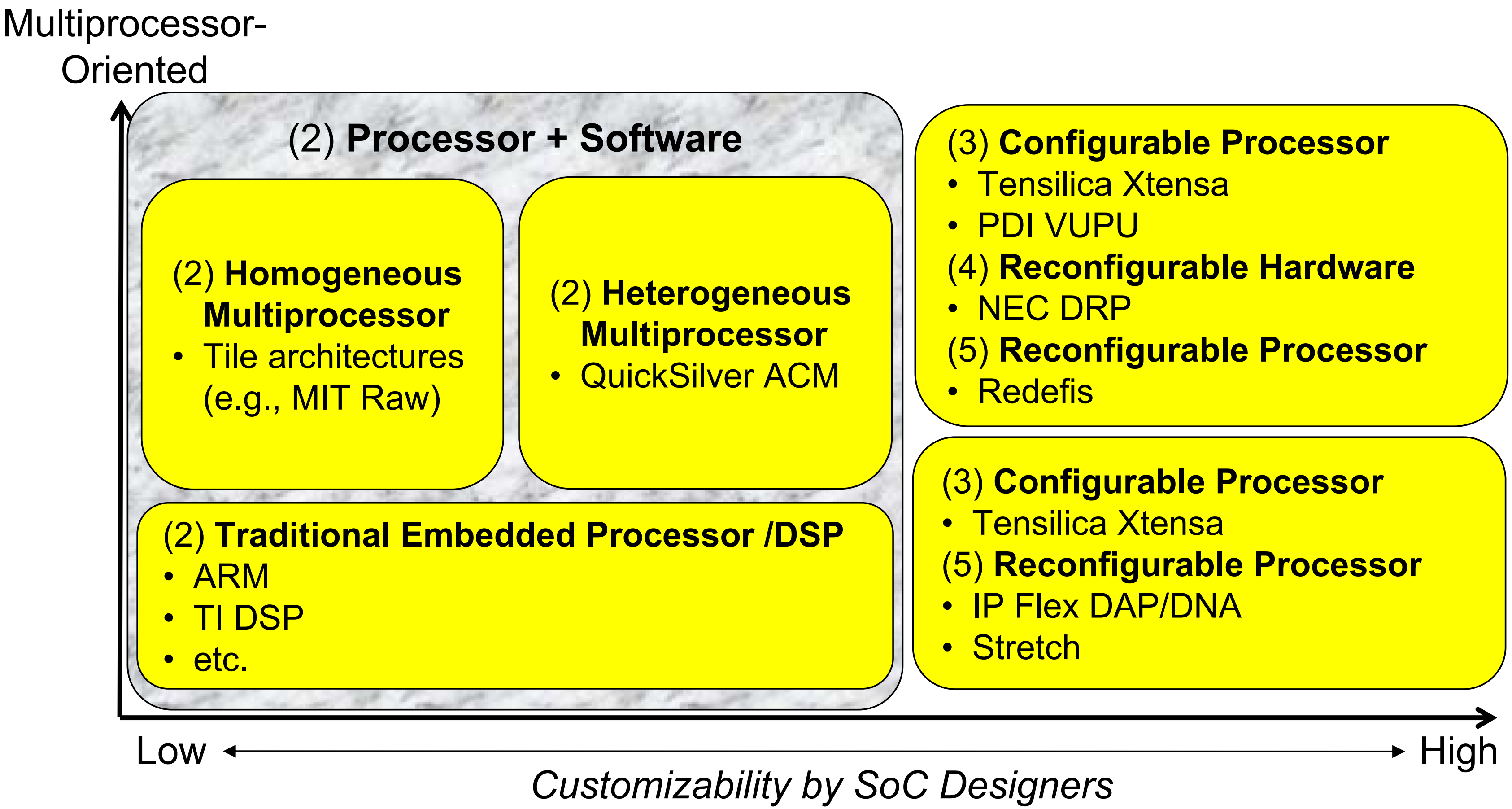


What Is Platform?

- Platform is a kind of "common" facilities used for implementing a variety of custom logic independently of the characteristics of the custom logic.
  - Analogy: Chasses for automobiles
- Platform-based SoC design: SoC design methodologies by means of the platforms
  - Counterparts: "From scratch" design, IP-core-based design
- Benefits to be expected:



Design Space of Platform Architectures



Multiprocessor-Oriented

(2) **Processor + Software**

(2) **Homogeneous Multiprocessor**
- Tile architectures (e.g., MIT Raw)

(2) **Heterogeneous Multiprocessor**
- QuickSilver ACM

(2) **Traditional Embedded Processor /DSP**
- ARM
- TI DSP
- etc.

(3) **Configurable Processor**
- Tensilica Xtensa
- PDI VUPU

(4) **Reconfigurable Hardware**
- NEC DRP

(5) **Reconfigurable Processor**
- Redefis

(3) **Configurable Processor**
- Tensilica Xtensa

(5) **Reconfigurable Processor**
- IP Flex DAP/DNA
- Stretch

Low ← → High

Customizability by SoC Designers

List of Platform Architectures

(2) Processor + Software

- Traditional Embedded Processor or DSP
- Homogeneous Multiprocessor
  - MIT Raw
- Heterogeneous Multiprocessor
  - QuickSilver ACM

(3) Configurable Processor + Software

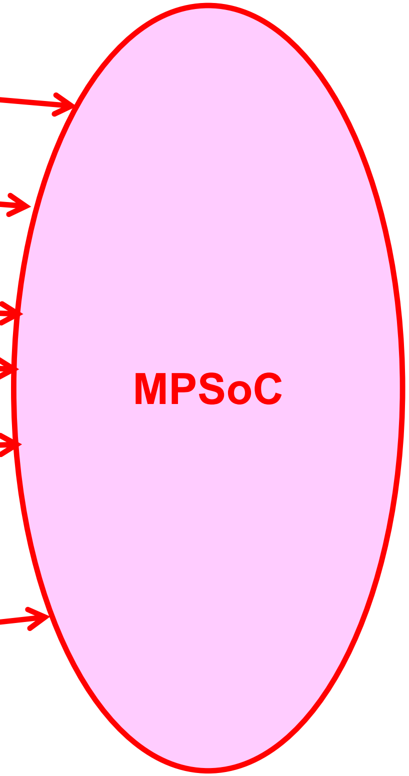
- Tensilica Xtensa
- PDI VUPU

(4) Reconfigurable Hardware

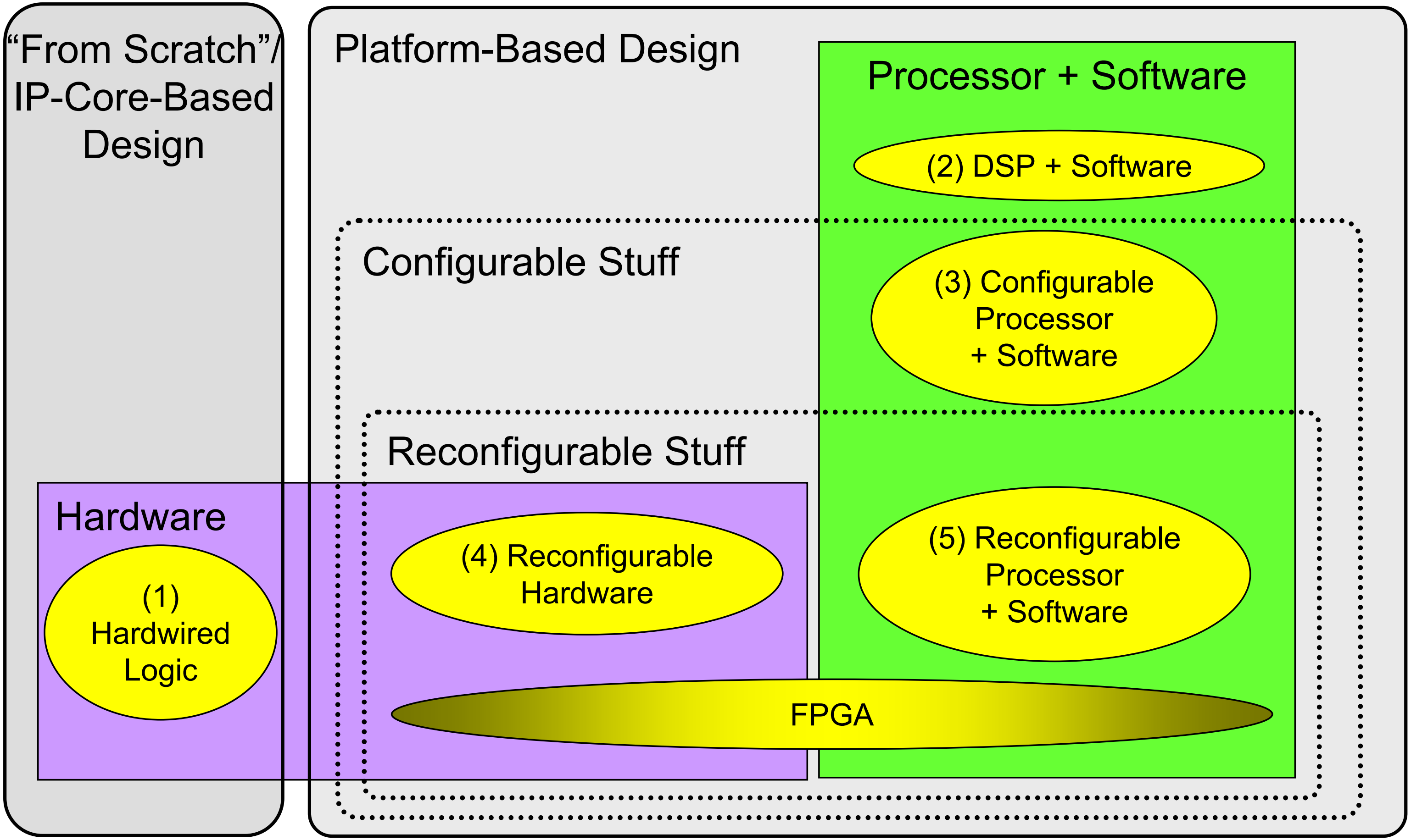
- NEC DRP

(5) Reconfigurable Processor + Software

- IP Flex DAP/DNA
- Stretch
- Redefis

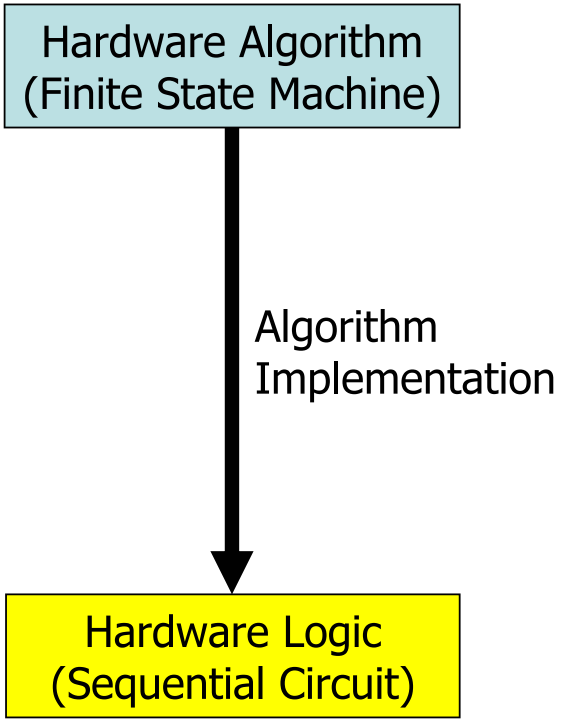


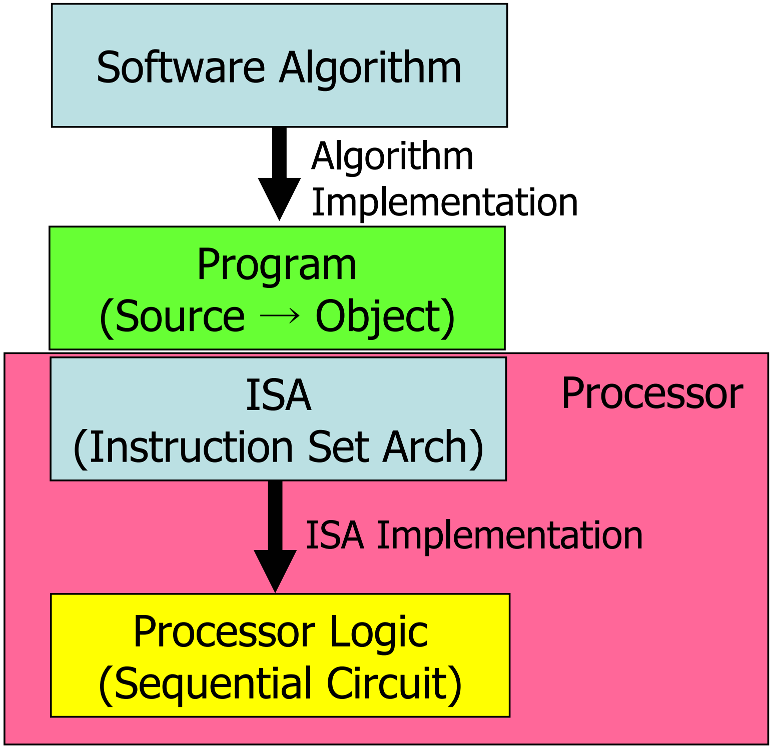
Five Ways to Design & Implement Custom Logic


"From Scratch"/ IP-Core-Based Design
Platform-Based Design
Processor + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

Hardware vs. Processor + Software

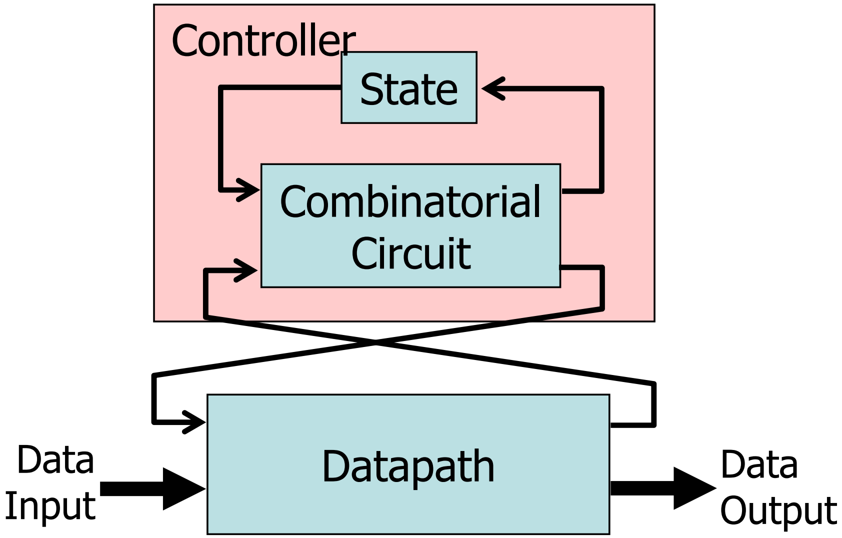
- Hardware
- Processor + Software



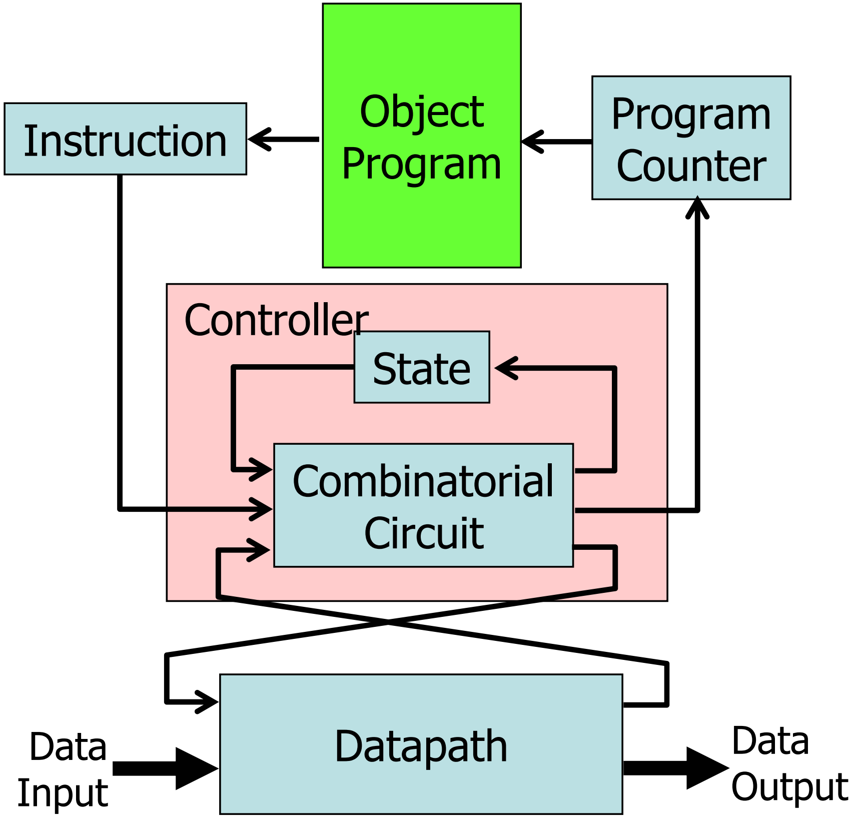


Hardware vs. Processor + Software

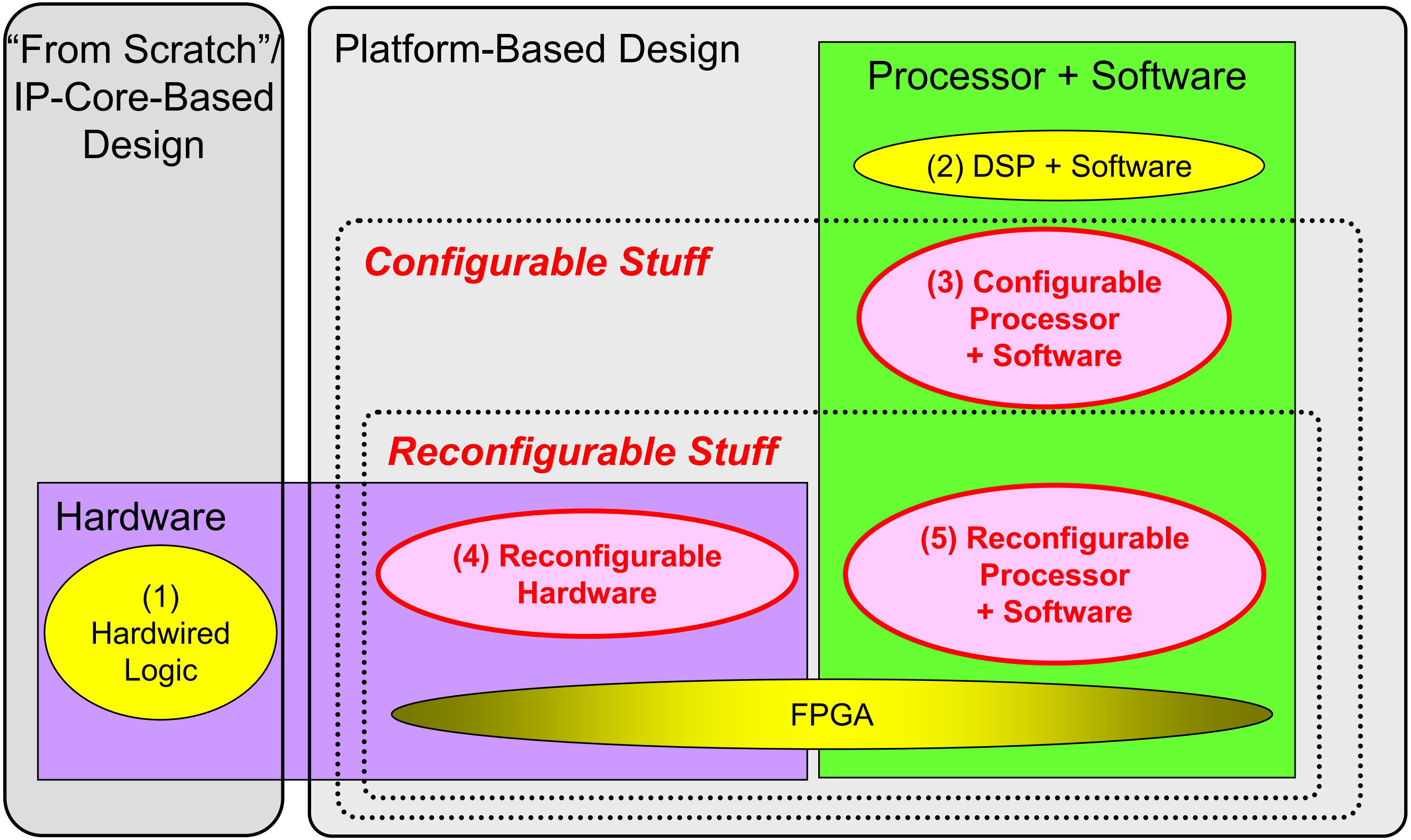
- Hardware



- Processor + Software



The Ways (3) – (5) to Design & Implement Custom Logic

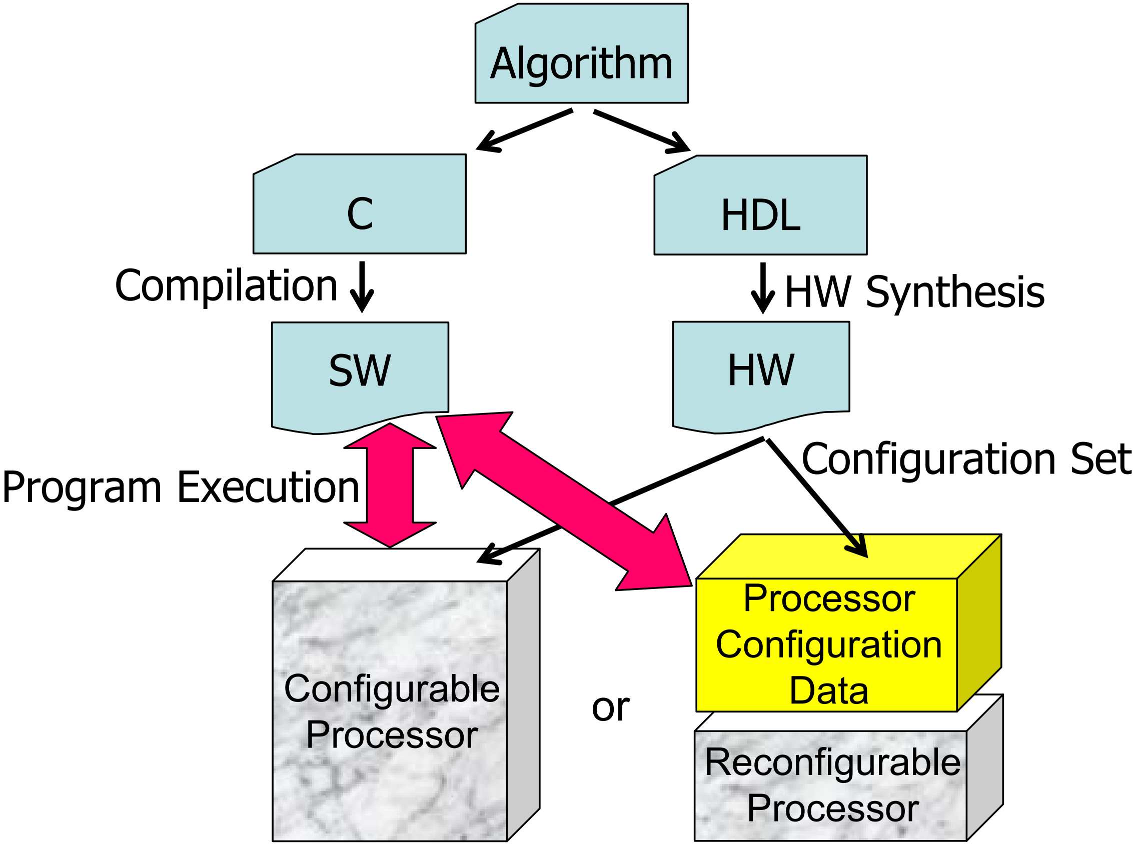

"From Scratch"/ IP-Core-Based Design
Platform-Based Design
Processor + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

What Are “Configurable” and “Reconfigurable”?

- Some functions to be implemented in hardware or processor can be designed and set by SoC designers
  - *Configurable*: Can be set just once
  - *Reconfigurable*: Can be set multiple times

	Hardware	Processor
Non-configurable	—	Traditional Processor
Configurable	Traditional Hardware Design	(3) Configurable Processor
Reconfigurable	(4) Reconfigurable Hardware	(5) Reconfigurable Processor

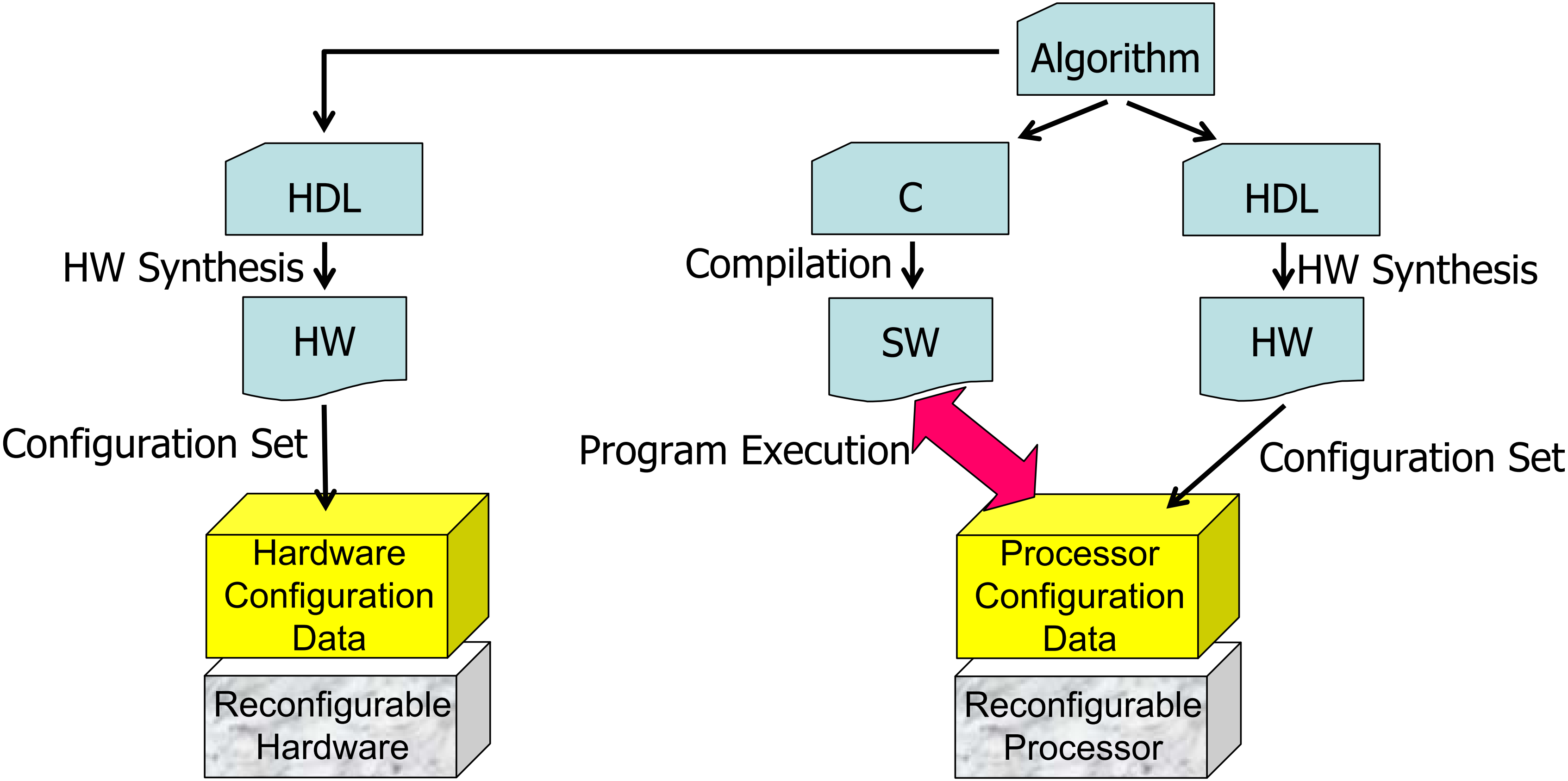
(3) "Configurable" vs. (5) "Reconfigurable" Processor



(3) Configurable Processor
- Tensilica Xtensa
- PDI VUPU

(5) Reconfigurable Processor
- IP Flex DAP/DNA
- Redefis

Reconfigurable
(4) “Hardware” vs. (5) “Processor”



- (4) Reconfigurable Hardware
  - NEC DRP

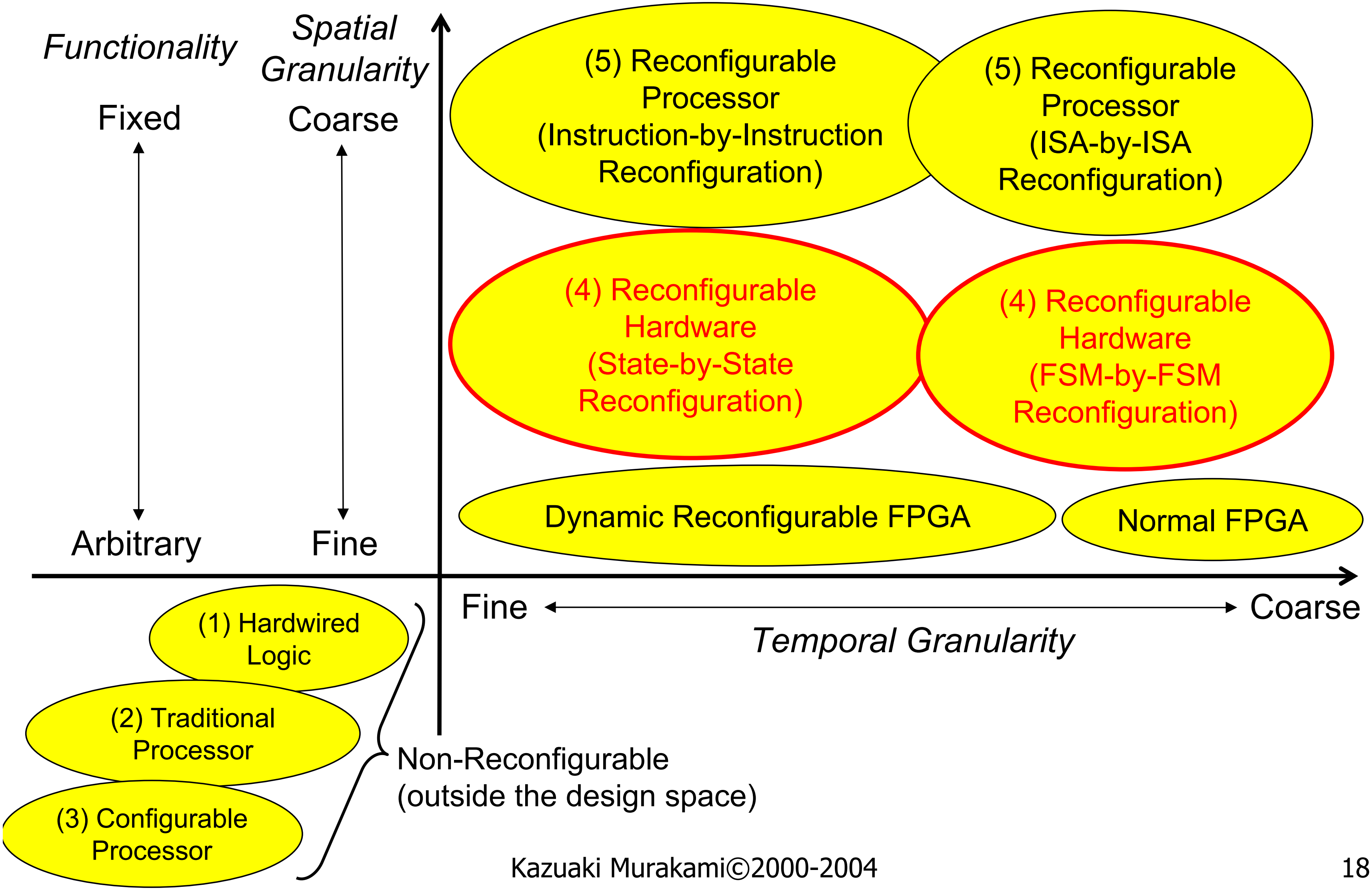
(5) Reconfigurable Processor
  - IP Flex DAP/DNA
  - Redefis

A Taxonomy

- w.r.t. HW vs. SW, Configurability, and Reconfigurability -

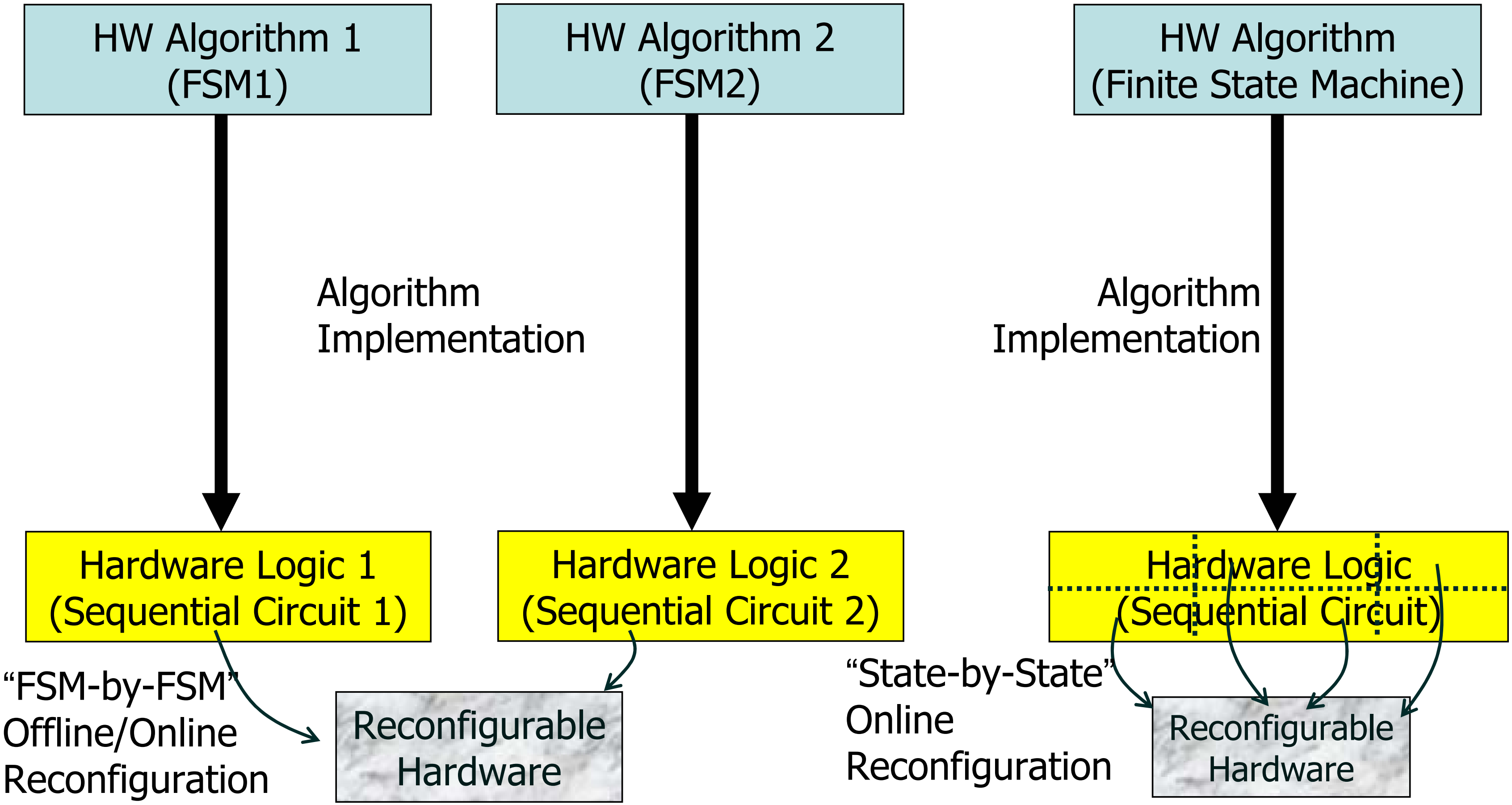
When is a HW/processor configuration generated?	How many can the configuration be set?	When is the configuration set?	Hardware	Processor + Software
- (Non-configurable)	-	-	-	•(3) Traditional Processor/DSP •(3) Multiprocessor –QuickSilver ACM
Static generation	Once	-	•(1) Hardwired Logic	•(3) Configurable Processor –Tensilica Xtensa –PDI VUPU
	Multiple times	Offline reconfiguration	•Traditional FPGA	•Traditional FPGA
		Online reconfiguration	•Dynamic Reconfigurable FPGA •(4) Reconfigurable Hardware –NEC DRP	•Dynamic Reconfigurable FPGA •(5) Reconfigurable Processor –IP Flex DAP/DNA –Stretch –Redefis
Dynamic generation	Multiple times	Online reconfiguration	•?	•?

Design Space on Reconfigurability


Functionality
Fixed
Arbitrary
Spatial Granularity
Coarse
Fine
(5) Reconfigurable Processor (Instruction-by-Instruction Reconfiguration)
(5) Reconfigurable Processor (ISA-by-ISA Reconfiguration)
(4) Reconfigurable Hardware (State-by-State Reconfiguration)
(4) Reconfigurable Hardware (FSM-by-FSM Reconfiguration)
Dynamic Reconfigurable FPGA
Normal FPGA
Fine
Coarse
Temporal Granularity
(1) Hardwired Logic
(2) Traditional Processor
(3) Configurable Processor
Non-Reconfigurable (outside the design space)

(4) Reconfigurable Hardware - Temporal Granularity -

- FSM-by-FSM
- State-by-State



HW Algorithm 1 (FSM1)

HW Algorithm 2 (FSM2)

Algorithm Implementation

Hardware Logic 1 (Sequential Circuit 1)

Hardware Logic 2 (Sequential Circuit 2)

"FSM-by-FSM" Offline/Online Reconfiguration

Reconfigurable Hardware

HW Algorithm (Finite State Machine)

Algorithm Implementation

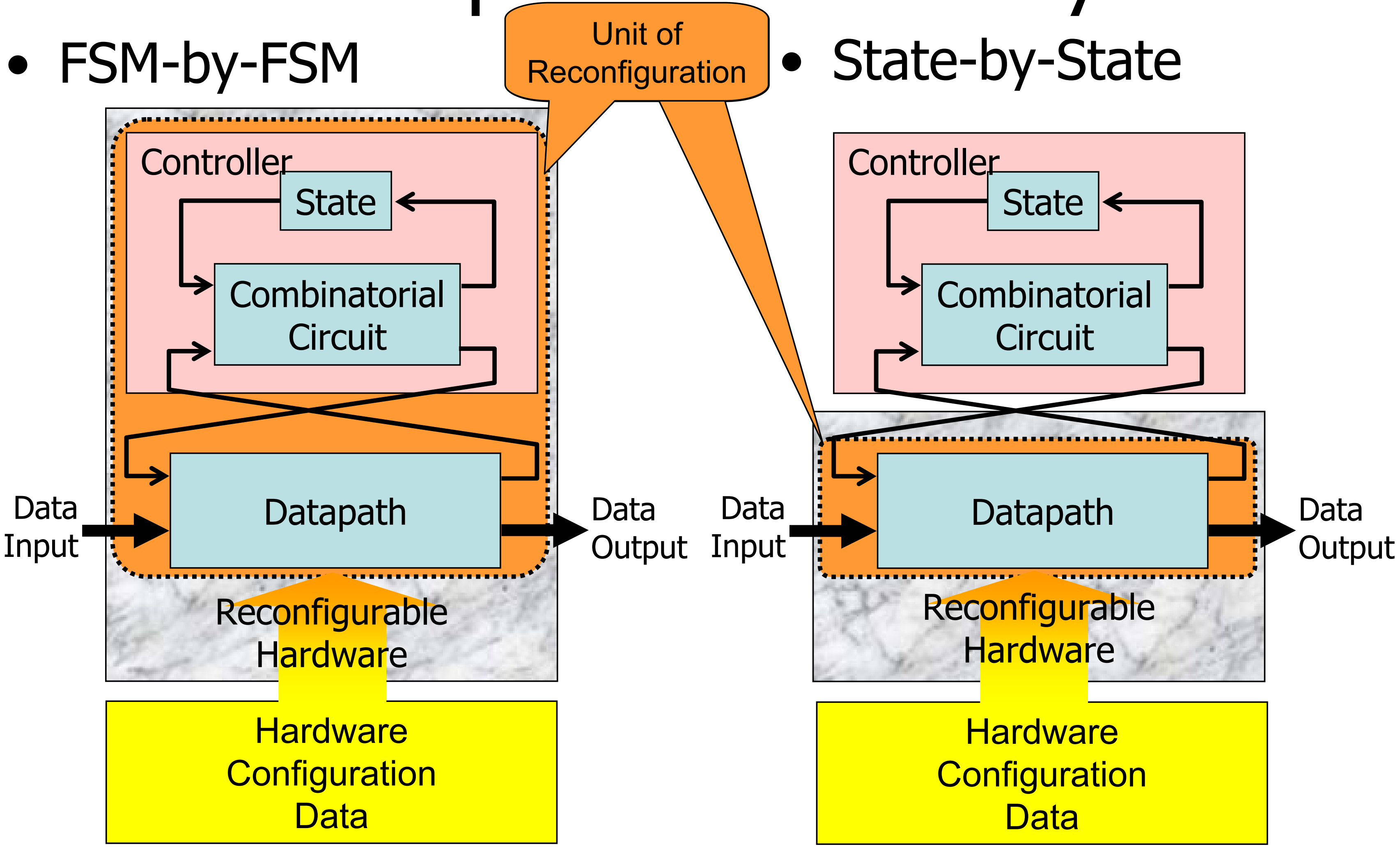
Hardware Logic (Sequential Circuit)

"State-by-State" Online Reconfiguration

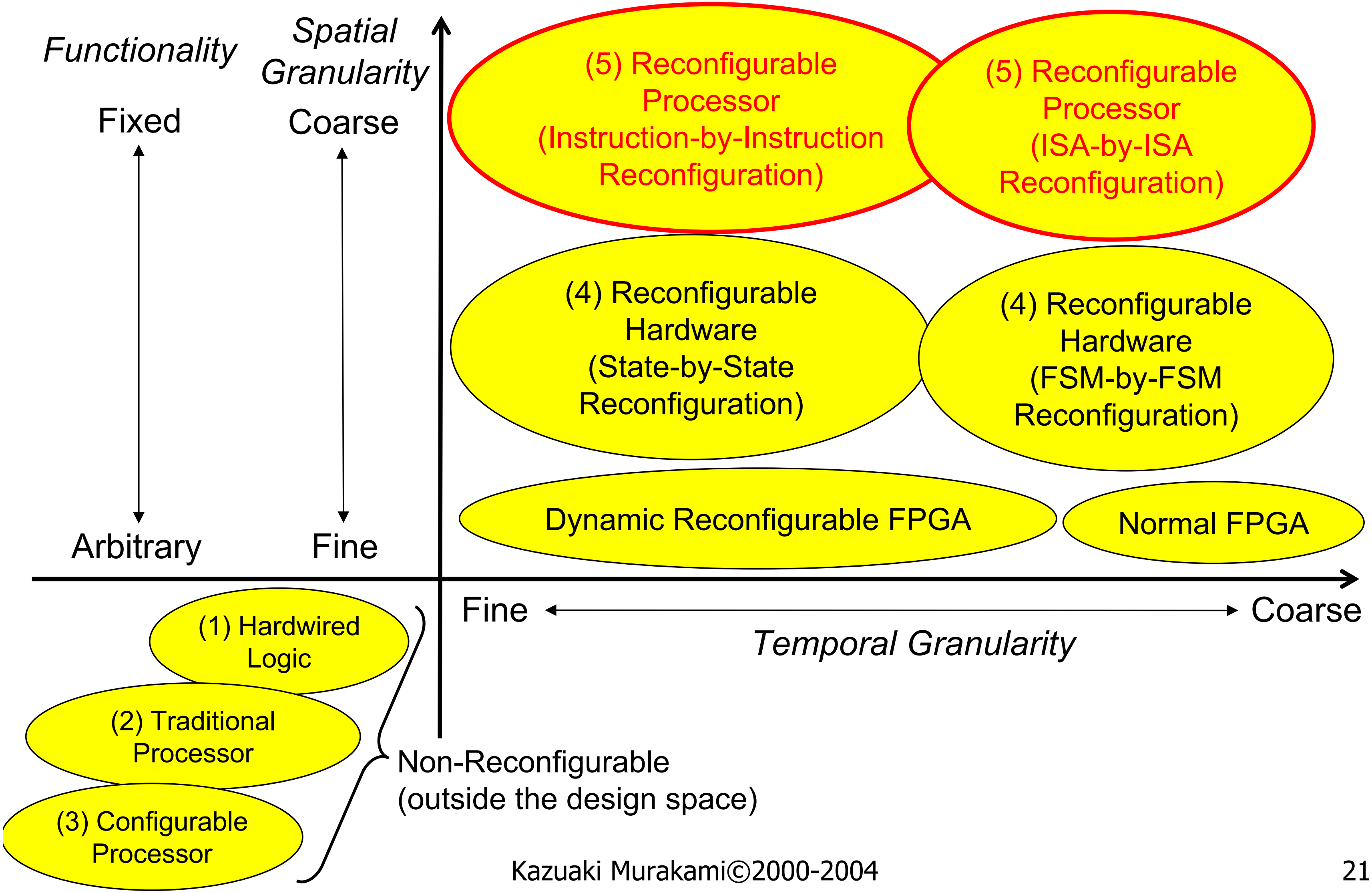
Reconfigurable Hardware

(4) Reconfigurable Hardware
- Temporal Granularity -

- FSM-by-FSM
- State-by-State



Design Space on Reconfigurability



Functionality

Fixed

Arbitrary

Spatial Granularity

Coarse

Fine

(5) Reconfigurable Processor (Instruction-by-Instruction Reconfiguration)

(5) Reconfigurable Processor (ISA-by-ISA Reconfiguration)

(4) Reconfigurable Hardware (State-by-State Reconfiguration)

(4) Reconfigurable Hardware (FSM-by-FSM Reconfiguration)

Dynamic Reconfigurable FPGA

Normal FPGA

Fine

Coarse

Temporal Granularity

(1) Hardwired Logic

(2) Traditional Processor

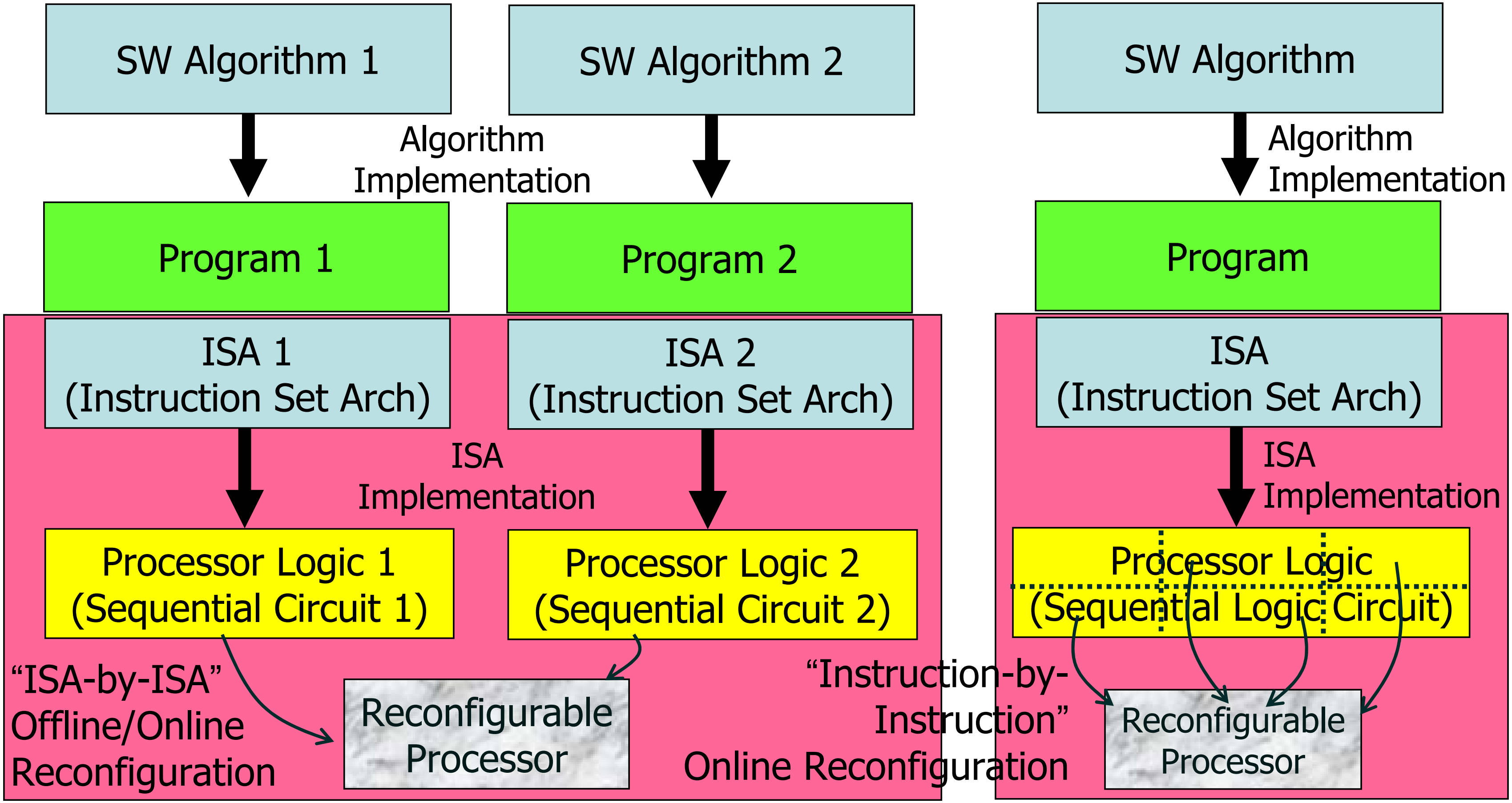
(3) Configurable Processor

Non-Reconfigurable (outside the design space)

Kazuaki Murakami©2000-2004

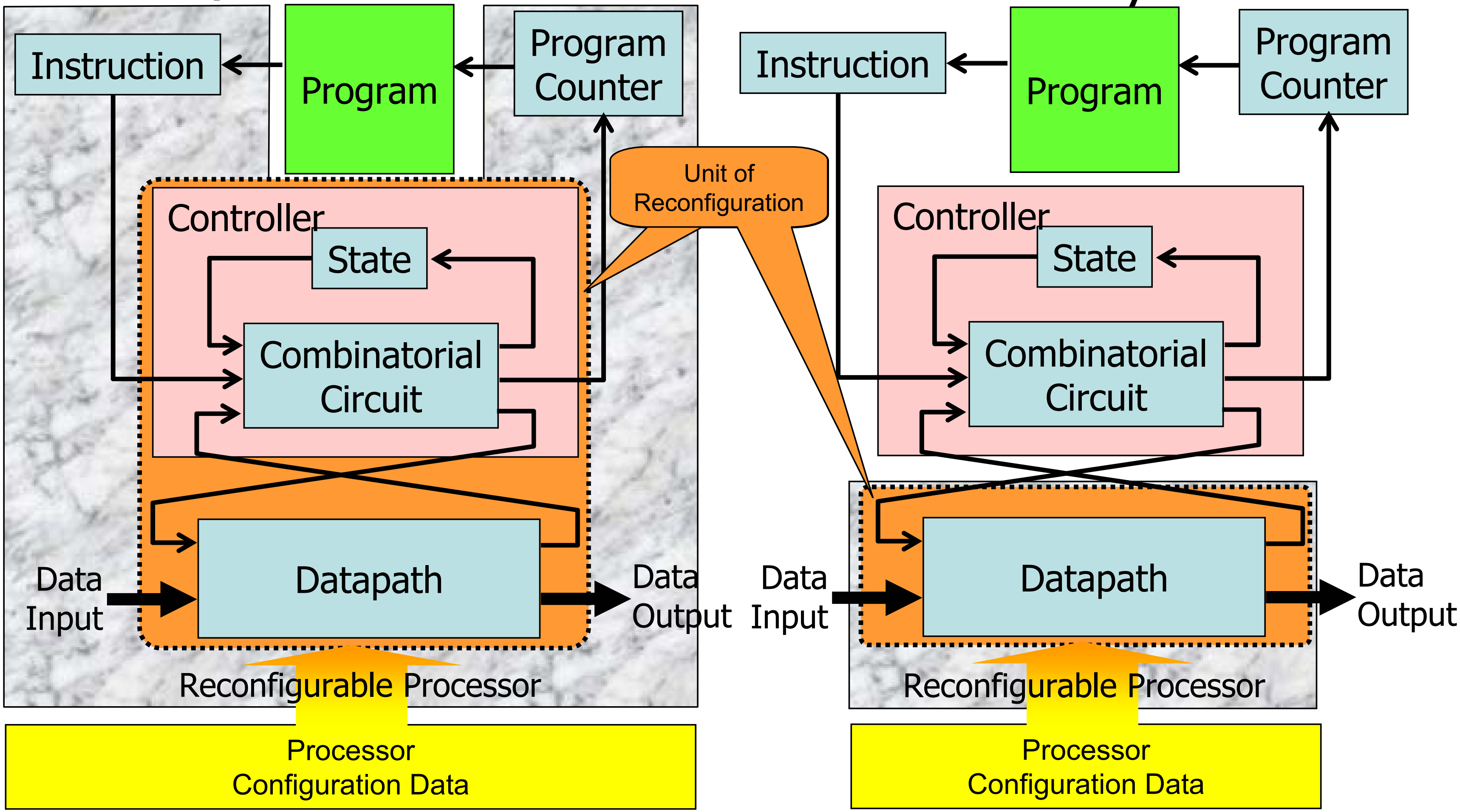
(5) Reconfigurable Processor - Temporal Granularity -

- ISA-by-ISA
- Instruction-by-Instruction

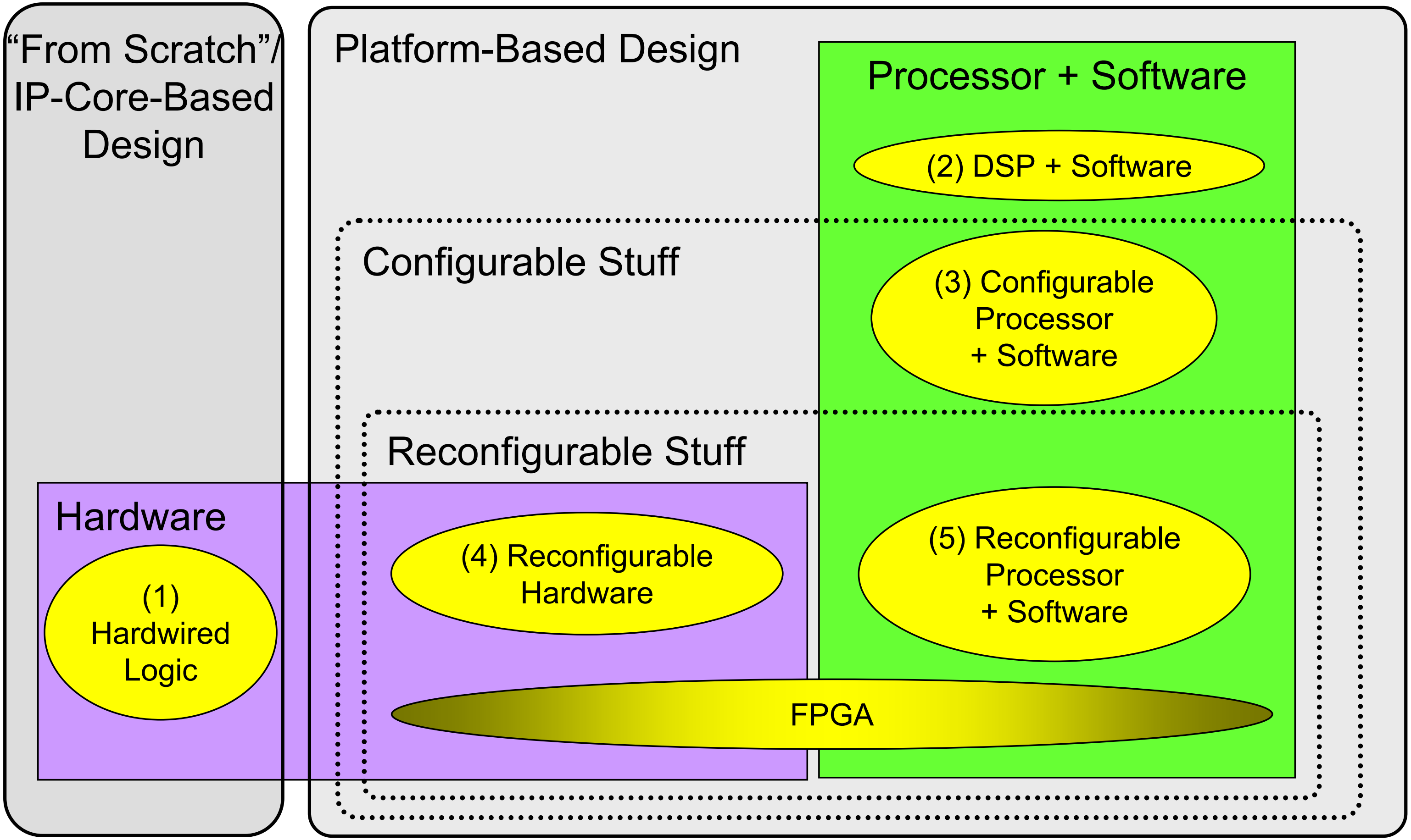

SW Algorithm 1
SW Algorithm 2
Algorithm Implementation
Program 1
Program 2
ISA 1 (Instruction Set Arch)
ISA 2 (Instruction Set Arch)
ISA Implementation
Processor Logic 1 (Sequential Circuit 1)
Processor Logic 2 (Sequential Circuit 2)
"ISA-by-ISA" Offline/Online Reconfiguration
Reconfigurable Processor
SW Algorithm
Algorithm Implementation
Program
ISA (Instruction Set Arch)
ISA Implementation
Processor Logic (Sequential Logic Circuit)
"Instruction-by-Instruction" Online Reconfiguration
Reconfigurable Processor

(5) Reconfigurable Processor
- Temporal Granularity -

- ISA-by-ISA
- Instruction-by-Instruction

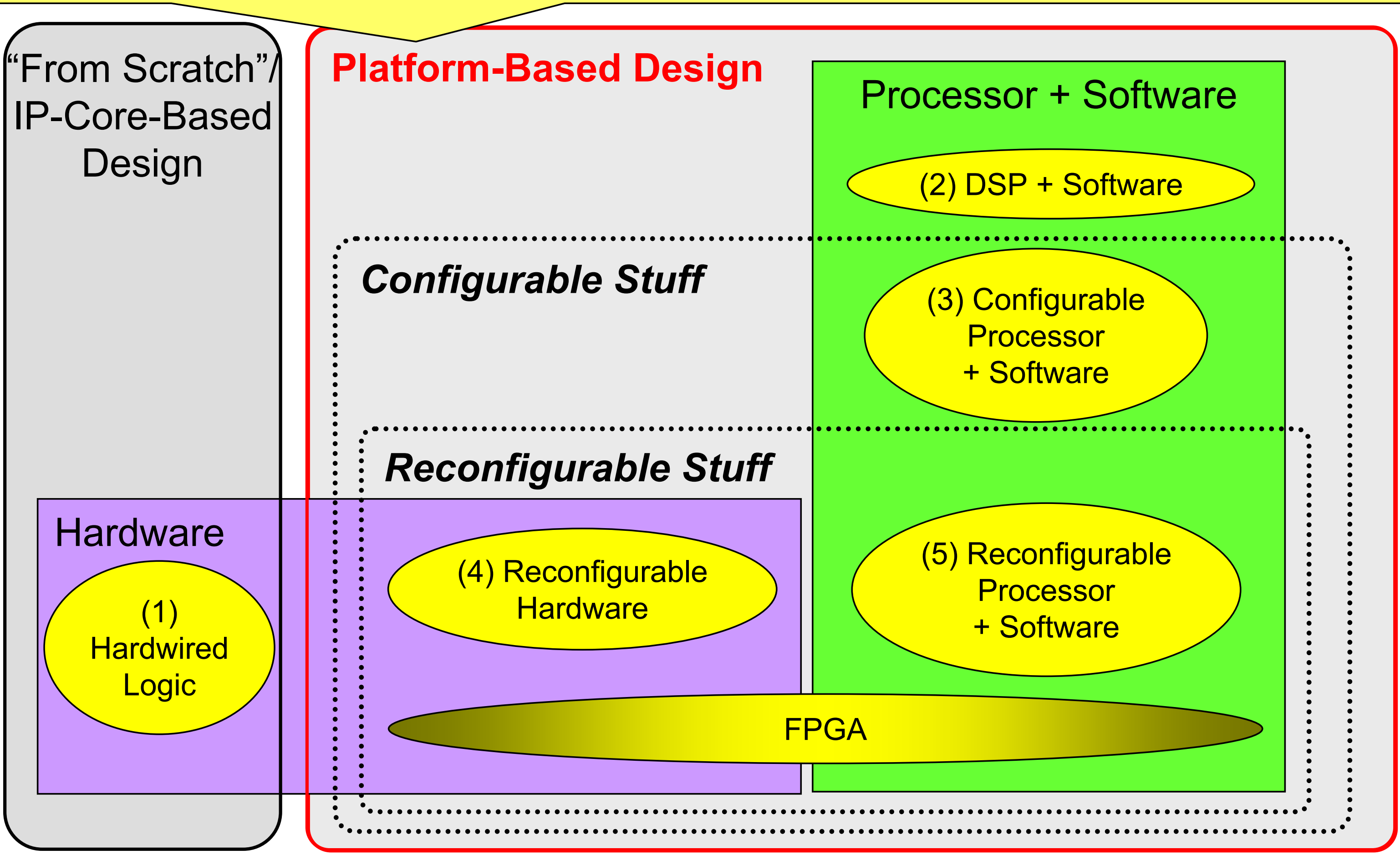


Five Ways to Design & Implement Custom Logic


"From Scratch"/ IP-Core-Based Design
Platform-Based Design
Processor + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

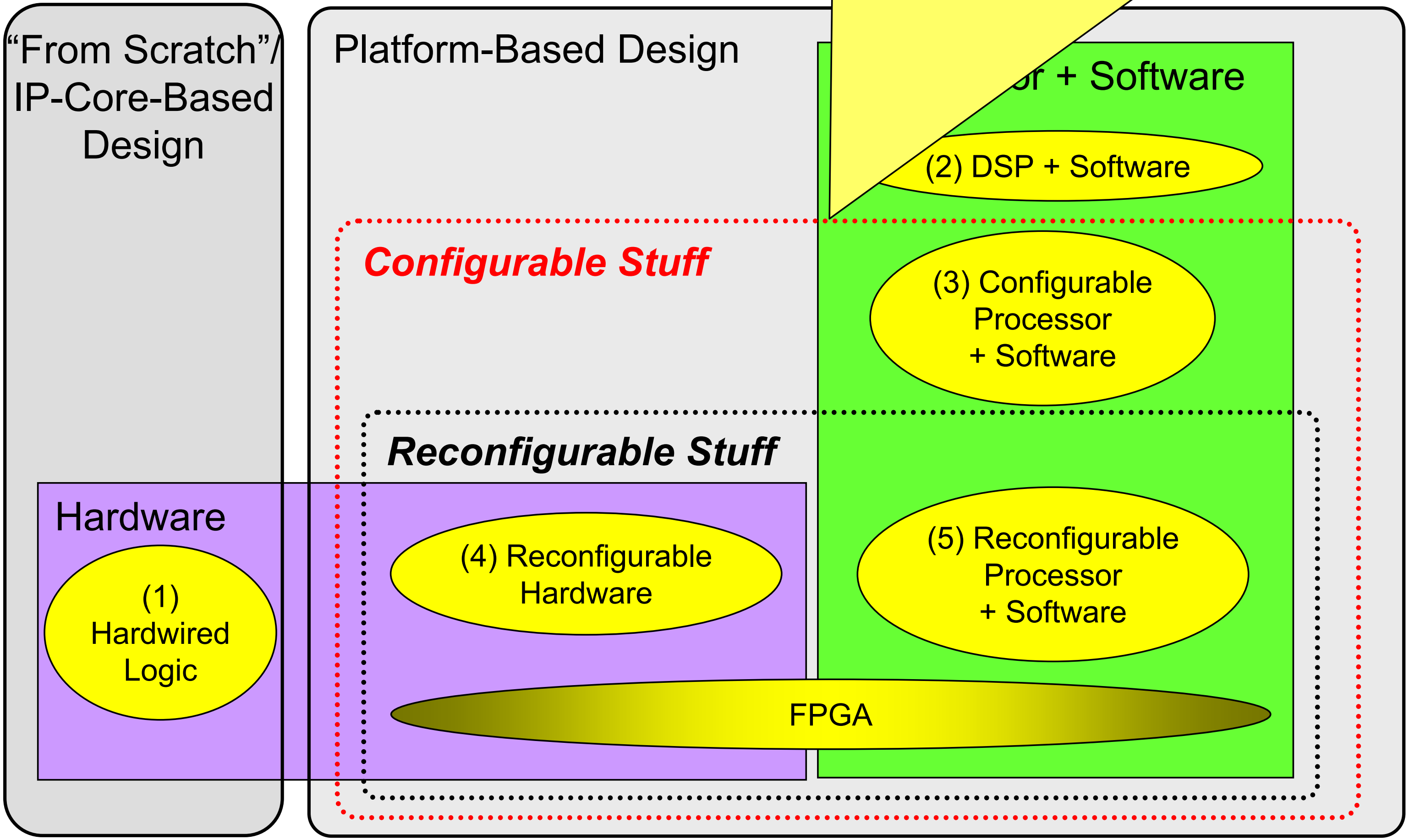
(2), (3), (4), and (5)

- Divide SoC design into two tasks: (i) platform design, and (ii) custom logic implementation on the platform
- Spend most of time on the custom logic implementation on the platform, and then reduce the TAT (turn-around time) of SoC
- Reuse a single platform-based SoC design for multiple applications like a "general-purpose" SoC


"From Scratch"/ IP-Core-Based Design
Platform-Based Design
Processor + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

<u>(3), (4), and (5)</u>

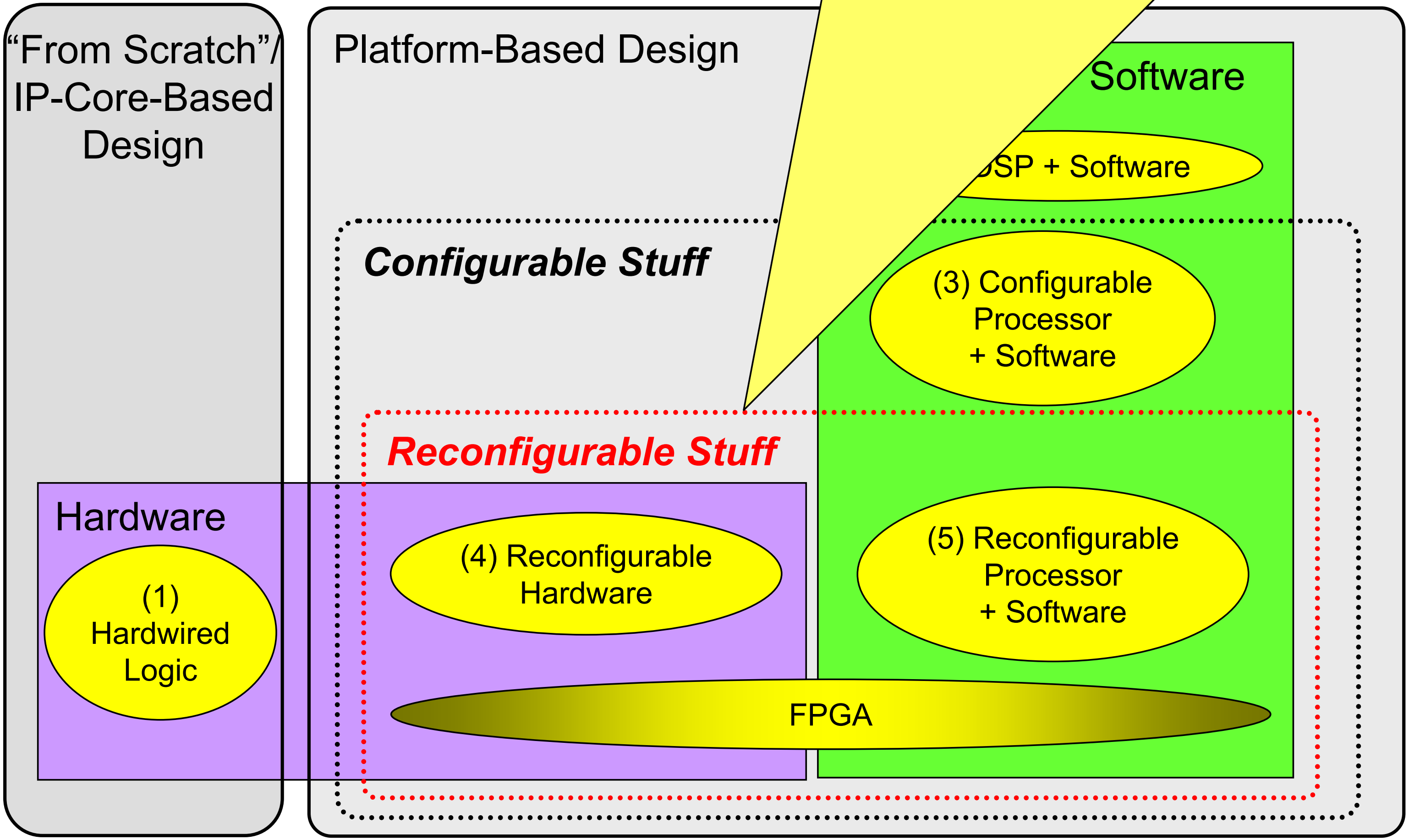
- Accommodate the HW/processor configuration to the characteristics of the custom logic to be implemented
  + Improve the cost/performance against (2) "DSP + software" approaches
  - May suffer the increase of the TAT of SoC for the accommodation task



"From Scratch"/ IP-Core-Based Design
Platform-Based Design
or + Software
(2) DSP + Software
Configurable Stuff
(3) Configurable Processor + Software
Reconfigurable Stuff
Hardware
(1) Hardwired Logic
(4) Reconfigurable Hardware
(5) Reconfigurable Processor + Software
FPGA

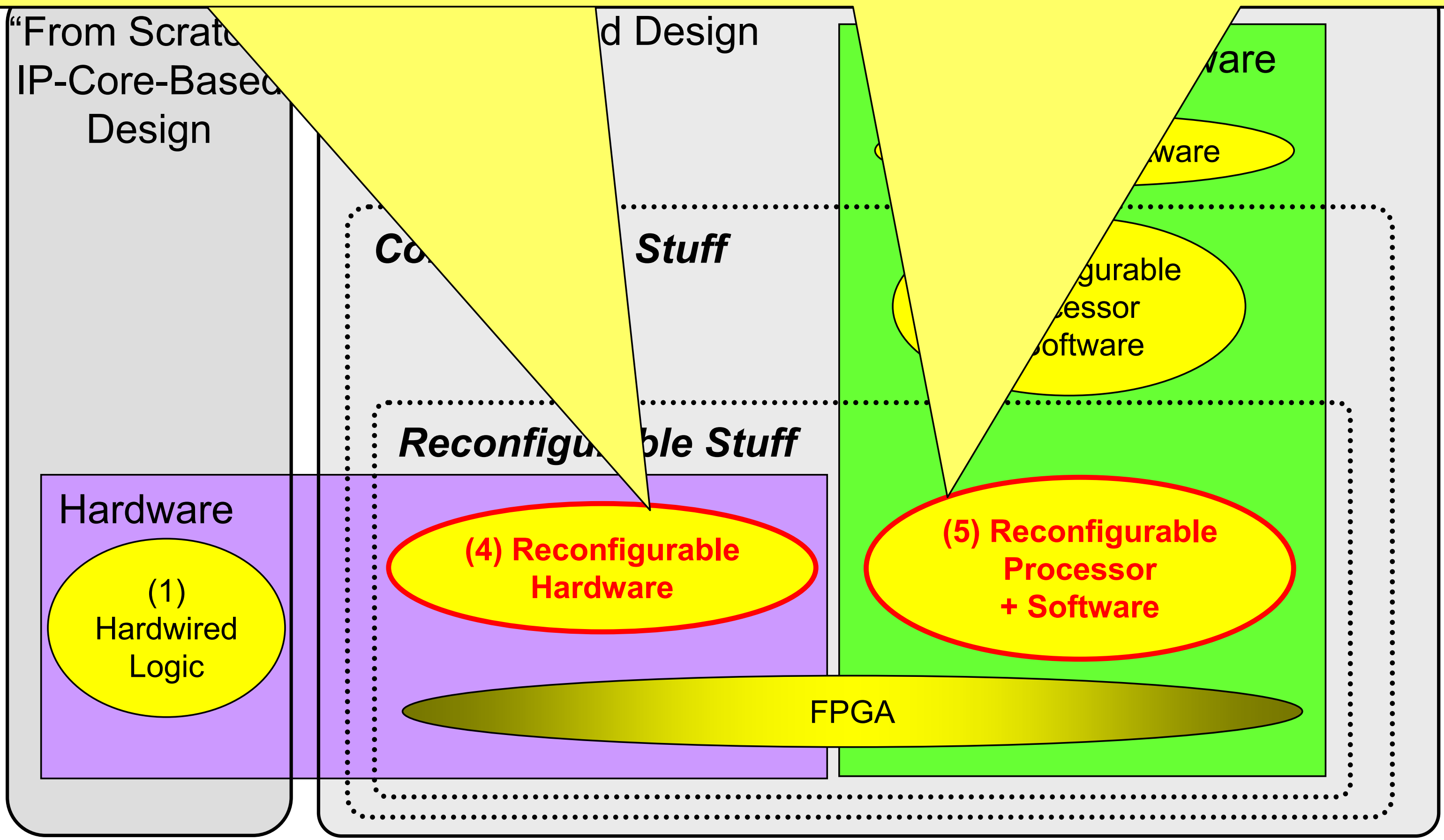
(4) and (5)

- Still accommodate the HW/processor configuration to the characteristics of the custom logic to be implemented even after the SoC is fabricated
  + Reduce the TAT of SoC more than (3) “configurable processor” approaches
  - May suffer the increase of area and performance overhead



(4) vs. (5)

- Cost/performance
  - + (4) Reconfigurable hardware
  - - (5) Reconfigurable processor
- Design productivity (affinity for C-level design)
  - - (4) Reconfigurable hardware
  - + (5) Reconfigurable processor



Summary

- Platform-based design will dominate the future SoC designs
- Among many platform architectures, “multiple reconfigurable processor” seems promising to me
- Anyway, we need a comprehensive quantitative analysis on the cost^3/performance of these architectures
  - This talk gave just some taxonomies and a qualitative comparison among them

Cost^3=Design Cost × Production Cost × Running Cost

Some References

(2) Processor + Software

- Traditional Embedded Processor or DSP
- Homogeneous Multiprocessor
  - MIT Raw ➔ http://catfish.csail.mit.edu/raw/
- Heterogeneous Multiprocessor
  - QuickSilver ACM ➔ http://www.quicksilvertech.com/

(3) Configurable Processor + Software

- Tensilica Xtensa ➔ http://www.tensilica.com/
- PDI VUPU ➔ http://www.pdi.co.jp/

(4) Reconfigurable Hardware

- NEC DRP

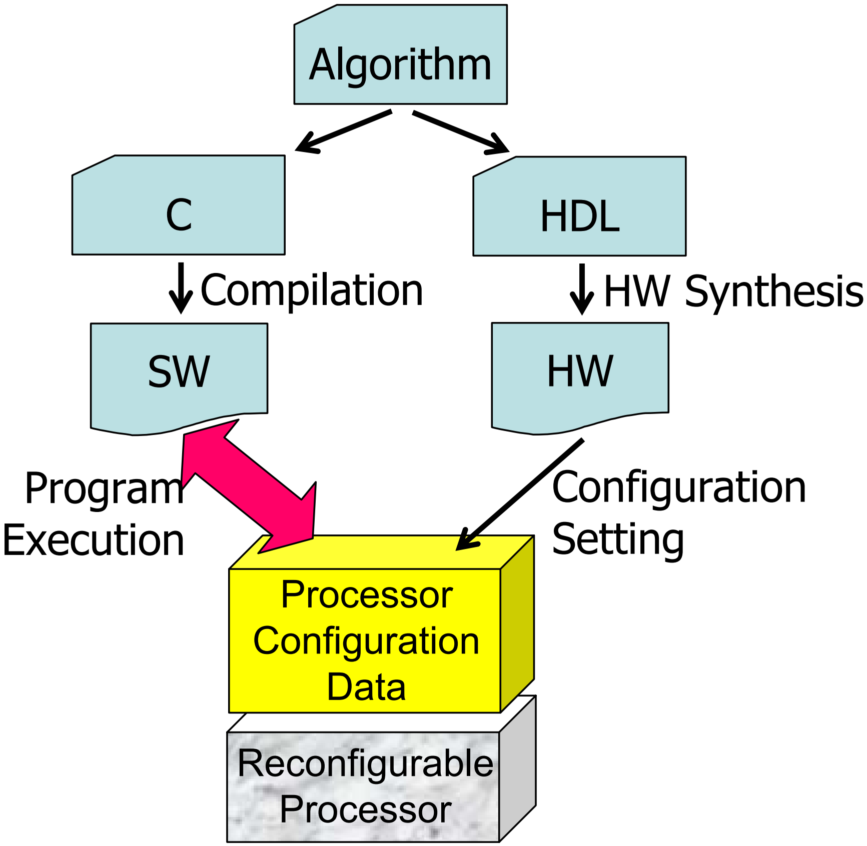
(5) Reconfigurable Processor + Software

- IP Flex DAP/DNA ➔ http://www.ipflex.com/
- Stretch ➔ http://www.stretchinc.com/
- Redefis

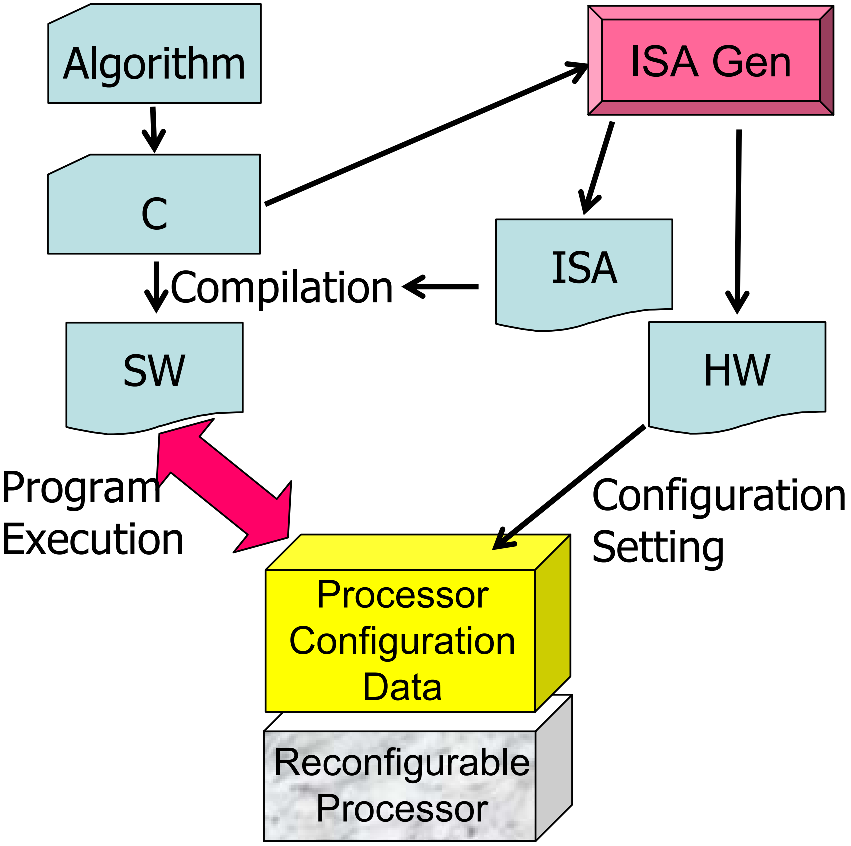
Backup Slides

Redefis (Redefinable ISA Processor): A Reconfigurable Processor

- Normal Reconfigurable Processor



- Redefis (Redefinable ISA Processor)



Algorithm
ISA Gen
C
ISA
Compilation
SW
HW
Program Execution
Configuration Setting
Processor Configuration Data
Reconfigurable Processor