FPGA Basis

Hiroki Nakahara
Tokyo Institute of Technology

Outline

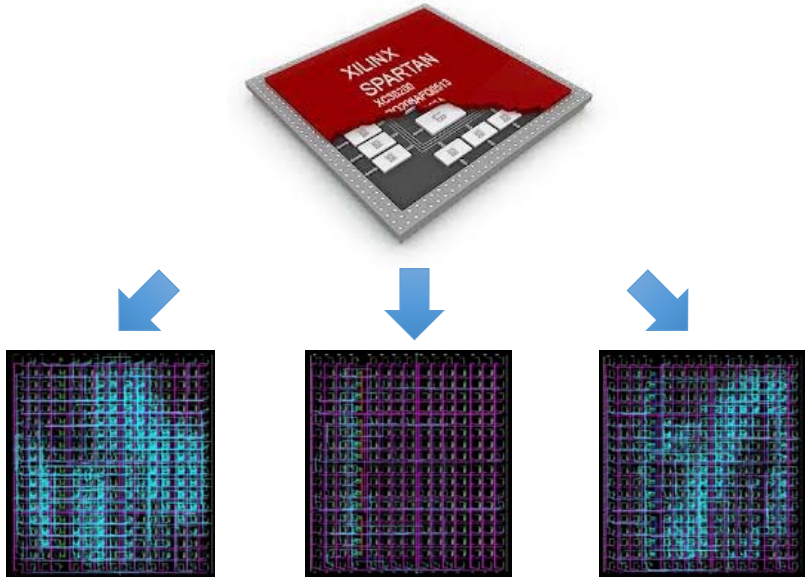
- FPGA Basis
  - FPGA Architecture
- Standard FPGA Design
- High Level Synthesis (HLS) for the FPGA
- Summary

FPGA Basis

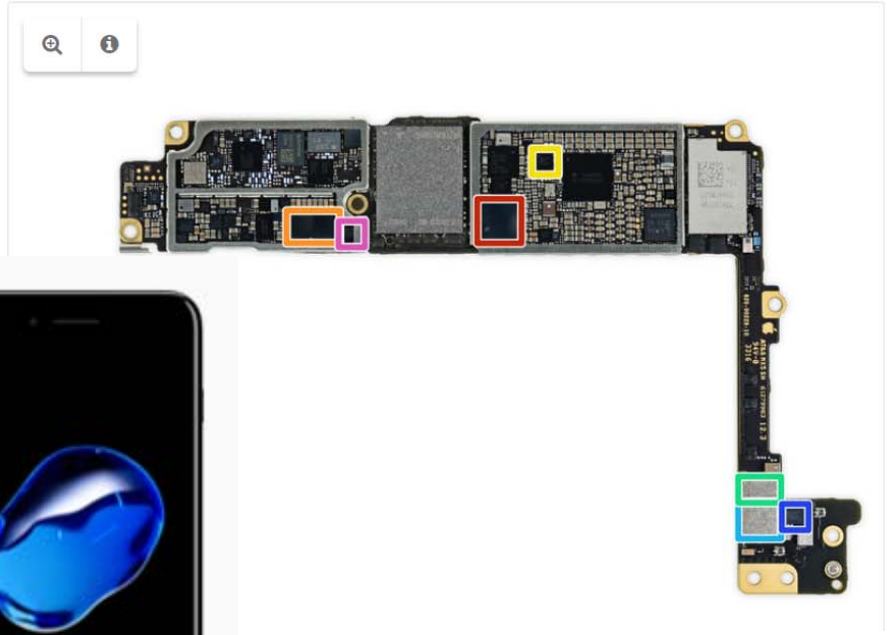
1.1 Programmable Device

FPGA

- Reconfigurable LSI or Programmable Hardware
- Programmable Logic Array and Programmable Interconnection
- Programmed by Reconfigurable Data



iPhone7 Plus



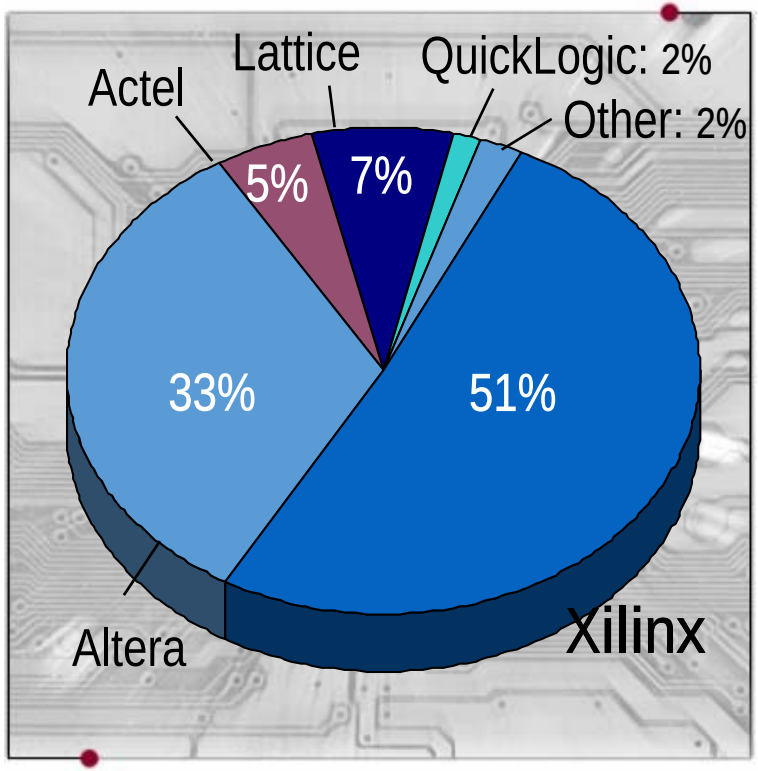
- But wait, there are even more ICs on the back!
- Apple/Cirrus Logic 338S00105 Audio Codec
- Cirrus Logic 338S00220 Audio Amplifier(x2)
- Lattice Semiconductor ICE5LP4K
- Skyworks 13702-20 Diversity Receive Module
- Skyworks 13703-21 Diversity Receive Module
- Avago LFI630 183439
- NXP 610A38

Source: https://www.ifixit.com/Teardown/iPhone+7+Plus+Teardown/67384

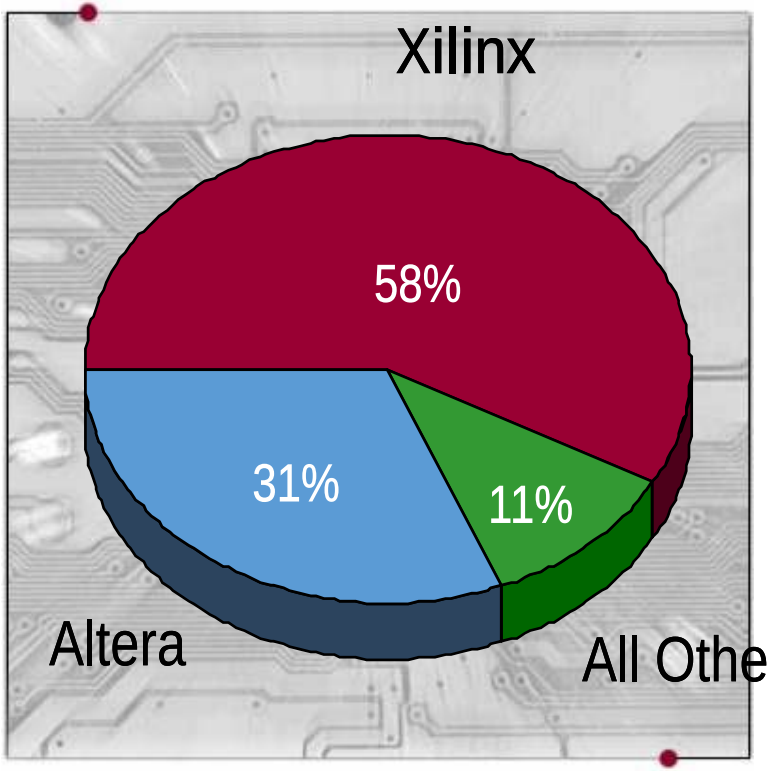
The Programmable Marketplace

Q1 Calendar Year 2005

PLD Segment



FPGA Sub-Segment



Two dominant suppliers, indicating a maturing market

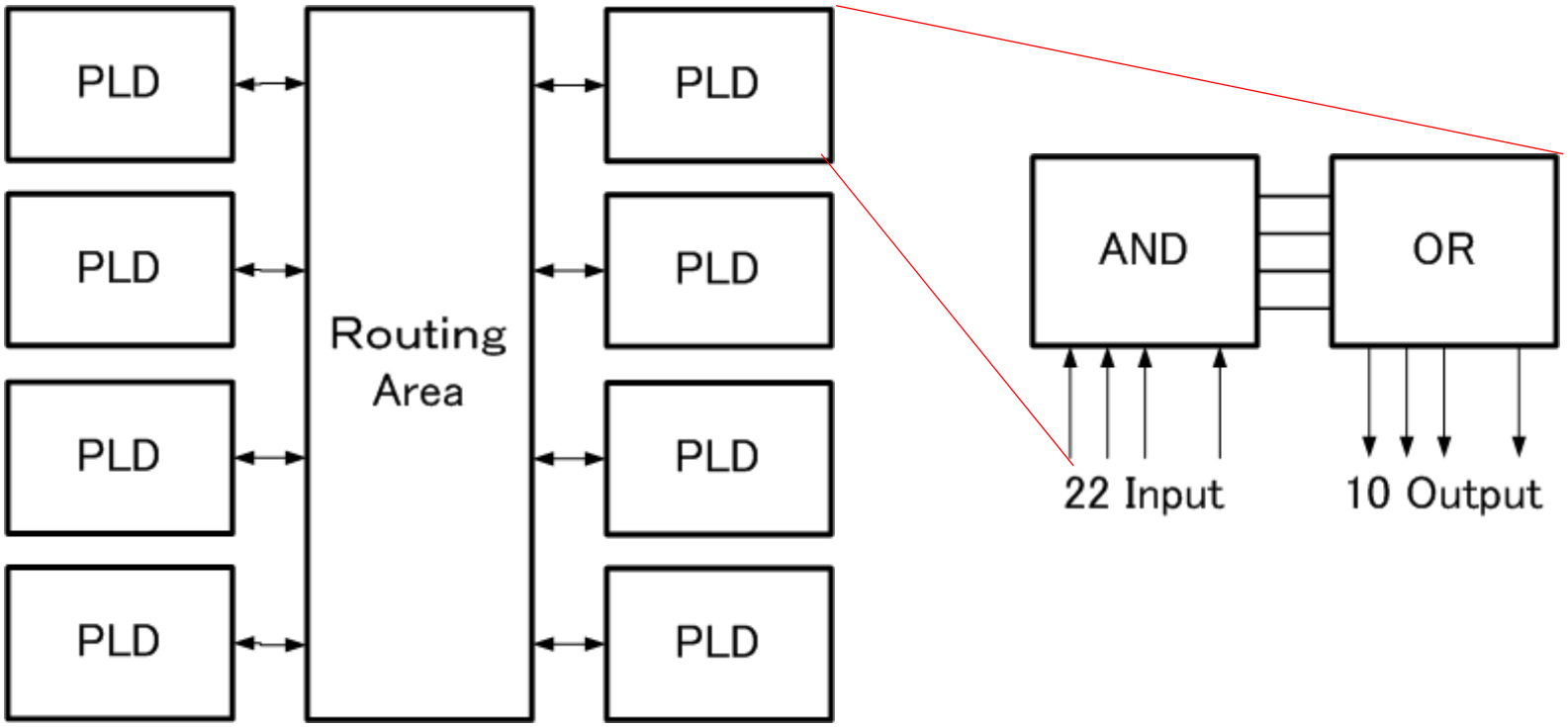
Source: Company reports
Latest information available; computed on a 4-quarter rolling basis

FPGA Families

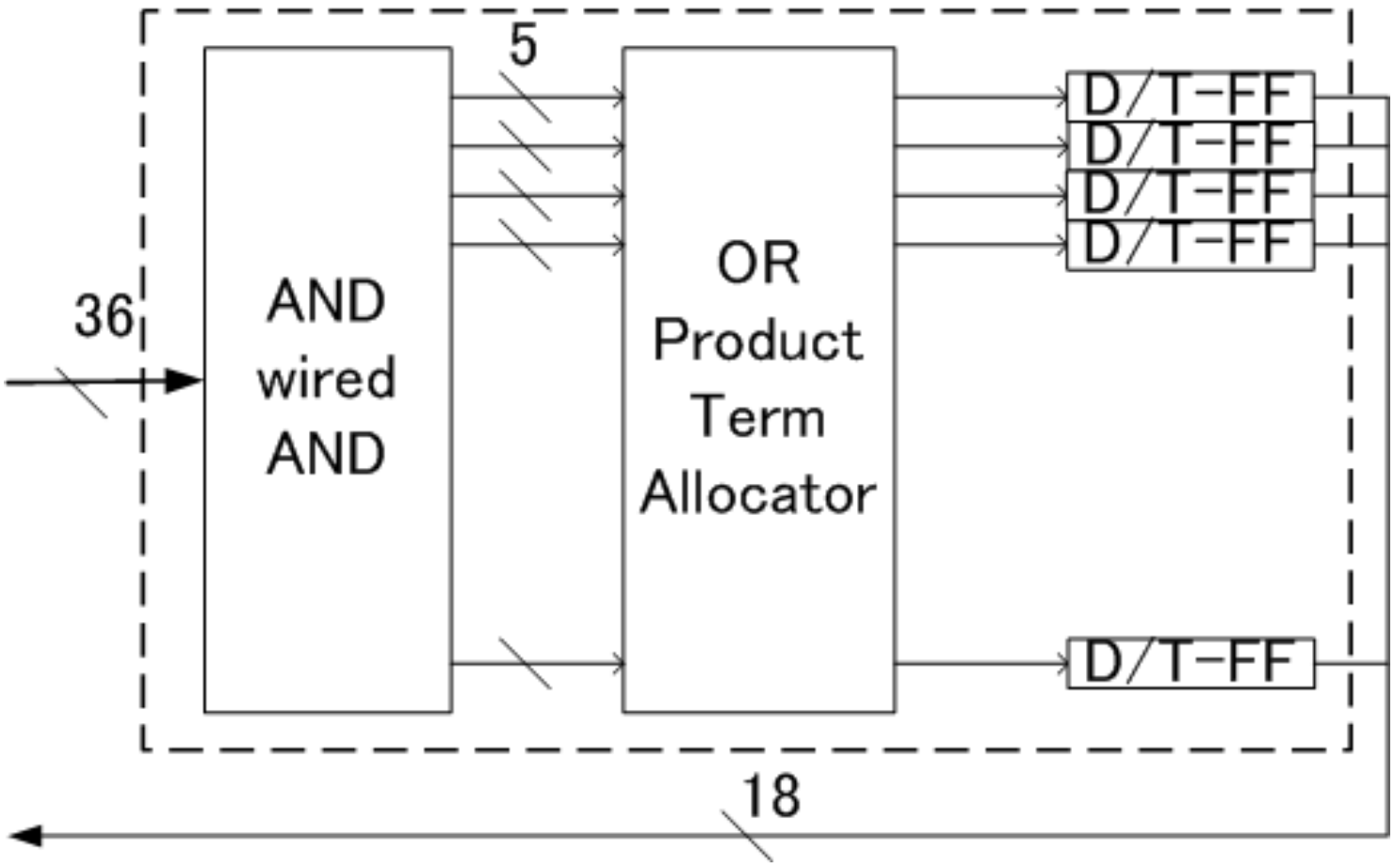
	Low-cost	Middle-end	High-Performance	Embedded
Xilinx Inc.	Artix-7 Spartan-7	Kintex-7 Kintex Ultra Scale	Virtex-7 Virtex Ultra Scale	Zynq
Altera Corp.	Cyclone V	Arria 10	Stratix V Stratix X	Cyclone SoC

1.2 Programmable Logic Device (PLD)

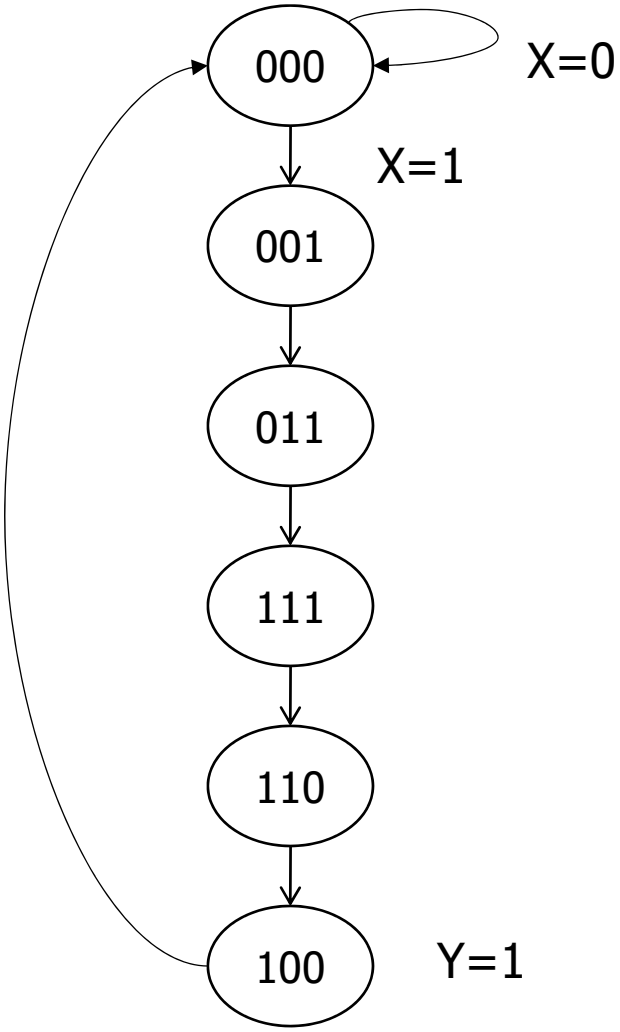
Complex PLD (CPLD)



Function Block in CPLD



Example of PLD Design

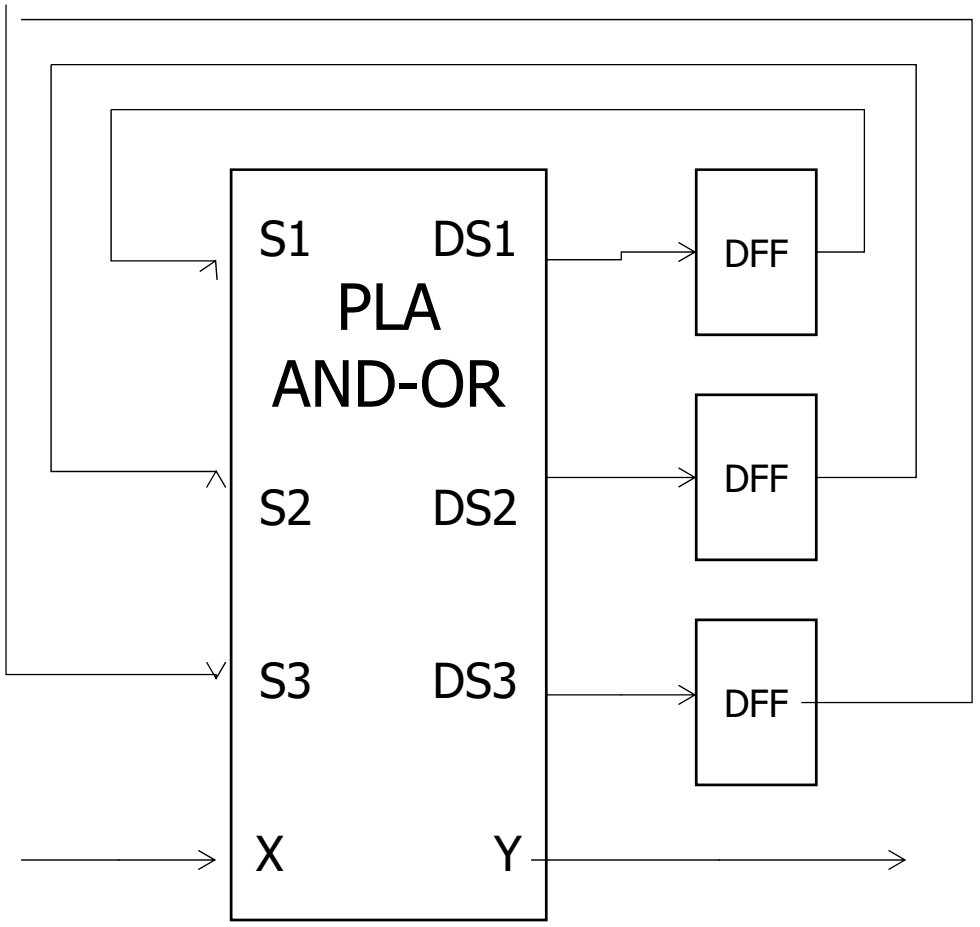


State Transition Table

S1	S2	S3	X	S1	S2	S3	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	*	0	1	1	0
0	1	1	*	1	1	1	0
1	1	1	*	1	1	0	0
1	1	0	*	1	0	0	0
1	0	0	*	0	0	0	1

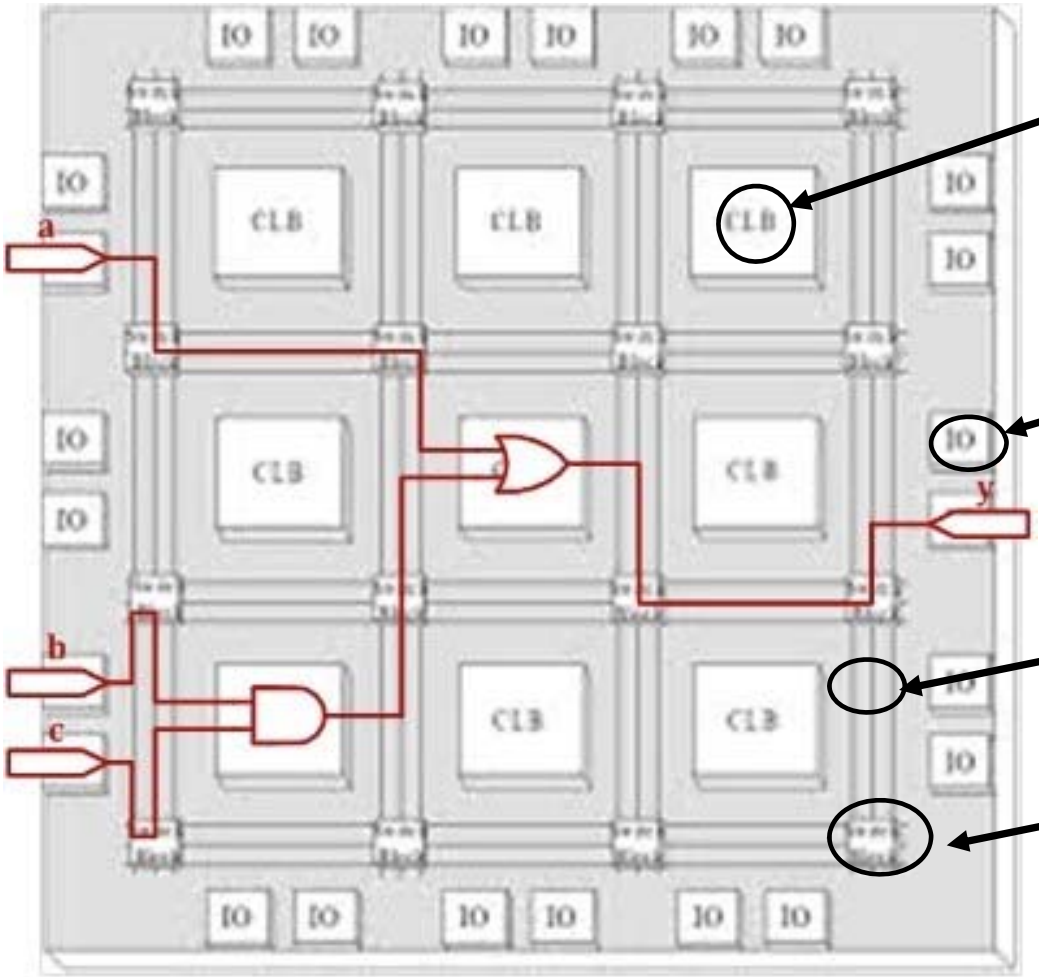
PLA & FF Realization

S1	S2	S3	X	DS1	DS2	DS3	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	*	0	1	1	0
0	1	1	*	1	1	1	0
1	1	1	*	1	1	0	0
1	1	0	*	1	0	0	0
1	0	0	*	0	0	0	1



1.3 FPGA

Island Style FPGA



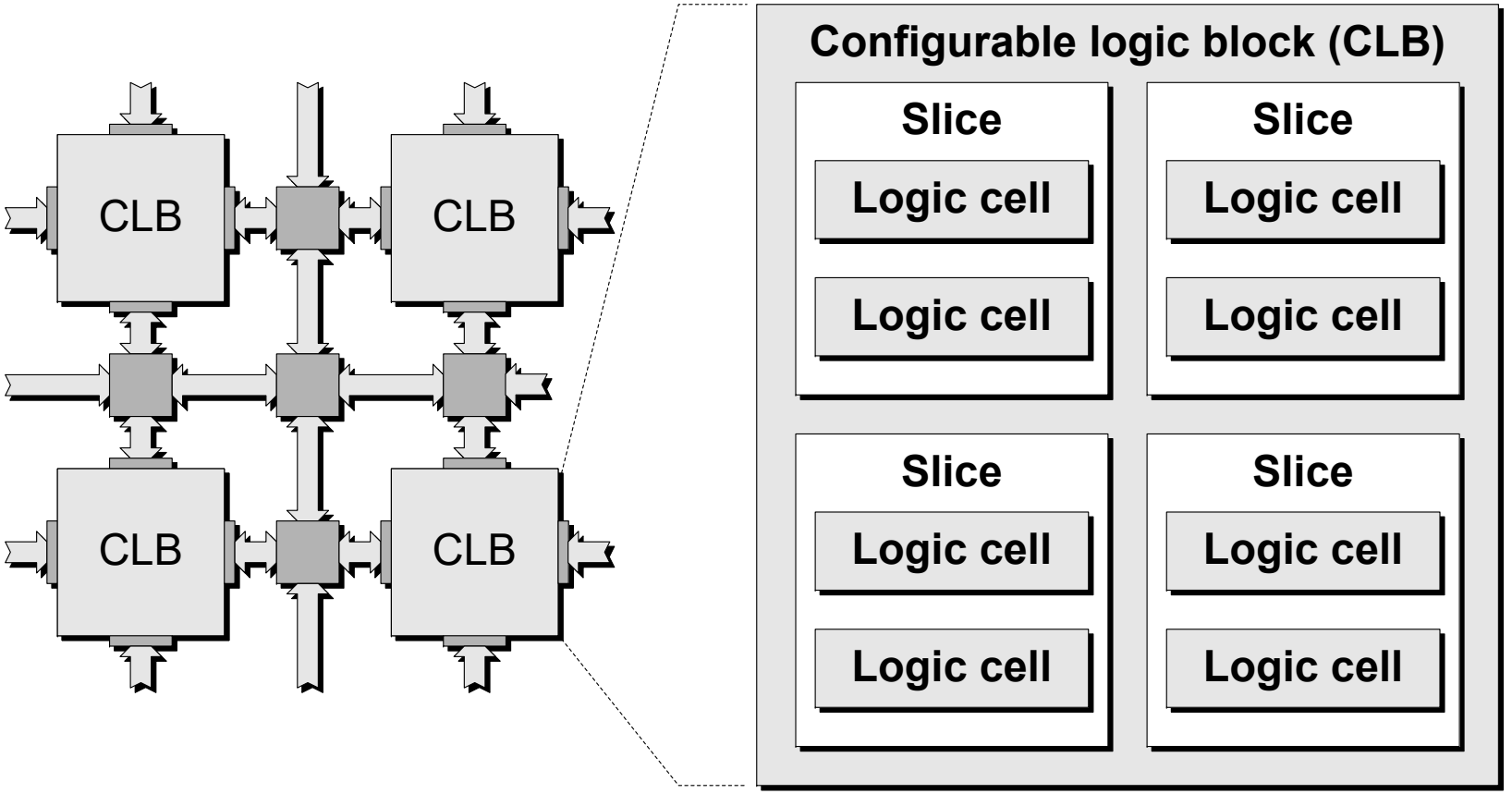
Configurable Logic Block (CLB)

I/O Block

Channels

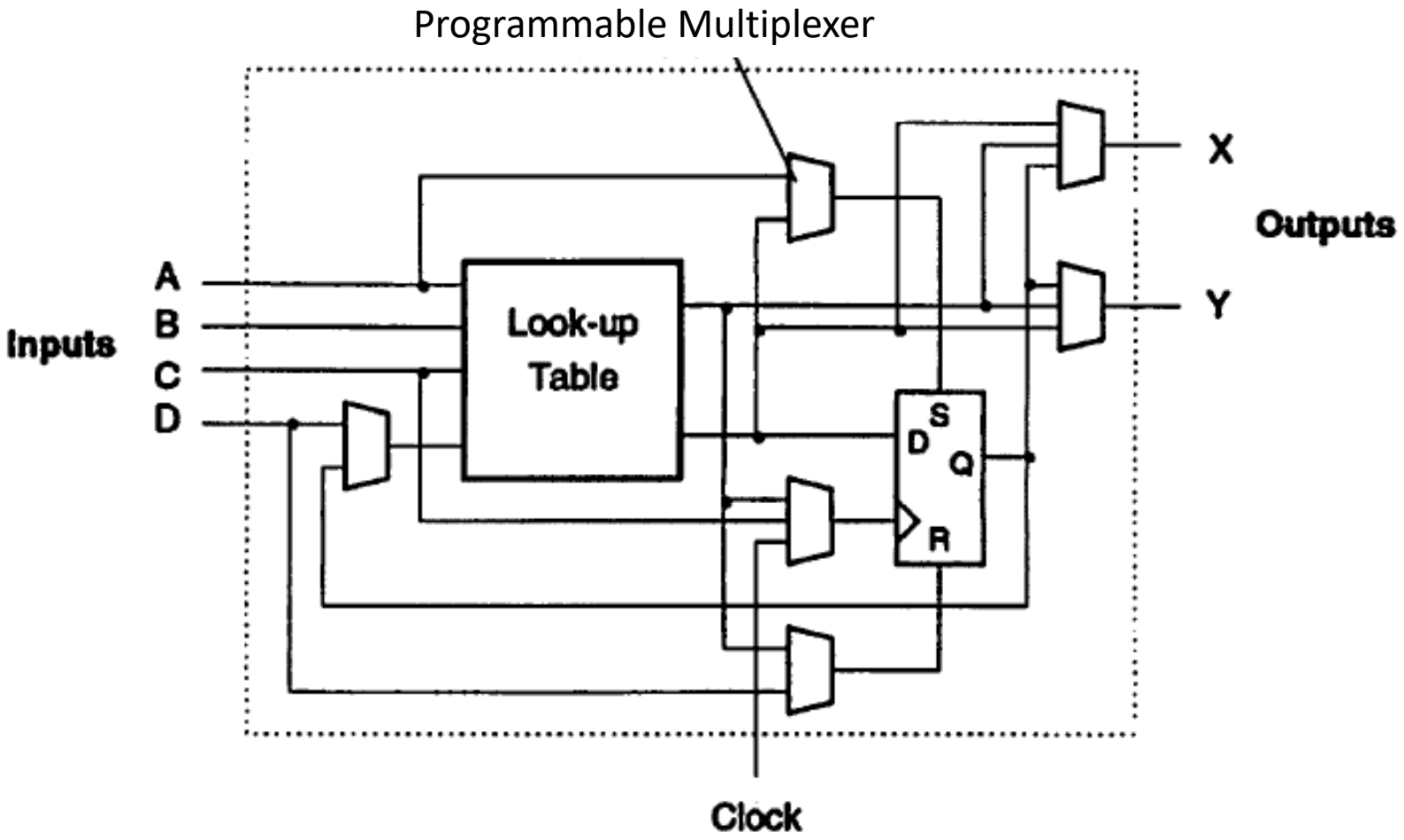
Switch Blocks

Xilinx CLB



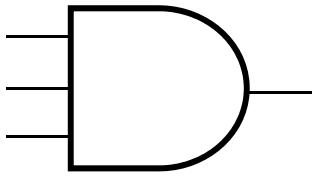
The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043

Logic Cell (Xilinx Inc. XC2000)

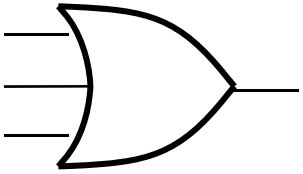


Realization of a Logic Function

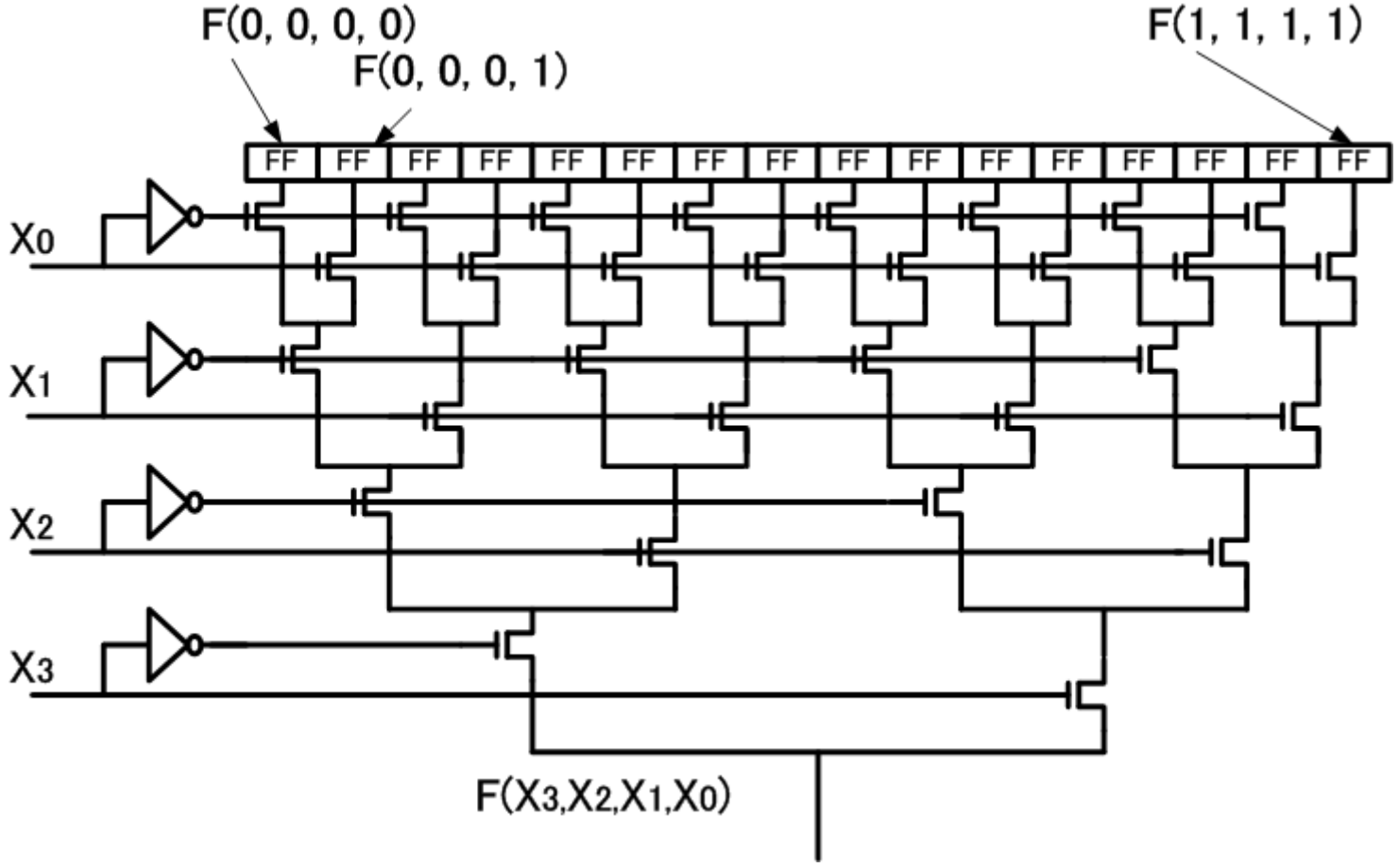
x0	x1	x2	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



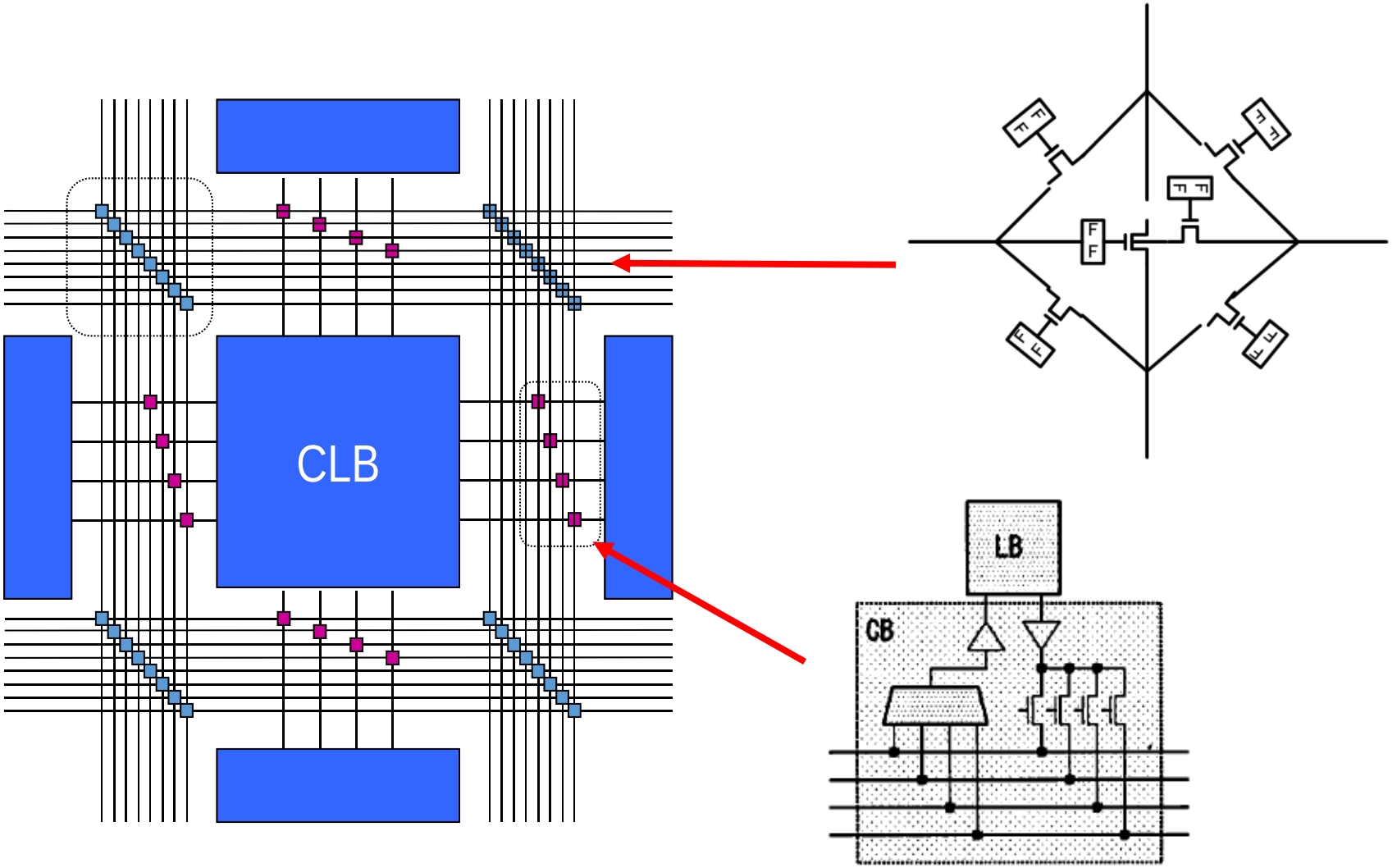
x0	x1	x2	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



LUT Structure



Channel and Switch Block



Variation of a Switch Block

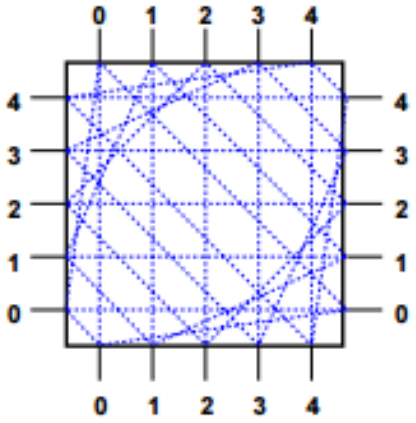


Figure 3.2(a): Disjoint

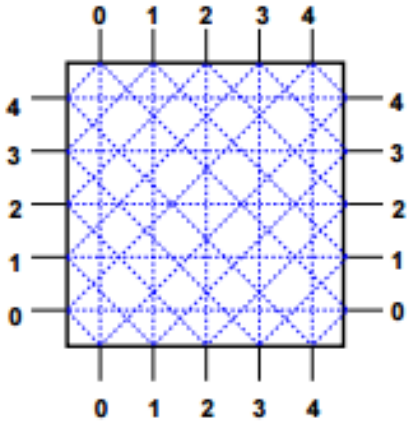


Figure 3.2(b): Universal

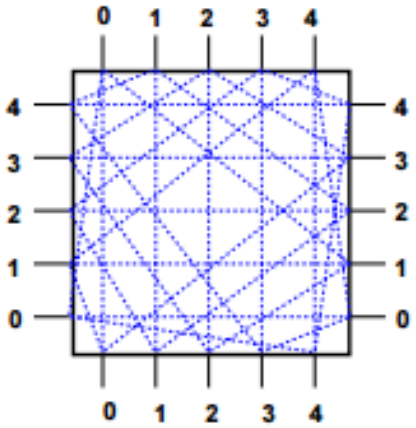
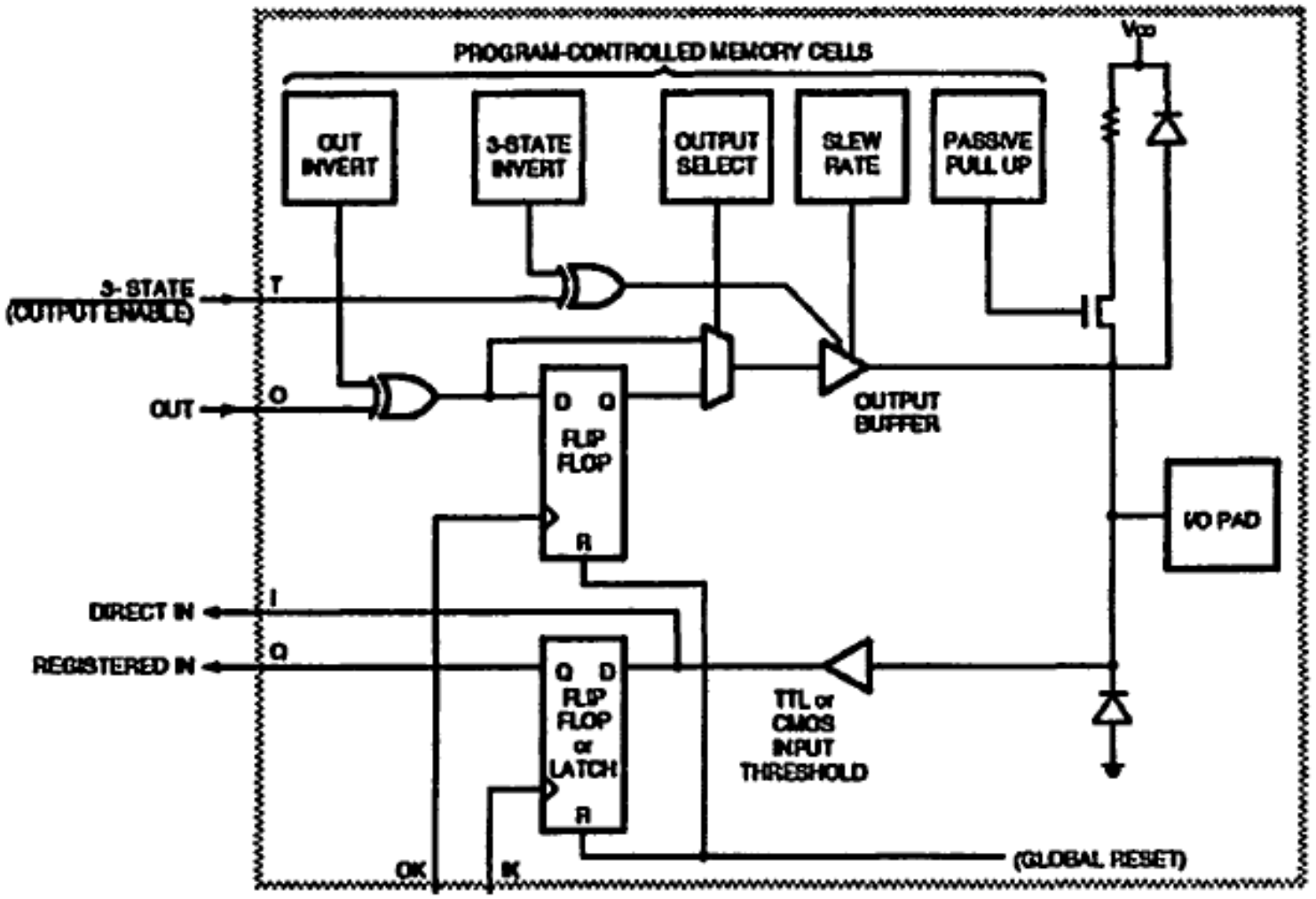


Figure 3.2(c): Wilton

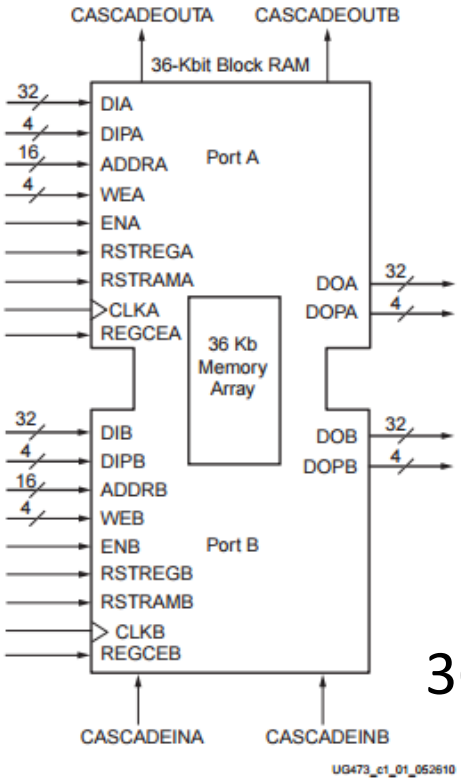
Steven J. E. Wilton, “Architecture and Algorithms for Field Programmable Gate Arrays with Embedded Memory,” Phd thesis, University of Toronto, 1997.

I/O Block

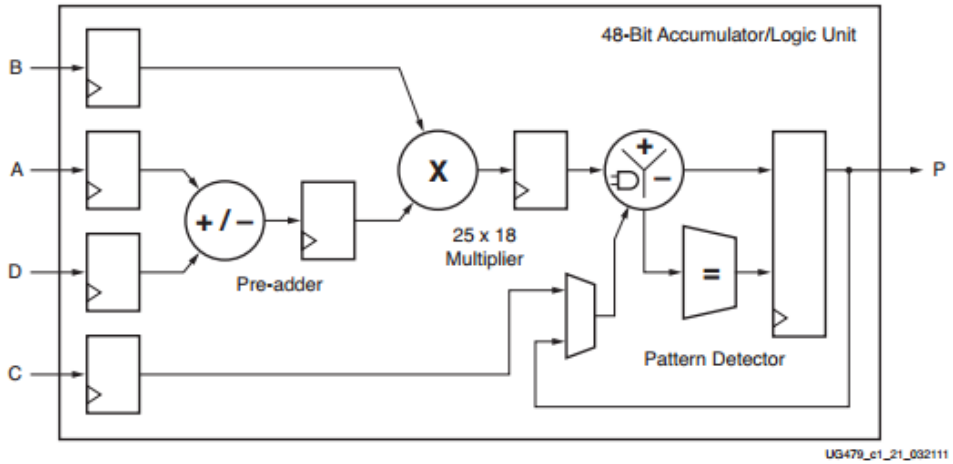

PROGRAM-CONTROLLED MEMORY CELLS
OUT INVERT
3-STATE INVERT
OUTPUT SELECT
SLEW RATE
PASSIVE PULL UP
3-STATE (OUTPUT ENABLE)
T
OUT
O
D Q
FLIP FLOP
R
OUTPUT BUFFER
I/O PAD
DIRECT IN
I
REGISTERED IN
Q
Q D
FLIP FLOP or LATCH
R
TTL or CMOS INPUT THRESHOLD
OK
IK
(GLOBAL RESET)

Hard macro (Dedicated Circuit)

- Standard IP cores are “Pre-built” in the FPGA

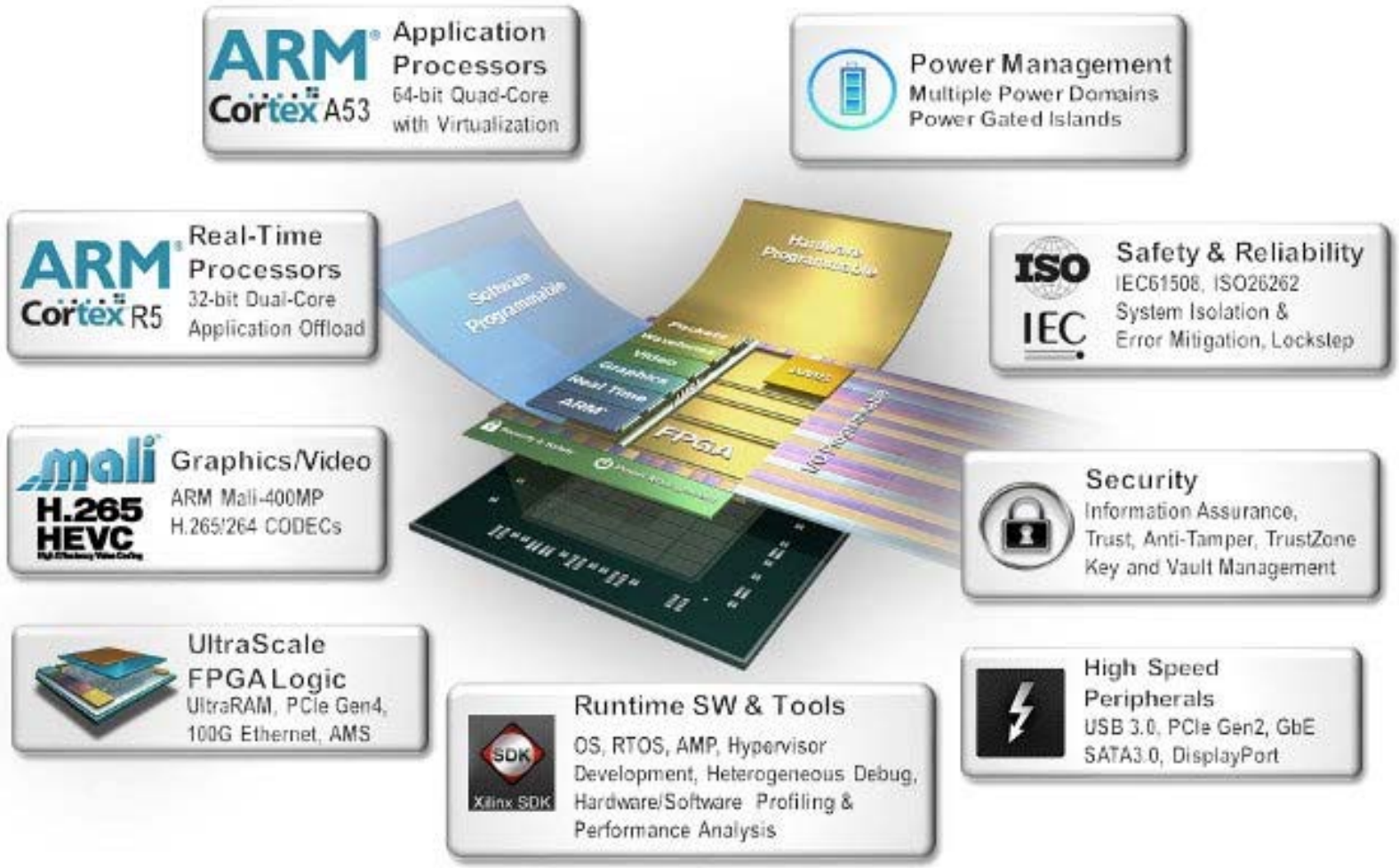


36Kb Block Memory



48bit DSP Block

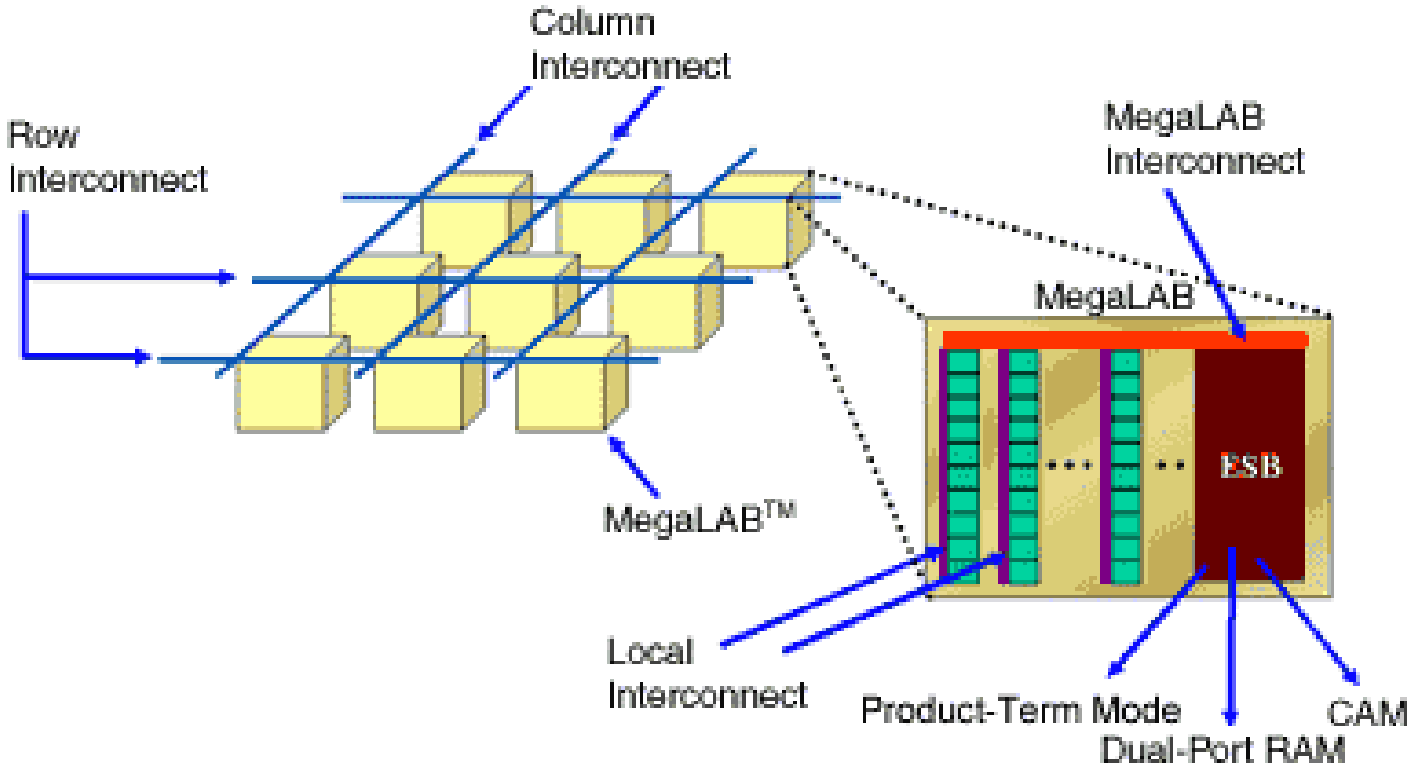
Xilinx Zynq Ultra Scale+



Pros. and Cons.

- Pros.
  - Short TAT(Turn-Around Time)
  - Small NRE (Non Recurrent Expense) Fee
  - Logic and Timing Design are required.
  - Full amount of IP (Intellectual Property)
- Cons.
  - Slow speed and Large Chip Area
  - High cost for volume manufacturing

Altera FPGA (Hierarchical Structure)

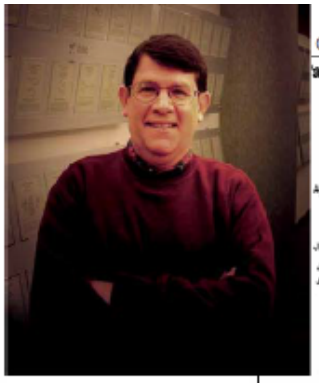


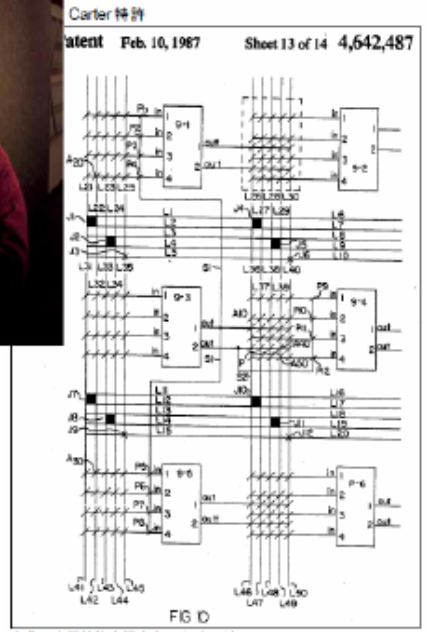
FPGA-world Famous Patents

US RE34363
by Ross H. Freeman
Island style architecture

出典：米国特許商標庁ホームページ

US 464248
by William S. Carter
Connection architecture

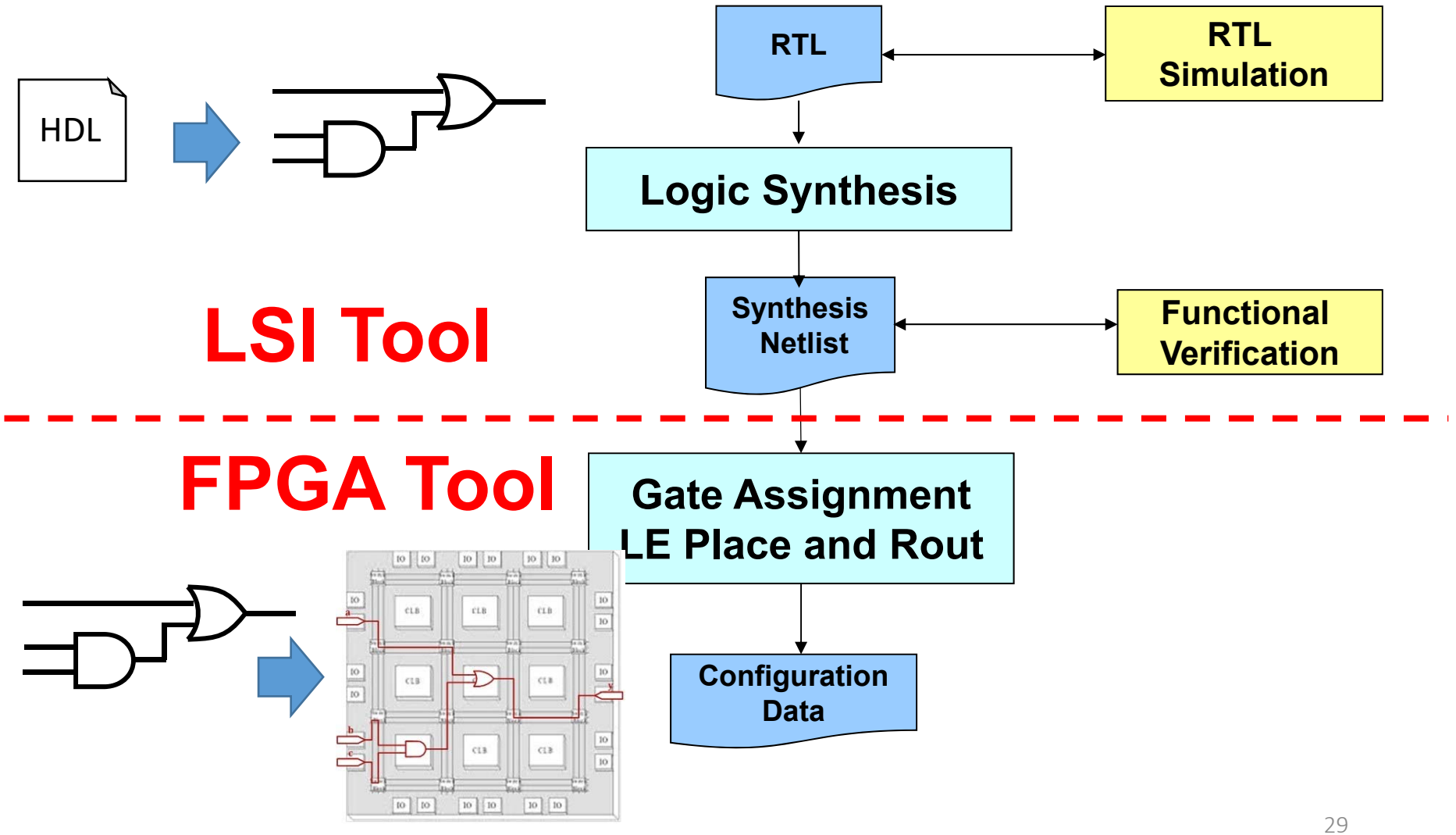




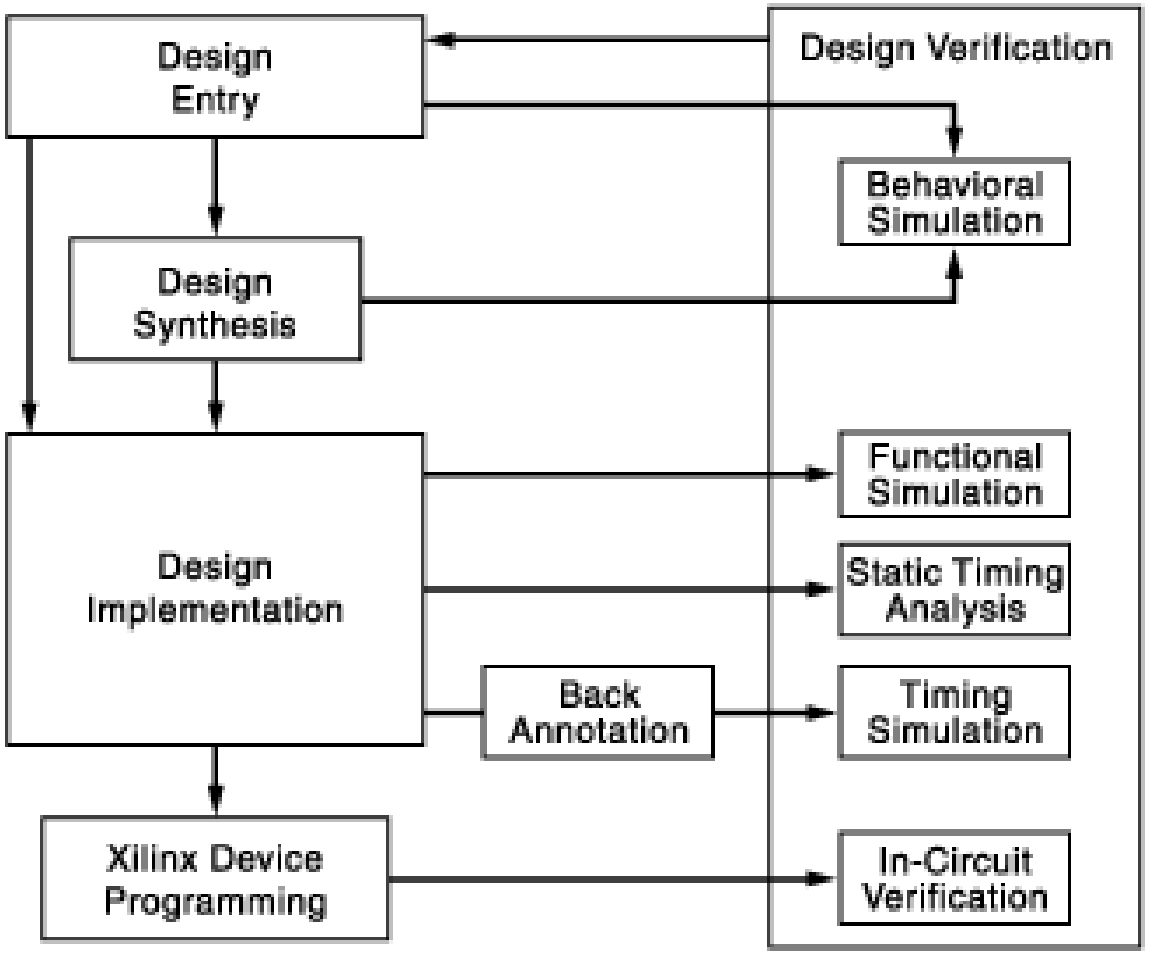
出典：米国特許商標庁ホームページ

2. Standard FPGA Design

Standard FPGA Design

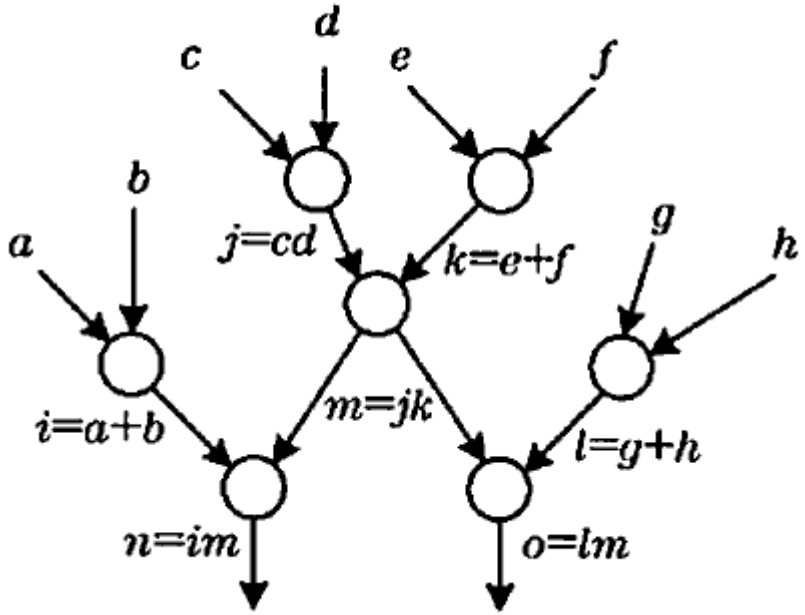


FPGA Design Flow



Boolean Network

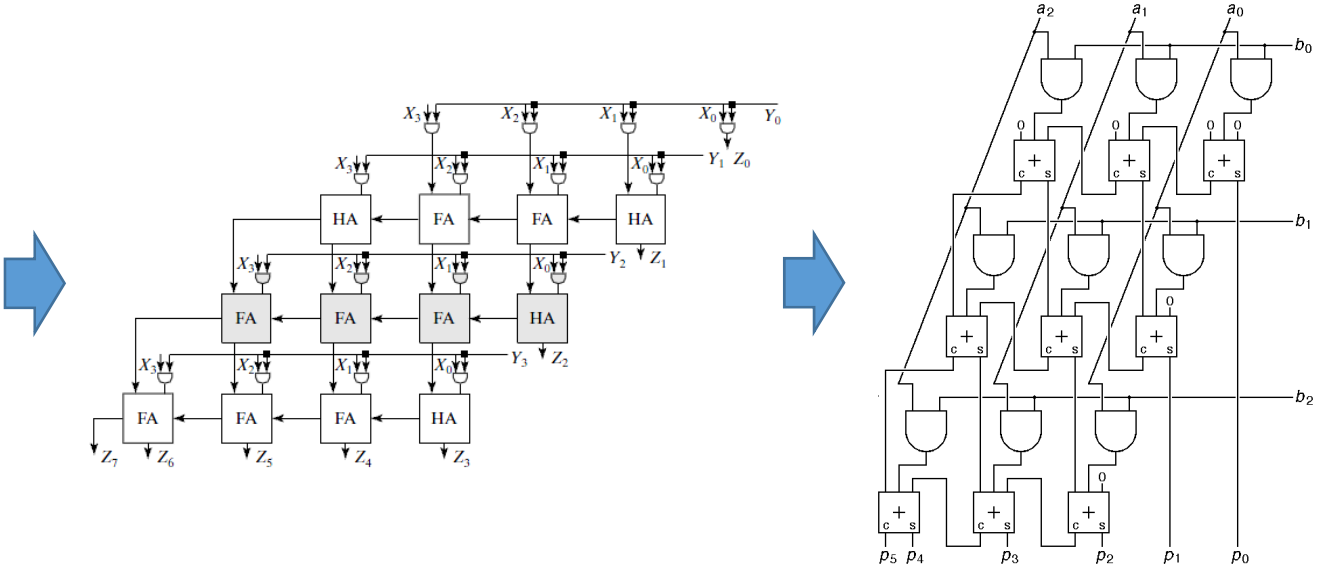
- Representation of a combinational logic circuit using a directed graph without a cycle
- Vertex：Logic gate, Edge：Input or output



Logic Synthesis

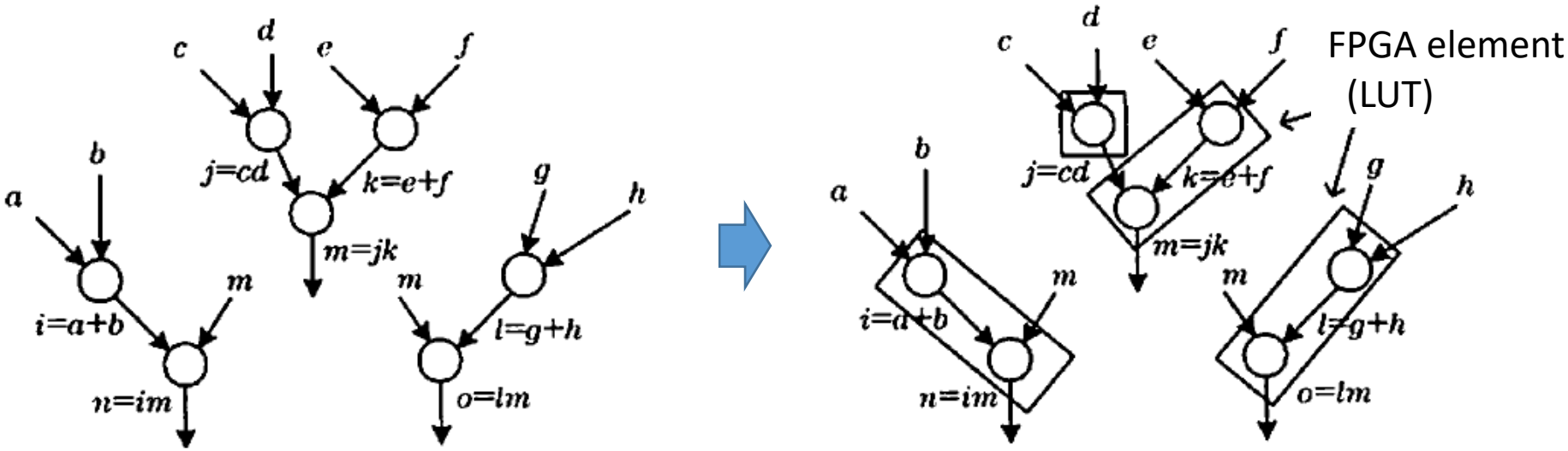
- Synthesize from a given HDL specification to a Boolean network

input [3:0]X,Y;
output [7:0]Z;
Z=X*Y



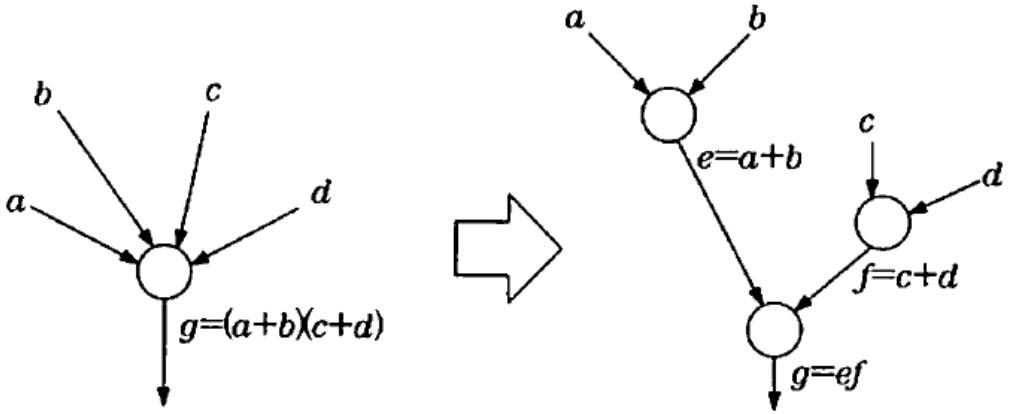
Technology Mapping

- A kind of a graph covering problem
- Goal: A depth optimized one by using a dynamic programming

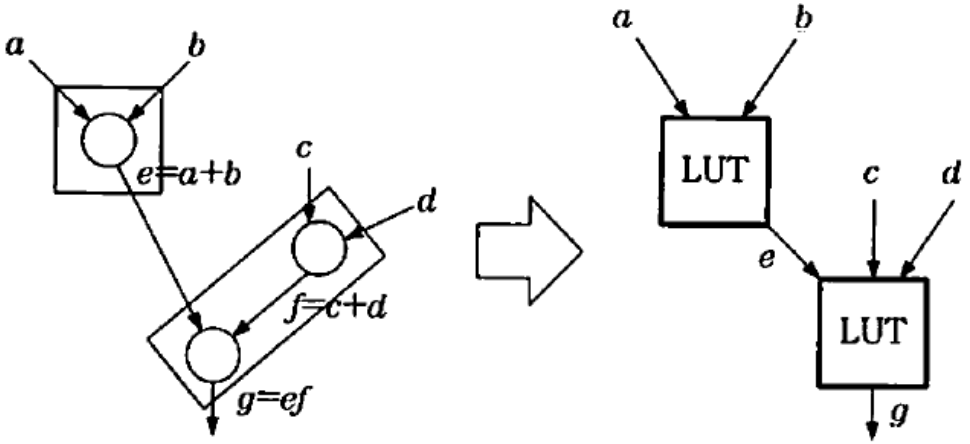


Decomposition and Covering

Decomposition

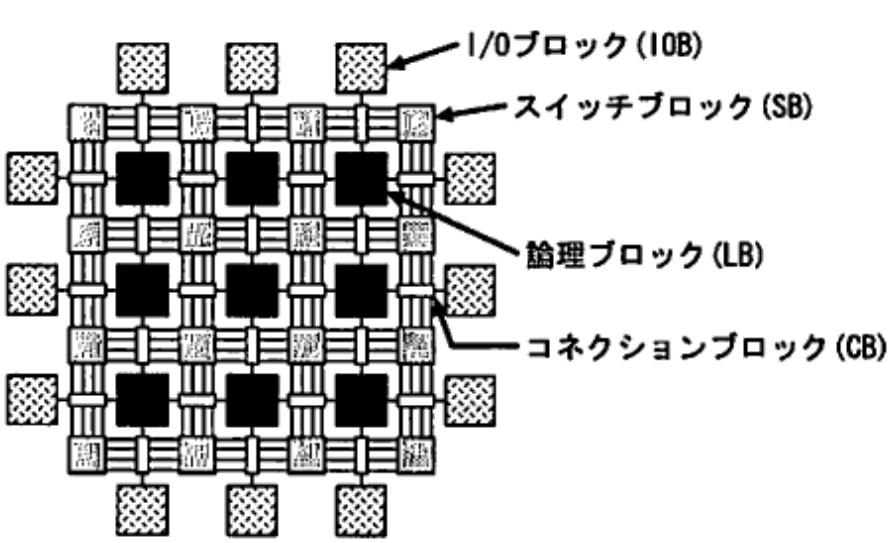


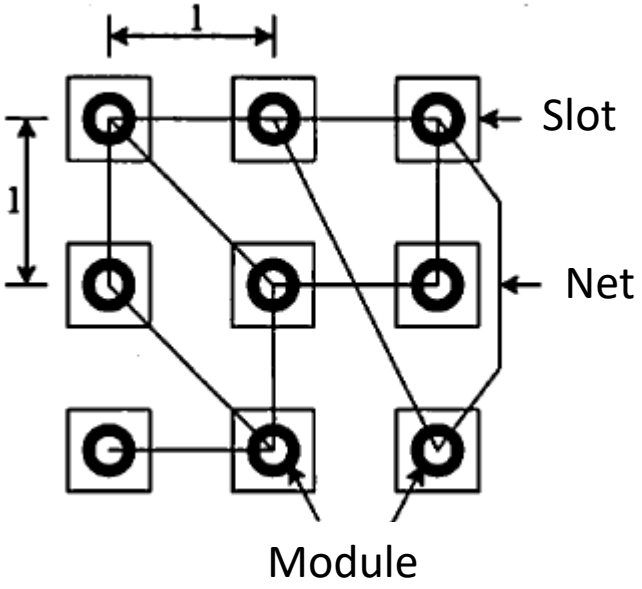
Covering



Placement

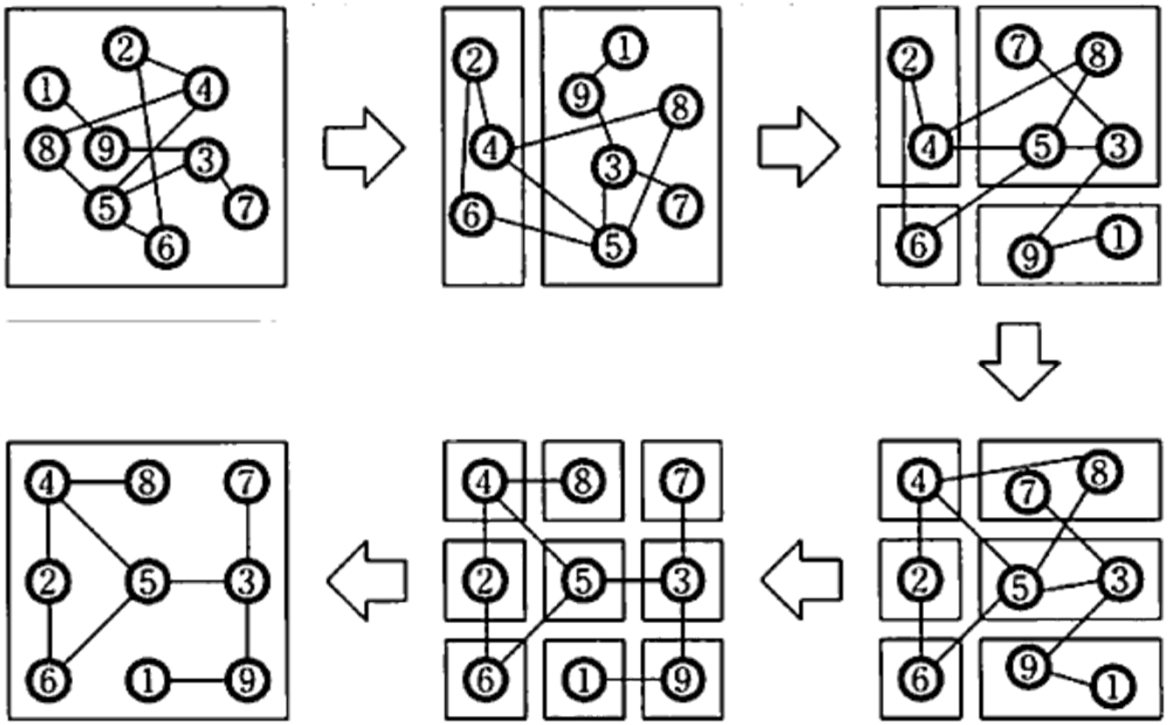
- Problem to place the module (logic gate) into the slot (location)
  - 2D allocation problem → NP-complete
  - Approximation (Simulated annealing, or min-cut tech.)





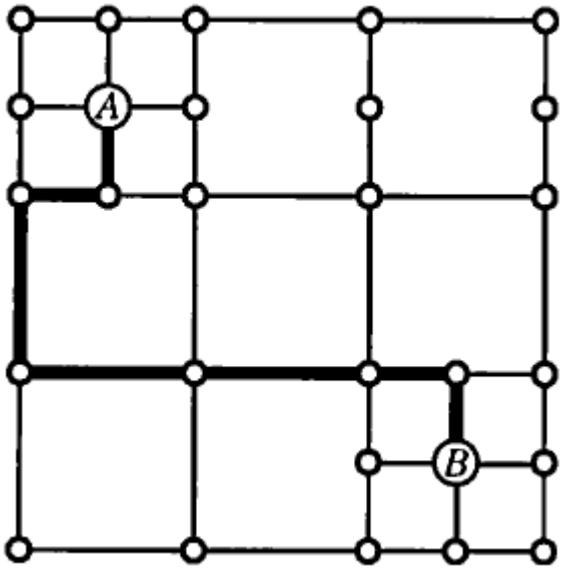
Min-cut Placement

- Recursively divided a module set into two ones
  - Near-optimal solution can be obtained in the short time
- Used in a commercial FPGA design tool

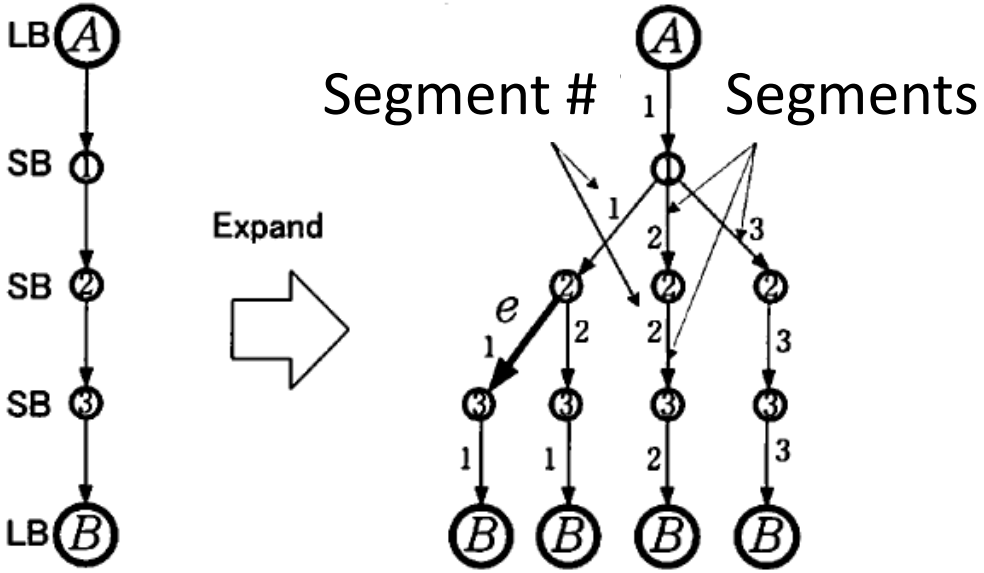


Routing

- Global routing: Determine the rough wiring path
- Local one: Determine the wiring segment and switch



Global routing



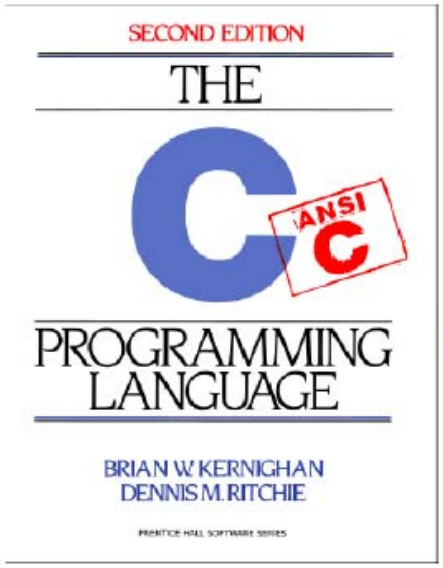
Local routing

High Level Synthesis (HLS) for an FPGA

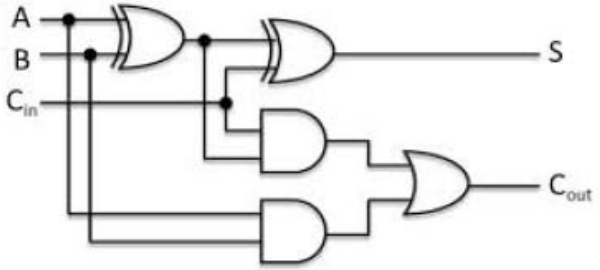
FPGA Potential

- Implementing computations in hardware can have speed/energy advantages over software:
  - Lithography simulation: 15x speed-up [Cong & Zhou, TRETS'09]
  - Linear system solver: 2.2x speed-up, 5x more energy efficient [Zhang, Betz, Rose, TRETS'12]
  - Monte Carlo simulation for photodynamic therapy: 80x faster, 45x more energy efficient [Lo et al., J. Biomed Optics'09]
  - Option pricing: 4.6x faster, 25x more energy efficient [Tse, Thomas, Luk, TVLSI'12]

Two Ways Computations



Write Software



Design Custom Circuits

Hardware Design is Difficult

- FPGA “success stories” are pervasive, yet the technology remains inaccessible to software engineers
  - Requires use of hardware description languages: Verilog and VHDL
- Hardware design skills are rare:
  - 10 software engineers for every hardware engineer*

↓

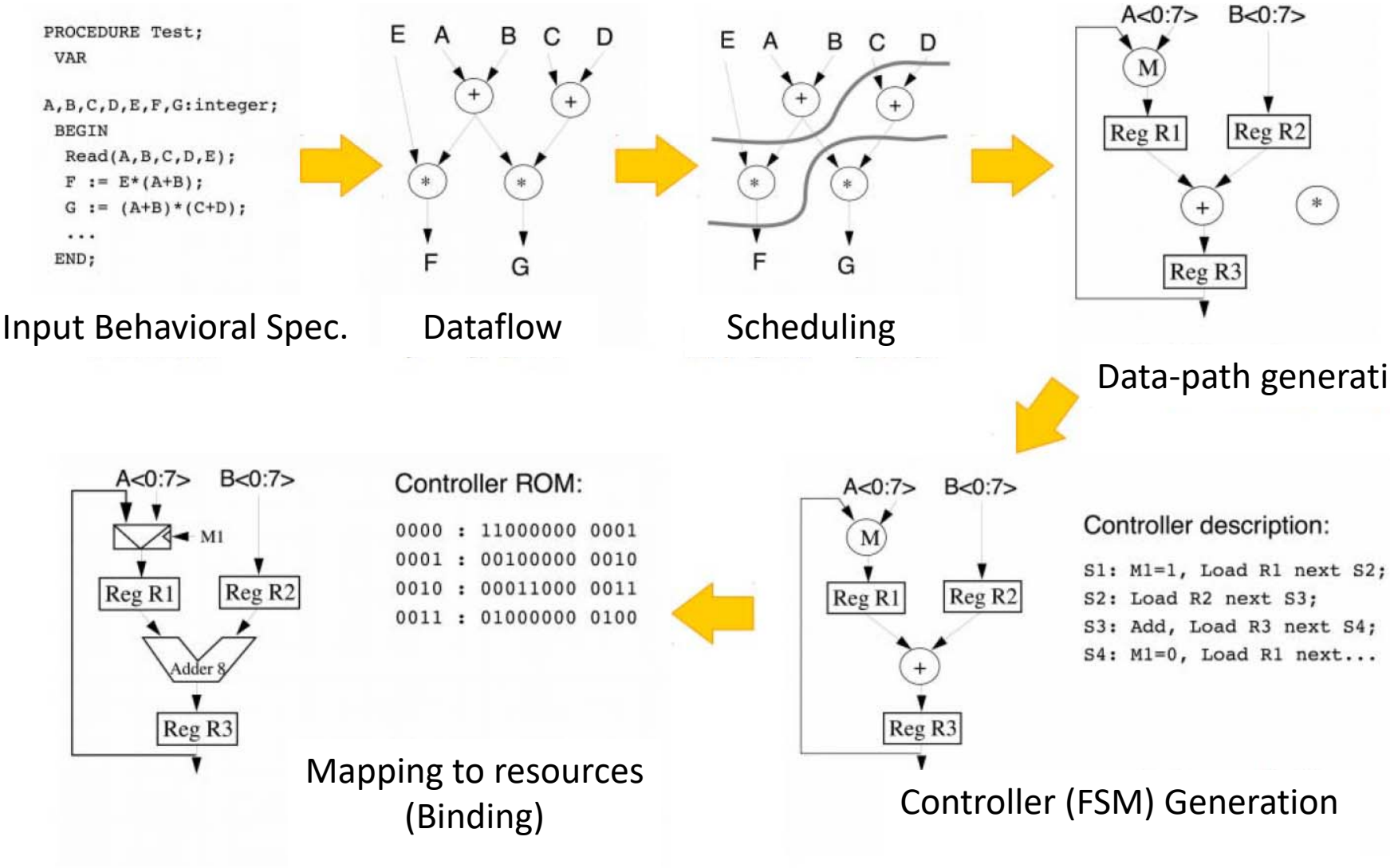
- Make hardware design easier for hardware engineers
- Allow software engineers to design hardware and reap its energy/performance benefits

*Source: US Bureau of Labor Statistics, 2012

High-level synthesis (HLS) with FPGAs

- Plays a central role, enabling the automatic synthesis of high-level, untimed or partially timed specifications (e.g. C or System C) to low-level cycle accurate RTL specifications
- Target devices includes application-specific integrated circuits (ASIC) or field-programmable gate arrays (FPGAs)
- It can be optimized taking into account the performance, power, and cost requirements

High-level Synthesis Flow



Input Behavioral Spec. Dataflow Scheduling

Data-path generation

Controller (FSM) Generation

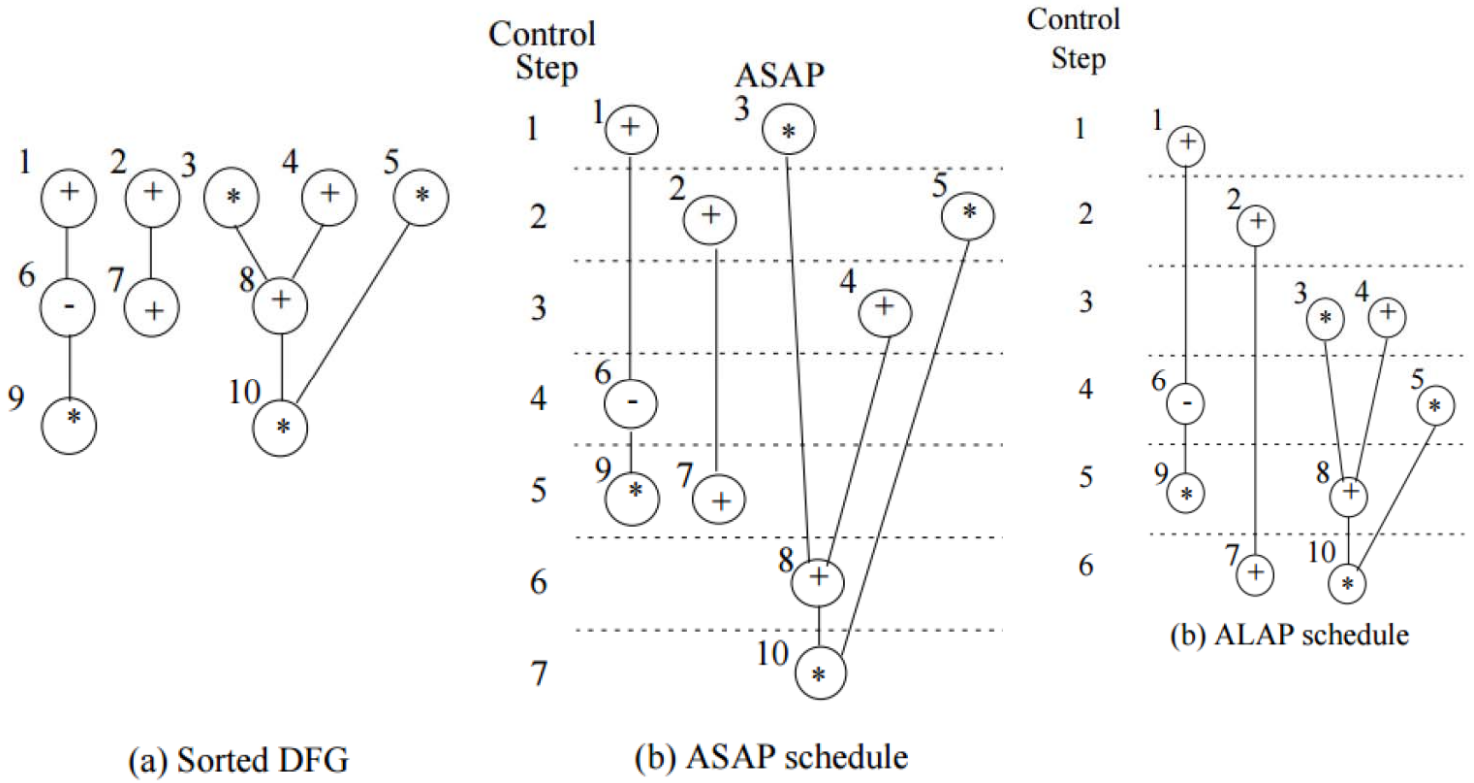
Mapping to resources (Binding)

Scheduling

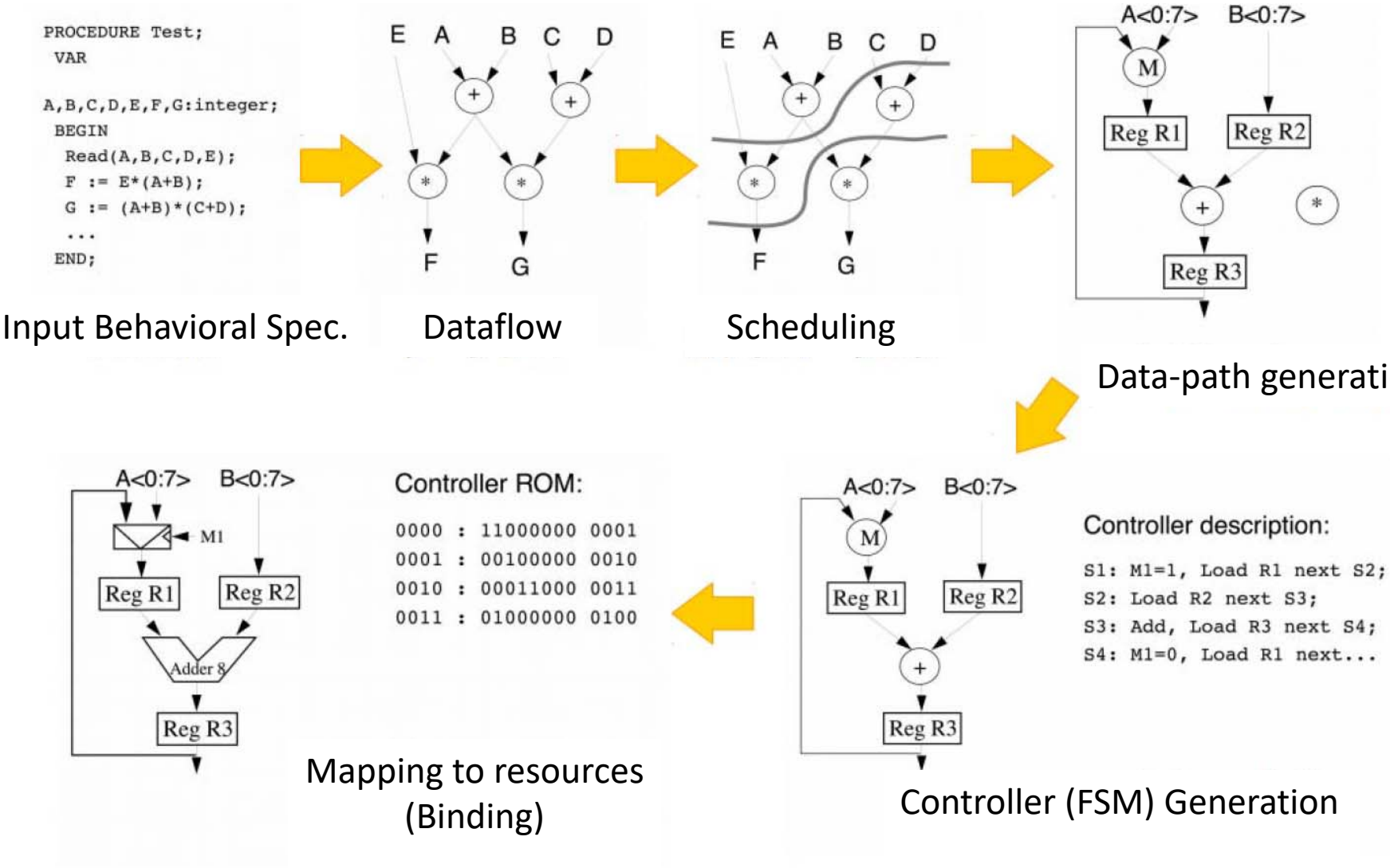
- How to assign the computations of a program into the hardware time steps (clock cycles)?
- Must consider under given target clock period:
  - Which operations can be scheduled in the same time step?
  - Which operations are dependent on others?

Schedule strategies

- As soon as possible (ASAP)
- As late as possible (ALAP)



High-level Synthesis Flow



Input Behavioral Spec. Dataflow Scheduling

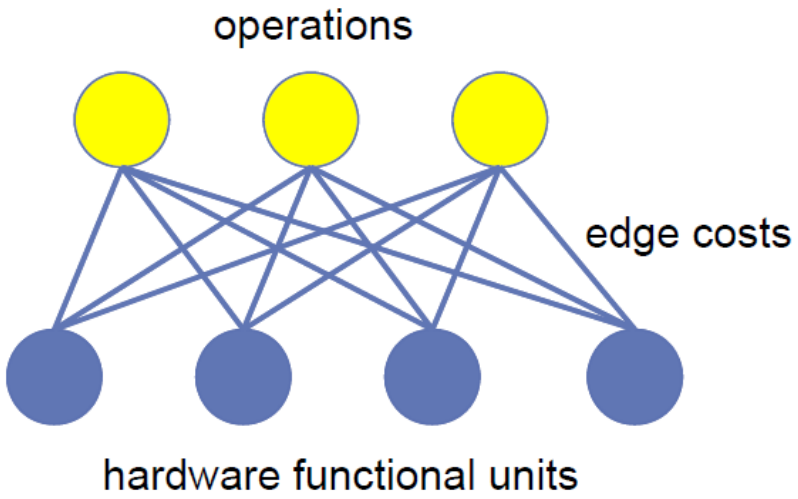
Data-path generation

Controller (FSM) Generation

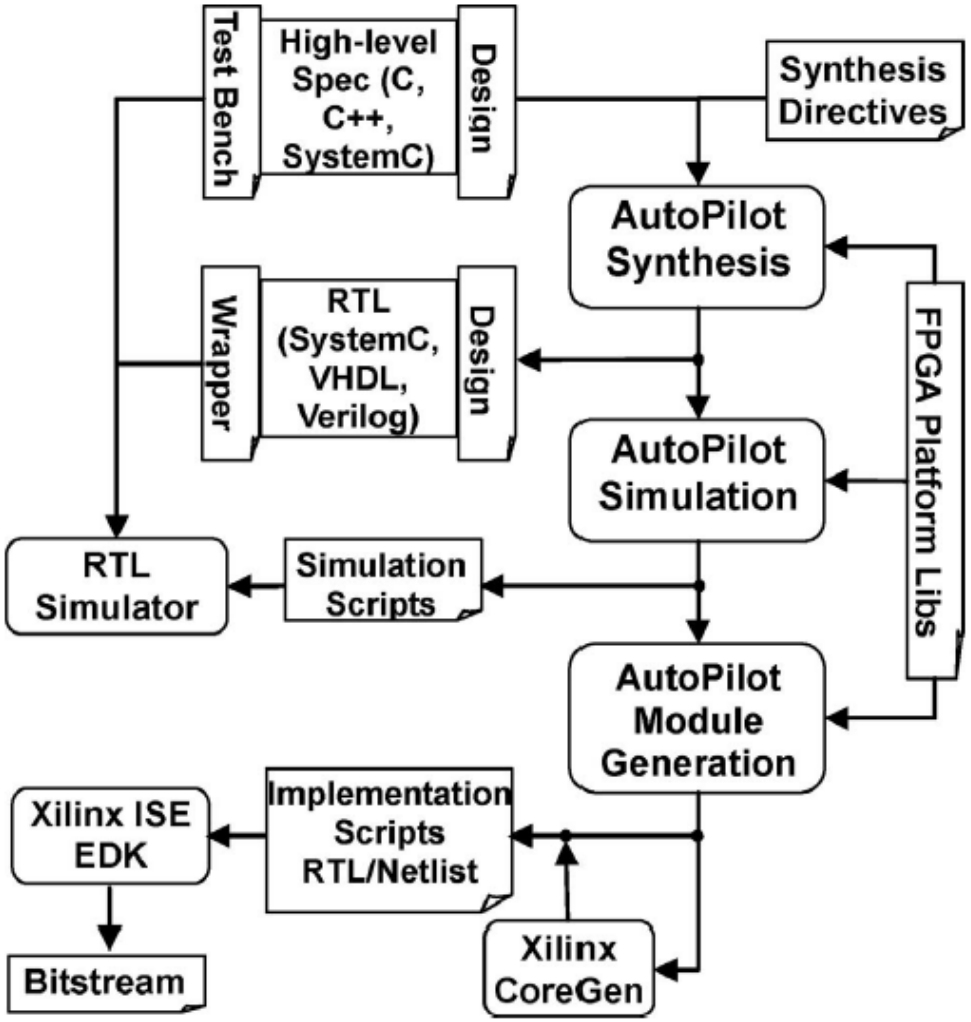
Mapping to resources (Binding)

Binding

- Assign (bind) each operation to a hardware functional unit
- Example:
  - LegUp uses bipartite matching-based binding [Huang DAC'90] using the Hungarian Method (Polynomial order)

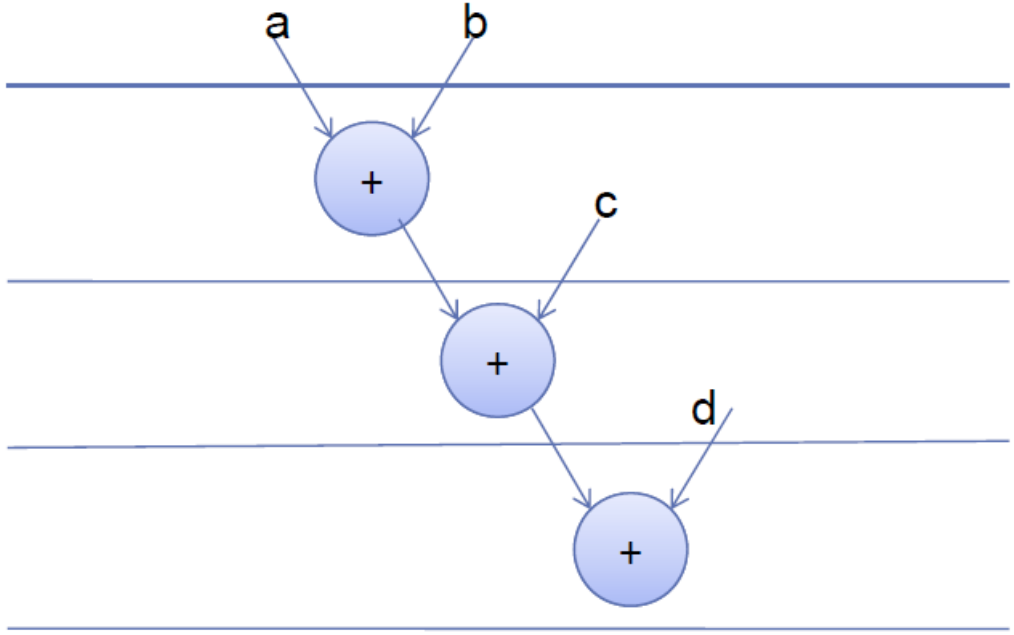


AutoESL and Xilinx C-to-FPGA design flow



Performance Feature: Loop Pipelining

```
for (int i = 0; i < N; i++)
    sum[i] = a + b + c + d
```



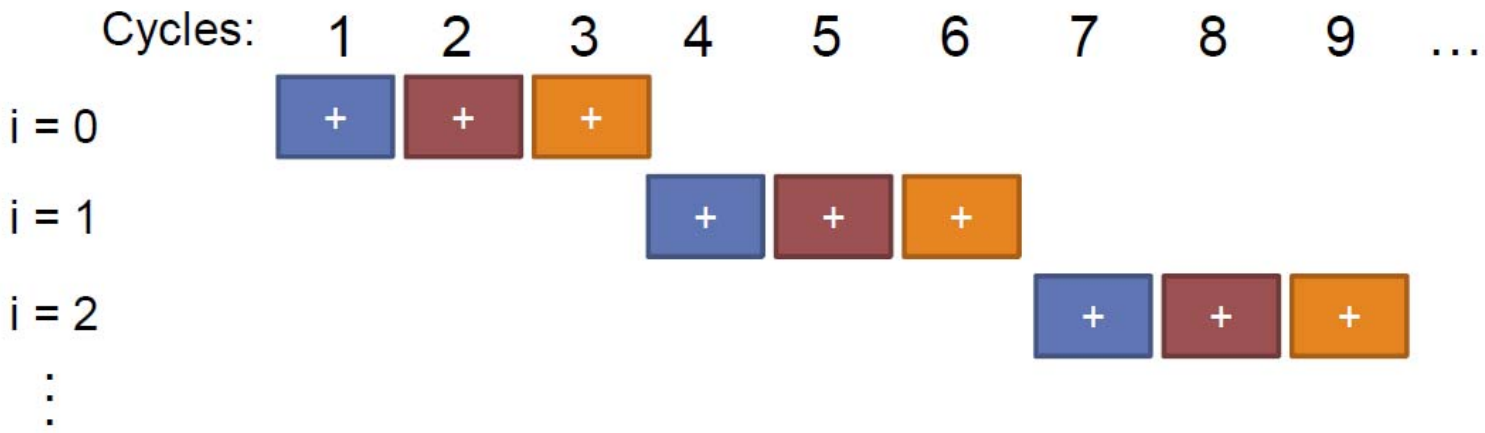
Sequential Execution

```
for (int i = 0; i < N; i++)
      sum[i] = a + b + c + d
```

Not efficient!

- 3 Cycles/Iteration
- Total Cycles: 3N
- Adders: 3
- Adder Utilization: 33%

1. Sequential Execution

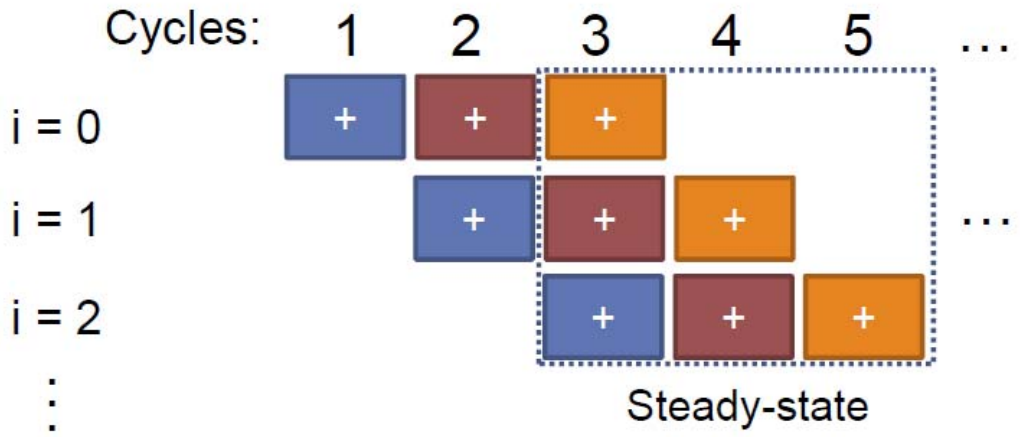


Poop Pipelining

#pragma pipeline

```
for (int i = 0; i < N; i++)
    sum[i] = a + b + c + d
```

3. Parallel Execution : Loop pipelining



Efficient!

- 1 Cycles/Iteration (steady-state)
- Total Cycles: N+2
- Adders: 3
- Adder Utilization: 100%

High-quality and rapid design using HLS: Why?

- Embedded processors are in almost every SoC
- Huge silicon capacity requires a higher level of abstraction
  - With the HLS, the code density can be easily reduced by 7x-10x in C/C++
- Behavioral IP reuse improves design productivity
- Verification drives the acceptance of HLS
  - It avoids slow and error-prone manually RTL cording
- Trend toward extensive use of accelerators and heterogeneous SoCs

Faster Adoption of HLS Tools with FPGAs: Why?

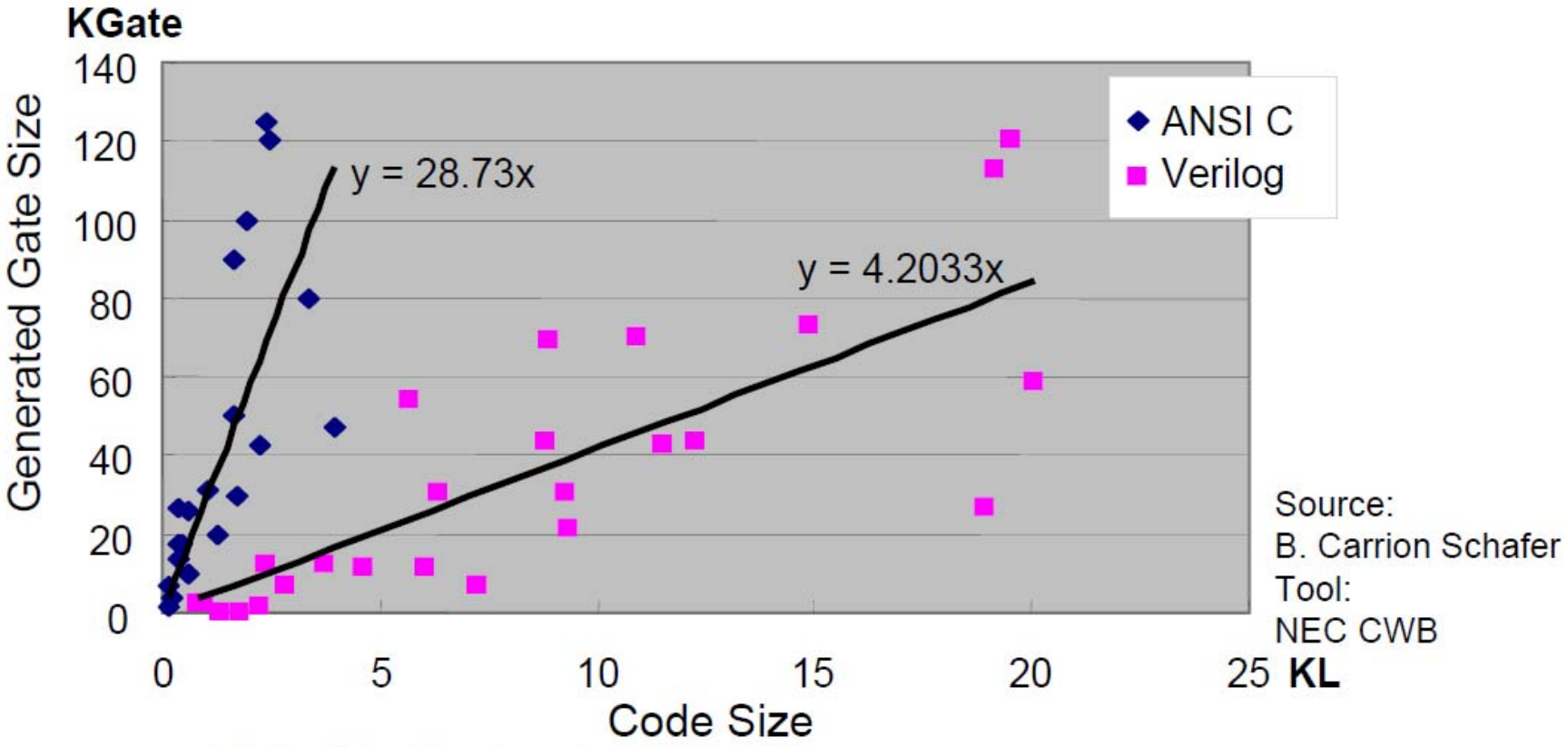
- Less pressure for formal verification
  - Iterations of the FPGA design can avoid huge manufacturing costs
- Ideal for platform-based synthesis
  - Achieve higher quality of results (QoR)
- More pressure for time-to-market
  - Designers may accept increased performance, power, or cost in order to reduce design time
- Accelerated or reconfigurable computing calls for C/C++ based compilation/synthesis to FPGAs

Overview of HLS Tools

Status	Compiler	Owner	License	Input	Output	Year	Domain	TestBench	FP	FixP
In Use	eXCite	Y Explorations	Commercial	C	VHDL/Verilog	2001	All	Yes	No	Yes
	CoDeve-loper	Impulse Accelerated	Commercial	Impulse-C	VHDL Verilog	2003	Image Streaming	Yes	Yes	No
	Catapult-C	Calypto Design Systems	Commercial	C/C++ SystemC	VHDL/Verilog SystemC	2004	All	Yes	No	Yes
	Cynthesizer	FORTE	Commercial	SystemC	Verilog	2004	All	Yes	Yes	Yes
	Bluespec	BlueSpec Inc.	Commercial	BSV	SystemVerilog	2007	All	No	No	No
	CHC	Altium	Commercial	C subset	VHDL/Verilog	2008	All	No	Yes	Yes
	CtoS	Cadence	Commercial	SystemC TLM/C++	Verilog SystemC	2008	All	Only cycle accurate	No	Yes
	DK Design Suite	Mentor Graphics	Commercial	Handel-C	VHDL Verilog	2009	Streaming	No	No	Yes
	GAUT	U. Bretagne	Academic	C/C++	VHDL	2010	DSP	Yes	No	Yes
	MaxCompiler	Maxeler	Commercial	MaxJ	RTL	2010	DataFlow	No	Yes	No
	ROCCC	Jacquard Comp.	Commercial	C subset	VHDL	2010	Streaming	No	Yes	No
	Synphony C	Synopsys	Commercial	C/C++	VHDL/Verilog SystemC	2010	All	Yes	No	Yes
	Cyber-WorkBench	NEC	Commercial	BDL	VHDL Verilog	2011	All	Cycle/ Formal	Yes	Yes
	LegUp	U. Toronto	Academic	C	Verilog	2011	All	Yes	Yes	No
	Bambu	PoliMi	Academic	C	Verilog	2012	All	Yes	Yes	No
	DWARV	TU. Delft	Academic	C subset	VHDL	2012	All	Yes	Yes	Yes
	VivadoHLS	Xilinx	Commercial	C/C++ SystemC	VHDL/Verilog SystemC	2013	All	Yes	Yes	Yes
N/A	Trident	Los Alamos NL	Academic	C subset	VHDL	2007	Scientific	No	Yes	No
	CHiMPS	U. Washington	Academic	C	VHDL	2008	All	No	No	No
	Kiwi	U. Cambridge	Academic	C#	Verilog	2008	.NET	No	No	No
	gcc2verilog [45]	U. Korea	Academic	C	Verilog	2011	All	No	No	No
	HercuLeS	Ajax Compiler	Commercial	C/NAC	VHDL	2012	All	Yes	Yes	Yes
Abandoned	Napa-C	Sarnoff Corp.	Academic	C subset	VHDL/Verilog	1998	Loop	No	No	No
	DEFACTO	U. South Cailf.	Academic	C	RTL	1999	DSE	No	No	No
	Garp	U. Berkeley	Academic	C subset	bitstream	2000	Loop	No	No	No
	MATCH	U. Northwest	Academic	MATLAB	VHDL	2000	Image	No	No	No
	PipeRench	U.Carnegie M.	Academic	DIL	bitstream	2000	Stream	No	No	No
	SeaCucumber	U. Brigham Y.	Academic	Java	EDIF	2002	All	No	Yes	Yes
	SA-C	U. Colorado	Academic	SA-C	VHDL	2003	Image	No	No	No
	SPARK	U. Cal. Irvine	Academic	C	VHDL	2003	Control	No	No	No
	AccelDSP	Xilinx	Commercial	MATLAB	VHDL/Verilog	2006	DSP	Yes	Yes	Yes
	C2H	Altera	Commercial	C	VHDL/Verilog	2006	All	No	No	No
	CtoVerilog	U. Haifa	Academic	C	Verilog	2008	All	No	No	No

J. Cong et. al, “A survey and evaluation of FPGA high-level synthesis tools,” IEEE Trans. on CAD, 2015.

Productivity



RTL (Verilog) 4 gates / line

Behavioral C 28 gates / line

➔**7x more productive**

RTL Expert v.s HLS Expert

SPHERE DECODER IMPLEMENTATION RESULTS

Metric	RTL Expert	AutoPilot Expert	Difference (%)
Dev. time (man-weeks)	16.5	15	−9
LUTs	32 708	29 060	−11
Registers	44 885	31 000	−31
DSP48s	225	201	−11
18K BRAMs	128	99	−26

8×8 RVD-QRD IMPLEMENTATION RESULTS

Metric	RTL Expert	AutoPilot Expert	AutoPilot Expert
Dev. time (man-weeks)	4.5	3	5
LUTs	5082	6344	3862
Registers	5699	5692	4931
DSP48s	30	46	30
18K BRAMs	19	19	19

J. Cong et. al, “High-level synthesis for FPGAs: From prototyping to deployment,” IEEE Trans. on CAD, 2011.

Summary

- FPGA: Reconfigurable LSI or Programmable Hardware
- It consists of a programmable logic array

  and a programmable interconnection
- Standard FPGA design supports an RTL based one
- HLS brings value to the FPGA design:
  - Make hardware design easier for hardware engineers
  - Allow software engineers to design hardware and reap energy/performance benefits
- Benefits: Productivity, lower non-recurring engineering costs, maintainability, faster time to market

Exercise 1

1. (Mandatory) Show an application using a FPGA
2. (Selective) The modern FPGA has several configuration devices as follows:
   - Static RAM (SRAM)
   - Flash memory
   - Anti-fuse

   In that case, both the SRAM-based FPGA and Flash-based one supports “multiple-time” configurations, while an anti-fuse only “one-time” configuration. Actually, some vender adopts the anti-fuse-based FPGA than SRAM-based one. Why?

Deadline is 17th, Oct., 2016 (At the beginning of the next lecture)