Development of Complex Curricula for Molecular Bionics and Infobionics Programs within a consortial* framework**

Consortium leader

PETER PAZMANY CATHOLIC UNIVERSITY

Consortium members

SEMMELWEIS UNIVERSITY, DIALOG CAMPUS PUBLISHER

The Project has been realised with the support of the European Union and has been co-financed by the European Social Fund ***

**Molekuláris bionika és Infobionika Szakok tananyagának komplex fejlesztése konzorciumi keretben

***A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Digital- and Neural Based Signal Processing & Kiloprocessor Arrays

Digitális- neurális-, és kiloprocesszoros architektúrákon alapuló jelfeldolgozás

Virtual machines: signal processing with multicore systems

Virtuális gépek: jelfeldolgozás sokprocesszoros rendszereken

J. Levendovszky, A. Oláh, K. Tornai

Contents

- Implementation of neural networks
- Motivation of multicore systems
- Multicore systems
  - Definitions
- Survey of multicore systems, architectures
- Applications
  - Cellular automation demonstration
  - FFT implementation

NN Implementation examples

Applications	Examples
High energy physics	Digital-neurochip
Pattern recognition	FPGA, Digital
Image/object recognition	RAM Based, Optical
Image segmentation	FPGA, Digital
Generic image/video processing	RAM Based, Analog
Intelligent video analytics	Optical, FPGA
Finger print feature extraction, Direct feedback control	Analog
Autonomous robotics	Digital, FPGA, DSP
Sensorless control	FPGA
Optical character/handwriting recognition	Digital
Acoustic sound recognition	DSP
Real-time embedded control	Digital
Audio synthesis	Analog

NN Implementation – Digital neuron

- In a digital neuron, synaptic weights are stored in shift registers, latches, or memories.
- Adders, subtracters, and multipliers are available as standard circuits, and non-linear AFs can be constructed using look-up tables or using adders, multipliers, …
- Advantages: simplicity, high signal-to-noise ratio, easily achievable cascadability and flexibility, and cheap fabrication
- Drawbacks: slower operations, Conversion of the digital representations to and from an analog form may be required
  - Usually input patterns are available in analog form and control outputs also often required to be in analog form

NN Implementation – Analog neuron

- In an analog neuron weights are usually stored using one of the following: resistors, CCD-s, capacitors and FG EEPROM. In VLSI, a variable resistor as a weight can be implemented as a circuit involving two MOSFETs.
- The signals are typically represented by currents and/or voltages.
- The scalar product and subsequent non- linear mapping is performed by a summing amplifier with saturation
- Advantages: analog elements are generally smaller and simpler,
- Drawbacks: obtaining consistently precise analog circuits, especially to compensate for variations in temperature and control voltages, requires sophis- ticated design and fabrication
- The main challenges for analog designs are the synapse multiplier over a useful range and the storage of the synapse weights

NN Implementation – Neurochips

- FPGA Based implemenation
  - Reconfigurable FPGAs provide an effective programmable resource for implementing NNs allowing different design choices to be evaluated in a very short time
  - Partial and online reconfiguration capabilities in the latest generation of FPGAs offer additional advantages.
  - The circuit density using FPGAs is still comparably lower and is limiting factors in the implementation of large models with thousands of neurons
  - Associative neural memories
  - RAM based implementations

CNN Implementation

- CNN implementations can achieve speeds up to several teraflops and are ideal for the applications which require low power consumption, high processing speed, and emergent computation, e.g., real-time image processing.
- ACE16k
  - Mixed-signal SIMD-CNN ACE (Analogic Cellular Engine) chips as a vision system on chip realizing CNN Universal Machine (CNN-UM)
  - Designed dusing 35um CMOStechnology with 85% analog elements.
  - Consists of an array of 128x128 locally connected mixed signal processing units operating under SIMD mode
  - ACE16k chips have been used in commercial Bi-i speed vision system developed by AnaLogic Computers Ltd and MTA-SZTAKI
- An FPGA based emulated-digital CNN-UM implementation using GAPU (Global Analogic Programming Unit)
- Falcon was earlier proposed as a reconfigurable multi-layer FPGA based CNN-UM implementation employing systolic array architecture

NN Implementation

- Neuromorphic refers to a circuit that emulates the biological neural design
  - The processing is mostly analog, although outputs can be digital
- Optical neural networks
  - Designed on the principles of optical computing
  - Optical technology utilizes the effect of light beam processing that is inherently massively parallel, very fast, and without the side effects of mutual interference
  - Optical transmission signals can be multiplexed in time, space, and wavelength domains, and optical technologies may overcome the problems inherent in electronics
  - The results range from the development of special-purpose associative memory systems through various optical devices (e.g., holographic elements for implementing weighted interconnections) to optical neurochips.
  - Optical techniques ideally match with the needs for the realization of a dense network of weighted interconnections.
  - Optical technology has a number of advantages for making interconnections, specifically with regard to density, capacity and 2D programmability

NN Implementation table

ANN	Digital	Analog	Hybrid	Neuromorphic	FPGA	Optical
MLP (Perceptron)	x				x	
RBF	x	x			x	
SOFM	x	x				
FFNN	x				x	X
Spiking NN	x			x	x	
Pulse coded NN		x		x	x	
CNN	x	x	x		x	x
AM		x	x		x	x
Recurrent NN		x				x
Stochastic NN					x	

Introduction – Motivation

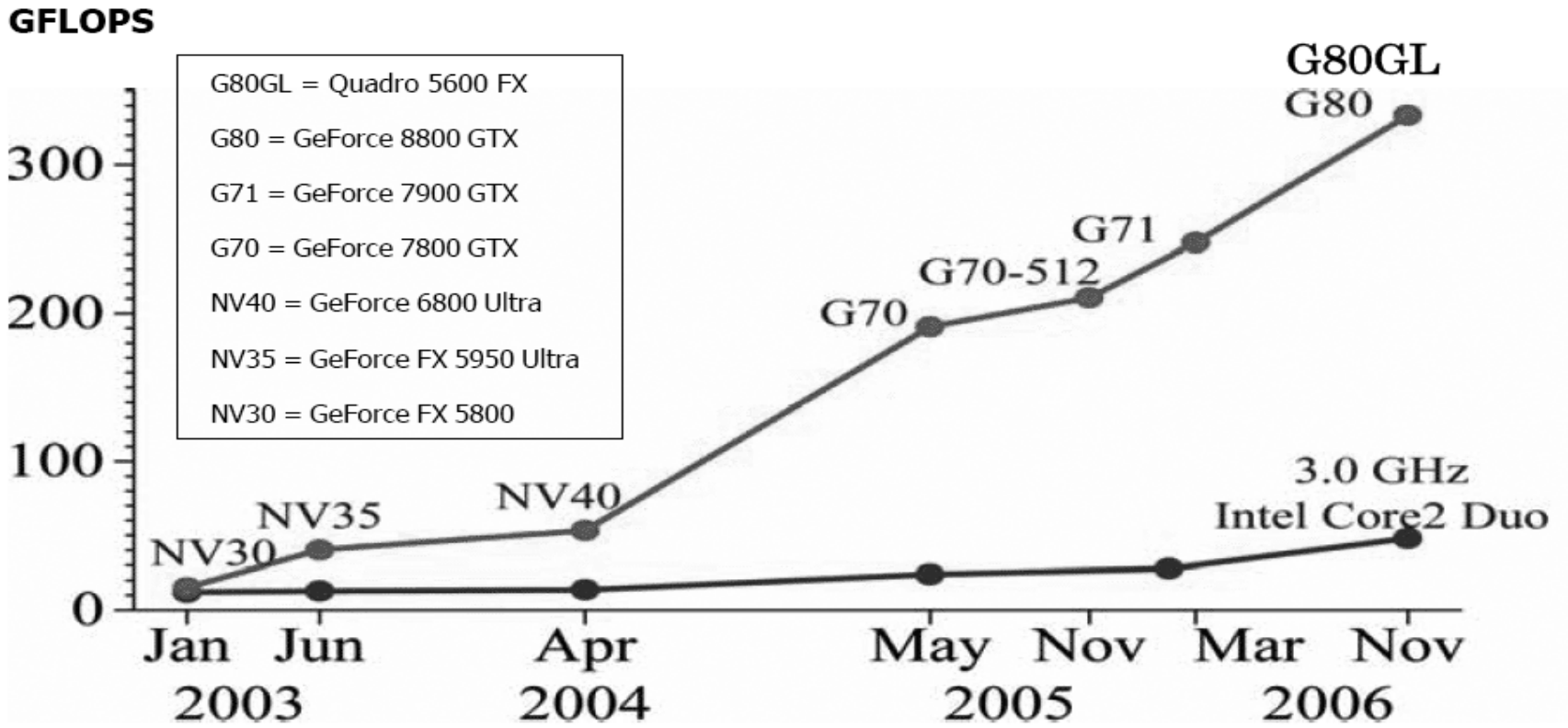
- Classical architecture: one processing unit
- The predicted improvements by Moore-law has physical constraints
  - Size
  - Frequency
  - Power consumption / heat dissipation
- Consequently the number of processing unit on one chip must be increased
  - Changing the art of architecture
  - Changing the art of programming

Trend

- Improving the computational capacity
  - Instead of using higher frequencies the number of cores on a single chips are increased
  - Decreasing the power consumption
- More and more chip with many cores are available in the market
  - New programs have to be adapted to the architecture

Trend of development

- Computational capacity of GPU and CPU



Classifications of architectures

- From the perspective of applications
  - General purpose
    - Coding, decoding, software radio
  - Specific purpose
    - Well defined application
    - ASIC: Application Specific Integrated Circuits
    - RISC: Reduced Instruction set

Classifications of architectures

- From the perspective of applications
  - Data Processing Dominated
    - Processing large data flows
      - Image or Video
      - Voice or Music
      - Processing radio signals
    - Repeating same instruction on multiple data
      - It can be parallelized well

Classifications of architectures

- From the perspective of applications
  - Control Processing Dominated
    - Packing or unpacking filed
    - Processing network signals
    - Conditional instructions, huge state space, enormous number of re-used data
      - Hardly parallelizable
    - Less GP core

Classifications of architectures

- Computational power versus power consumption
  - In most cases the parameters are restricted
    - Mobile phone with capability of playing videos
  - The goal is to increase the computational power
    - Furthermore the power consumption is also an issue
      - Previous example: mobile phone
      - This issue must be considered as designing

ISA

- ISA: Instruction set architecture
  - Defines the microarchitecture
  - Defines the hardware-software interface
- Each core of traditional ISA is a modified classical processor core
  - Containing atomic instructions for synchronization
  - Supported by compiler and software
  - Not necessarily efficient with high power consumption

ISA

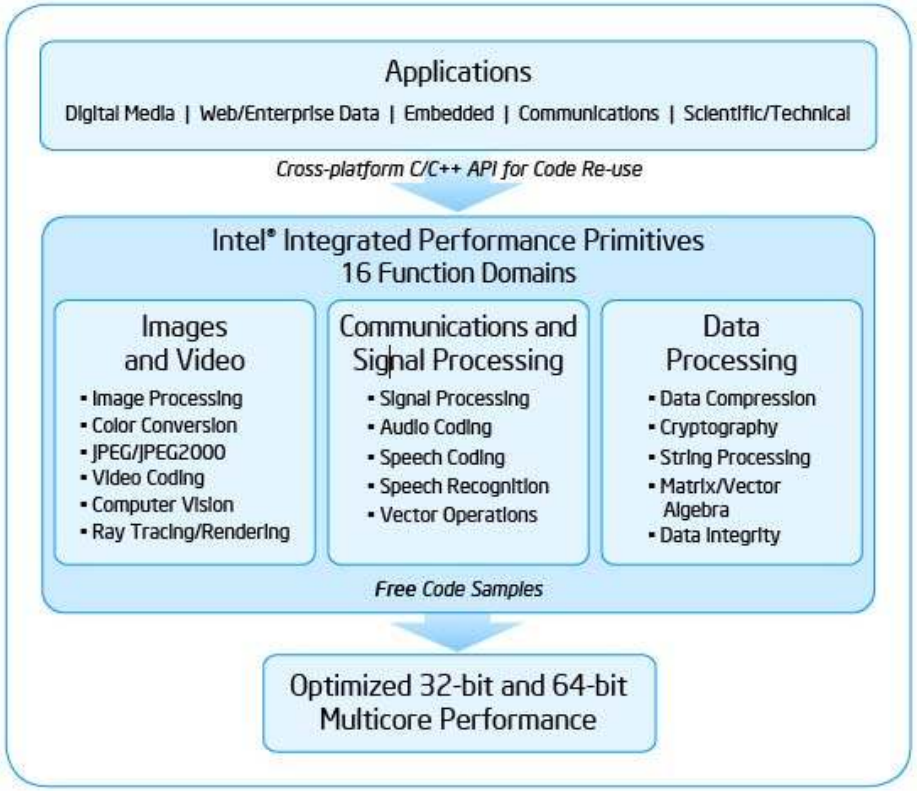
- ISA
  - RISC – Reduced Instruction Set Computer
    - Simple microarchitecture and compiler
    - The final code is more complicated
  - CISC – Complex Instruction Set Computer
    - More, complex instructions
    - Complex compiler and microarchitecture
    - More optimized final code

ISA

- ISA
  - Instruction set extensions
    - MMX: MultiMedia eXtension (64 bit FP-Instructions)
    - Streaming SIMD Extensions (SSE)
      - 70 new special instructions (FP-I, AL-I)
    - SSE2, SSE3, SSE4, SSE5 extensions
    - 3DNow! (FP-I, DSP instructions)
    - Advanced Vector Extensions (AVX) (SIMD)
    - X86-64, AMD64, EM64
    - More optimized, harder to compile

Intel IPP

- Add-on
  - Different instruction sets
  - To use it efficiently a special library should be used
  - Intel c++ lib: IPP



Microarchitecture

- In-order processing element
  - The execution order of instructions is equivalent to the given order of the program code
  - With pipelines (increasing the size and numbers) the computational power can be further extended
    - Superscalar
    - Needs logical control
    - The length of the instruction is critical
  - Small size, complexity and power consumption
    - A number of them can be placed on single chip
    - The performance is lower

Microarchitecture

- Out-of-order processing element
  - In order to fully utilize the pipelines the order of the instructions are changed
  - This can be extremely fast due to scheduling
  - The complexity is huge and the space on the chip is large
- SIMD (Single Instruction Multiple Data)
  - Solving data oriented problems
    - Very efficient if the data can be formed as vectors
    - Unable to use when the application is control dominated

Microarchitecture

- Very Long Instruction Word (VLIW)
  - Processing multiple data in same time
  - Uses pipelines
  - Despite of superscalar architecture it uses less hardware logic for scheduling
    - The compiler optimizes the scheduling
  - Reduced size, less complexity
  - Needs special compiler
  - It is possible to obtain worse code in special cases

Memory

- Consistency model
  - Defines how the memory instructions may be rearranged
  - Defines the style of the programming method
  - Has influences on the computational power
- Strong consistency model
  - The order of the instructions can not be changed
  - Easy to code and simple behavior model
- Weak consistency model
  - Simpler memory controllers
  - The order of memory accessing may vary on different runs

Cache

- On multicore systems the importance of cache memory is higher
  - Decreases the load of the slow central memory
  - Decreases the access time of data
- Automatically managed cache memory
  - The processes do not know each other. Virtually each process have the resources
  - Therefore the performance varies due to the overhead
- Software managed cache
  - Application dependent whether it is practical to use

Size of cache

- Application dependent
  - Larger cache results lager computational performance
    - Only when data is reused
  - There are chips using two kind of cache mode
    - Streaming mode
    - Normal mode
- The size is determined by manufacturing cost and chip size
- Cache levels
  - Speeding up the memory access of the farther memory (farther in sense of time)

Intrachip connections

- Bus
  - Easy to implement
  - Each core can access the common resources with the same latency
  - Very slow above a certain number of cores
- Ring
  - Needs traffic control logic
  - Different latencies
  - Supports more cores/processors

Intrachip connections

- Network on Chip (NoC)
  - Similar to Ring, but latencies are smaller
  - More complex logical circuits
- Crossbar
  - Equal latencies for each cores
  - High number of cores can be connected
  - Complex logic
  - Large size on the chip surface

Intrachip connections

- Tasks
  - Communication
  - Maintaining cache coherency
    - It is common not to deal with the coherency
    - Broadcast based
      - Broadcasting the changes
    - Directory based
      - The memory is divided into blocks
      - Each block has responsible manages (home directory)
      - The synchronization is made from and into the home directory

Further elements

- Special integrated devices and special purpose circuits
  - Memory controller
  - Coder and decoder supporter
  - Speeding up the image processing
    - Hardware implemented special instructions
  - Rastering logics

Comparison of different architectures

[TABLE 4] TABLE OF HIGH-PERFORMANCE MULTICORES.

	ISA	MICROARCHITECTURE	NUMBER OF CORES	CACHE	COHERENCE	INTERCONNECT	CONSISTENCY MODEL	MAX. POWER	FREQUENCY	OPS/CLOCK
AMD RADEON R700 [20]	N/A	FIVE-WAY VLIW	160 CORES, 16 CORES PER SIMD BLOCK, TEN BLOCKS	16 KB LCL STORE/SIMD BLOCK	NONE	N/A	NONE	150 W	750 MHZ	800-1,600 OPS/CLOCK
NVIDIA G200 [8], [21]	N/A	ONE-WAY IN-ORDER	240, EIGHT CORES PER SIMD UNIT, 30 SIMD UNITS	16 KB LCL STORE/EIGHT CORES	NONE	N/A	NONE	183 W	1.2 GHZ	240-720 OPS/ CLOCK
INTEL LARRABEE* [22]	X86	TWO-WAY IN-ORDER, 4-WAY SMT, 512-B SIMD	UP TO 48*	32 KB IL1 AND 32 KB DL1/ CORE, 4 MB L2	BROADCAST	BIDIRECTIONAL RING	PROCESSOR	N/A	N/A	96-1,536 OPS/ CLOCK
IBM CELL [9], [23]	POWER	TWO-WAY IN-ORDER, TWO-WAY SMT PPU, 2-WAY IN-ORDER 128-B SIMD SPU	1 PPU, EIGHT SPUs	PPU: 32 KB IL1 AND 32 KB DL1, 512 KB L2; SPU: 256 KB LCL STORE	NONE	BIDIRECTIONAL RING	WEAK (PPU), NONE (SPU)	100 W	3.2 GHZ	72 OPS/CLOCK
MICROSOFT XENON [10]	POWER	TWO-WAY IN-ORDER, TWO-WAY SMT, 128-B SIMD	THREE	32 KB IL1 AND 32 KB DL1/ CORE, 1 MB L2	BROADCAST	CROSSBAR	WEAKLY ORDERED	60 W	3.2 GHZ	6-24 OPS/ CLOCK

*All values are estimates as processor is not yet in production.

Comparison of different architectures

[TABLE 5] TABLE OF DSP AND EXOTIC MULTICORES.

	ISA	MICROARCHITECTURE	NUMBER OF CORES	CACHE	COHERENCE	INTERCONNECT	CONSISTENCY MODEL	MAX. POWER	FREQUENCY	OPS/CLOCK
AMBRIC AM2045 [24], [25]	N/A	ONE-WAY IN-ORDER SR, THREE-WAY IN-ORDER SRD	168 SR, 168 SRD	21 KB LCL STORE/EIGHT CORES	NONE	NoC	NONE	6–16 W	350 MHZ	672 OPS/CLOCK
ELEMENT CXI ECA-64 [26], [3]	N/A	ONE-WAY IN-ORDER, DATAFLOW CONNECTIONS TO 15 RECONFIGURABLE ALUs	FOUR CLUSTERS OF ONE CORE+ALUs	32 KB OF LCL STORE/ CLUSTER	NONE	HIERARCHIAL NoC	NONE	1 W	200 MHZ	64 OPS/CLOCK (16-8)
TI TMS320-DM6467 [14]	ARM, C64X	ONE ARM9 ONE-WAY IN-ORDER, ONE C64X EIGHT-WAY VLIW	TWO	ARM9: 16 KB IL1, 8 KB DL1; C64X: 32 KB IL1 AND DL1, 128 KB L2	NONE	BUS	WEAKLY ORDERED	3–5 W	ARM: 297–364 MHZ, C64X: 594–729 MHZ	1–9 OPS/CLOCK
TI OMAP 4430 [12]	ARM, C64X	TWO ARM THREE-WAY OUT-OF-ORDER, ONE C64X EIGHT-WAY VLIW	THREE	N/A	BROADCAST AMONG ARM CORES	BUS	WEAKLY ORDERED	1 W	1GHZ	6–140 PS/CLOCK
TILERA TILE64 [27], [28]	N/A	THREE-WAY VLIW	36–64	8 KB IL1 AND DL1/CORE, 64 KB L2/CORE	DIRECTORY	NoC	N/A	15–22 W	500–866 MHZ	108–192 OPS/CLOCK
HIVEFLEX CSP2X00* [29]	N/A	TWO-WAY VLIW CONTROL CORE, FIVE-WAY VLIW COMPLEX CORE	TWO TO FIVE	2X CONFIGURABLE L1 FOR BASE CORE, LCL STORE FOR COMPLEX CORE	NONE	BUS	NONE	0.25 W	200 MHZ	2–22 OPS/CLOCK

*Numbers are estimates because design is offered only as a customizable soft core.

Comparison of different architectures

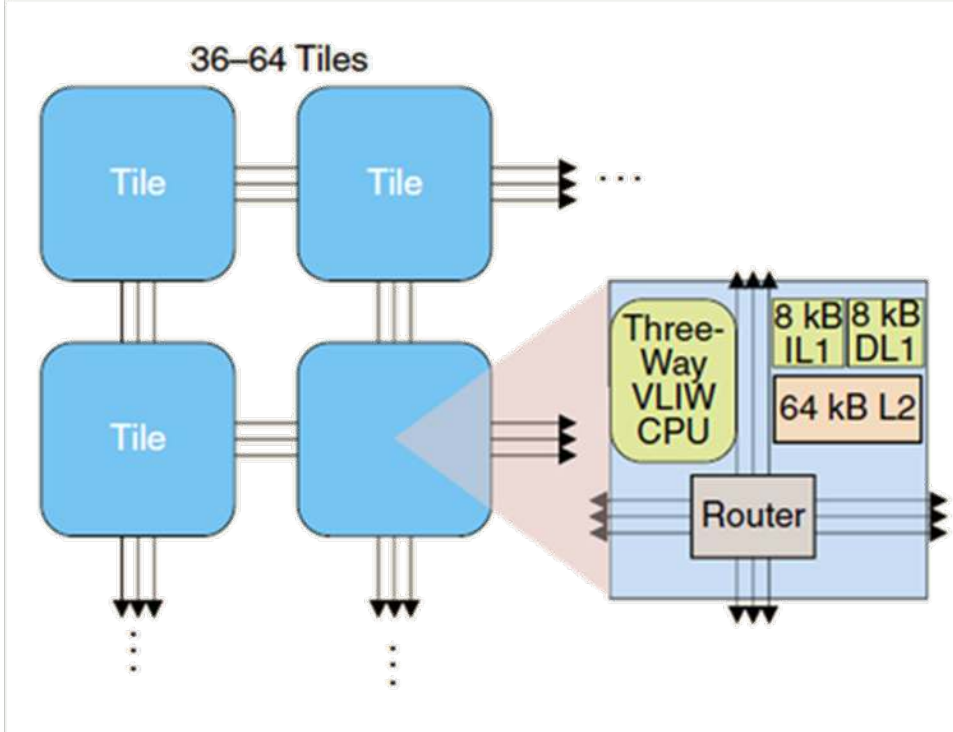
[TABLE 3] TABLE OF GENERAL-PURPOSE SERVER AND MOBILE/EMBEDDED MULTICORES.

	ISA	MICROARCHITECTURE	NUMBER OF CORES	CACHE	COHERENCE	INTERCONNECT	CONSISTENCY MODEL	MAX. POWER	FREQUENCY	OPS/CLOCK
AMD PHENOM [11], [15]	X86	THREE-WAY OUT-OF-ORDER SUPERSCALAR, 128-B SIMD	FOUR	64 KB IL1 AND DL1/ CORE, 256 KB L2/CORE, 2–6 MB L3	DIRECTORY	POINT TO POINT	PROCESSOR	140 W	2.5 GHZ–3.0 GHZ	12–48 OPS/ CLOCK
INTEL CORE I7 [2], [5]	X86	FOUR-WAY OUT-OF-ORDER, TWO-WAY SMT, 128-B SIMD	TWO TO EIGHT	32 KB IL1 AND DL1/ CORE, 256 KB L2/CORE, 8 MB L3	BROADCAST	POINT TO POINT	PROCESSOR	130 W	2.66 GHZ–3.33 GHZ	8–128 OPS/ CLOCK
SUN NIAGARA T2 [16], [17]	SPARC	TWO-WAY IN-ORDER, EIGHT-WAY SMT	EIGHT	16 KB IL1 AND 8 KB DL1/ CORE, 4 MB L2	DIRECTORY	CROSSBAR	TOTAL STORE ORDERING	60–123 W	900 MHZ–1.4 GHZ	16 OPS/CLOCK
INTEL ATOM [18], [5]	X86	TWO-WAY IN-ORDER, TWO-WAY SMT, 128-B SIMD	ONE TO TWO	32 KB IL1 AND DL1/ CORE, 512 KB L2/CORE	BROADCAST	BUS	PROCESSOR	2–8 W	800 MHZ–1.6 GHZ	2–16 OPS/ CLOCK
ARM CORTEX-A9[†] [6]	ARM	THREE-WAY OUT-OF-ORDER	ONE TO FOUR	(16,32,64) KB IL1 AND DL1/CORE, UP TO 2 MB L2	BROADCAST	BUS	WEAKLY ORDERED	1 W (NO CACHE)	N/A	3–12 OPS/ CLOCK
XMOS XS1-G4 [19]	XCORE	ONE-WAY IN-ORDER, EIGHT-WAY SMT	FOUR	64 KB LCL STORE/CORE	NONE	CROSSBAR	NONE	0.2 W	400 MHZ	4 OPS/CLOCK

[†]Numbers are estimates because design is offered only as a customizable soft core.

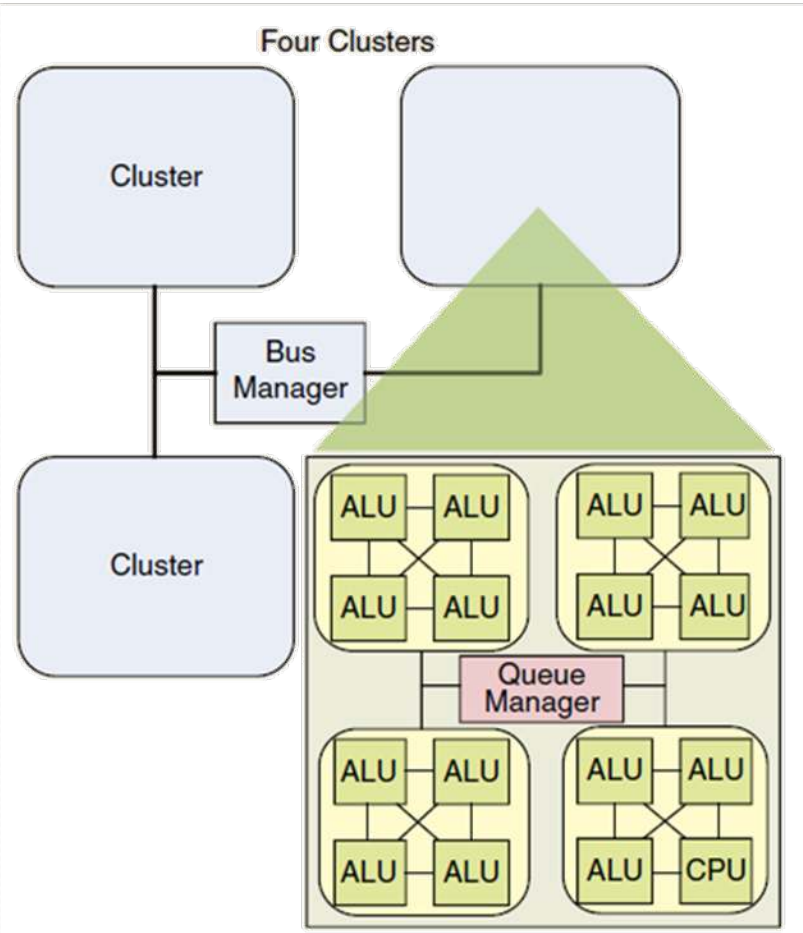
Tilera TILE64 – DSP

- Maximum 64 VLIW
- NoC
- Shared memory
  - GP
- Big power consumption
- Directory based coherency



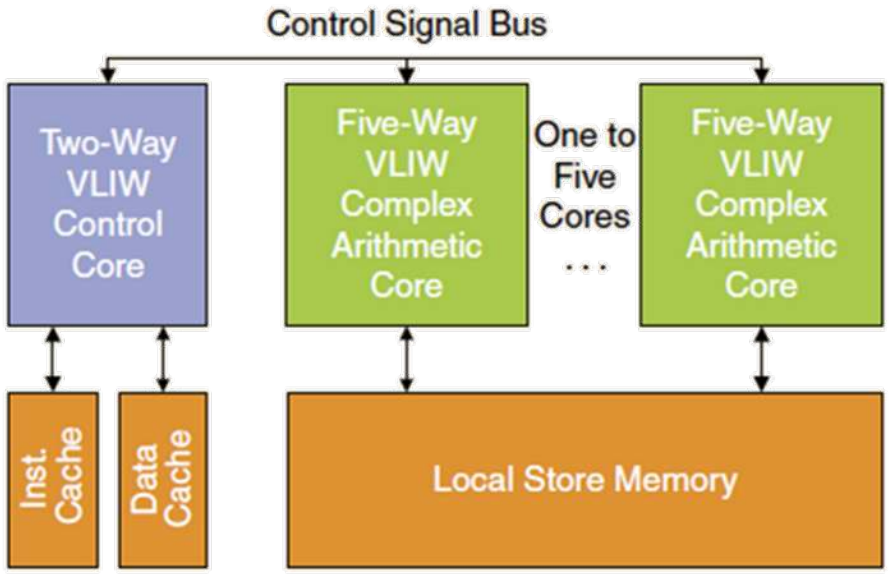
Element CXI ECA-64-DSP

- Data driven applications
- Software controlled memory
- Small power consumption
- Cluster
  - 15 ALU, 1 CPU
  - 32 kB local memory
  - Hierarchical connections



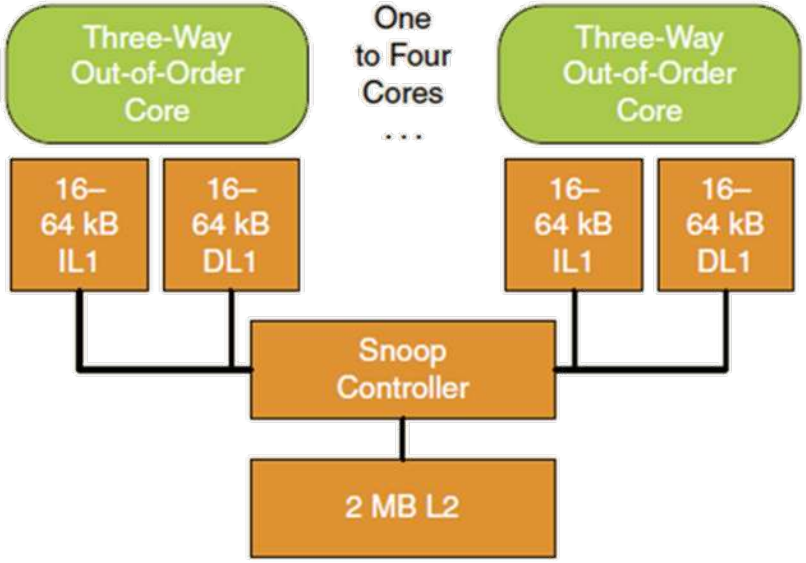
Silicon Hive Hiveflex DSP

- Small power consumption
- Heterogeneous construction
- Simple memory architecture
- It is hard to write efficient software



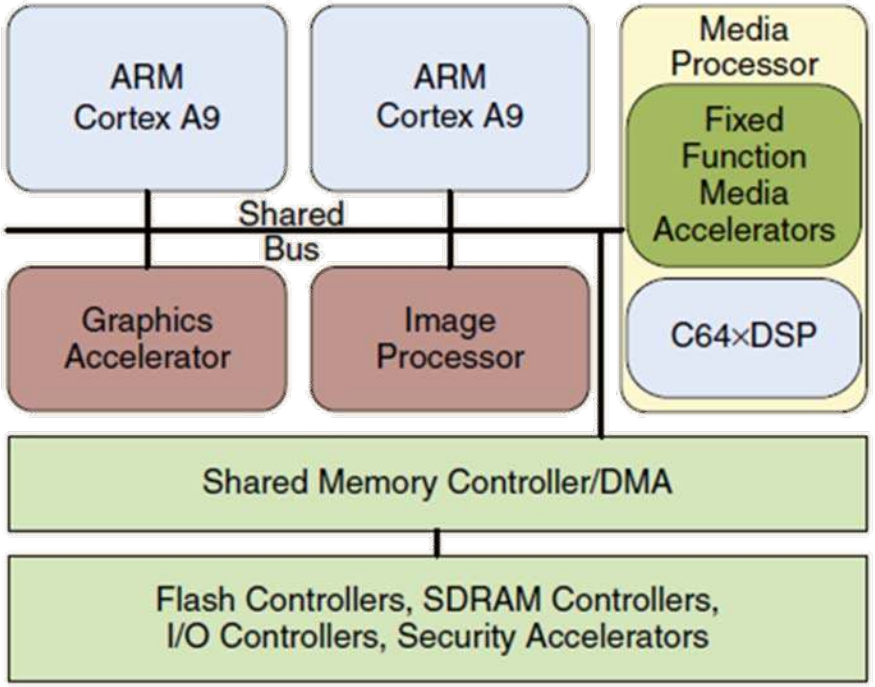
ARM Cortex-A9 (GP, Mobile)

- Capable of running Operating Systems and traditional applications
- Broadcast coherency
- Poor performance on data driven applications
- Variably core number



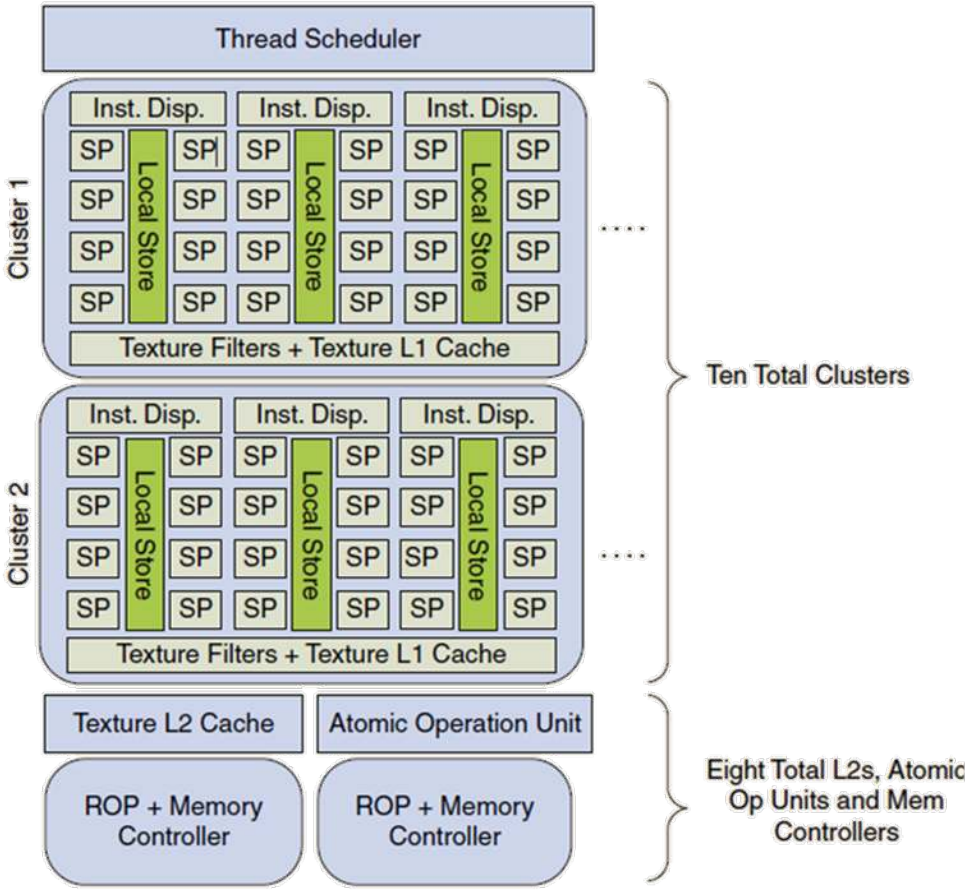
TI OMAP 4330-GP SoC

- Smartphones
- ARM for general purpose applications
- C64x for data driven multimedia applications
- Shared common memory



Nvidia G200 - GPU

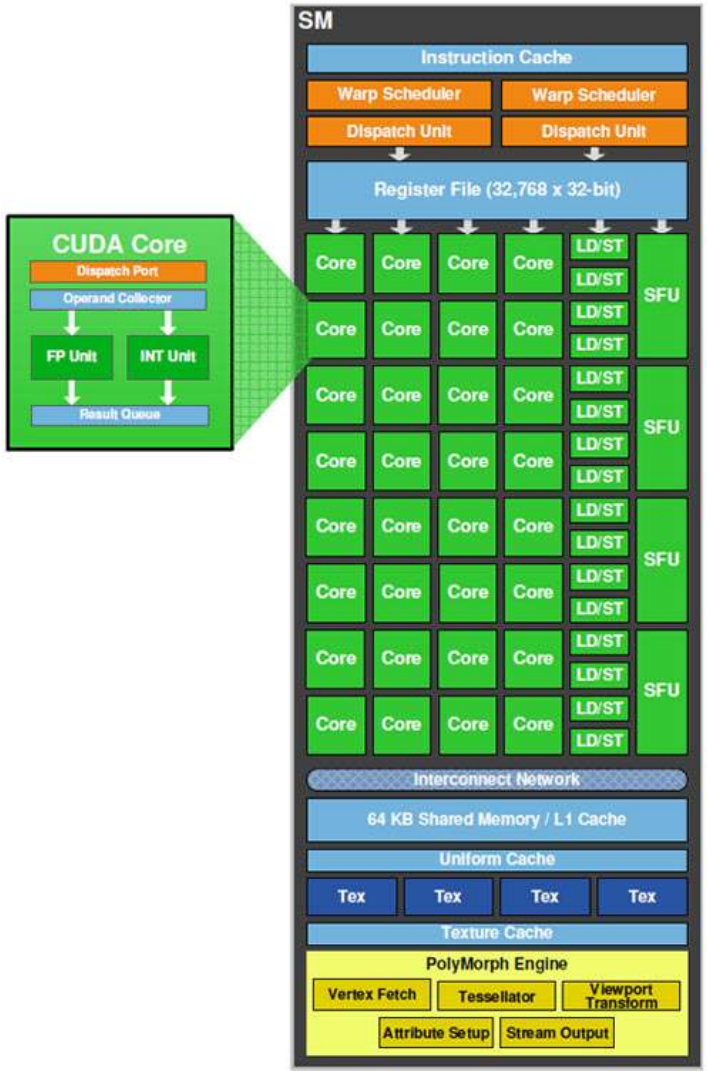
- Summary: 240 SP core
- 30 processors
  - 8 SIMD in each group
  - Capable of branching, but all of 24 core must do the same
- Relatively small amount of memory per core
- MIMD design but for GPU purposes



Nvidia Fermi

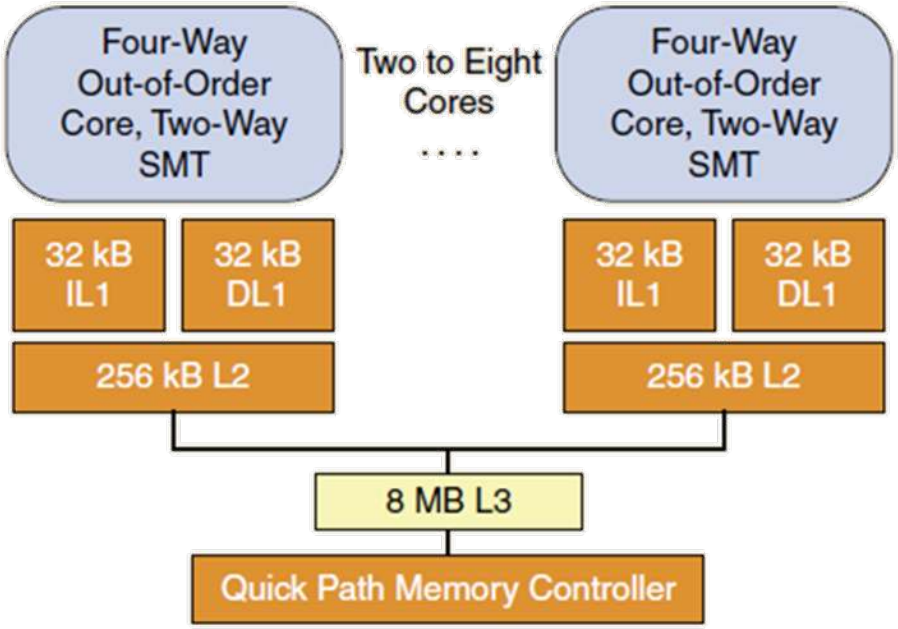
- 32 core in one SM processor
  - 4 times more than in the previous generation
- More local memory and cache size

- Each CUDA core (SP) has a double precision FPU and integer ALU
- It is important to properly design the algorithm for GPGPU and to fully utilize the available memory



Intel Core i7 – GP

- High power consumption
- 8 cores maximum
- Each core
  - can be multithread
  - 128 bit SIMD unit
- Memory coherence
- Huge cache
  - Broadcast based

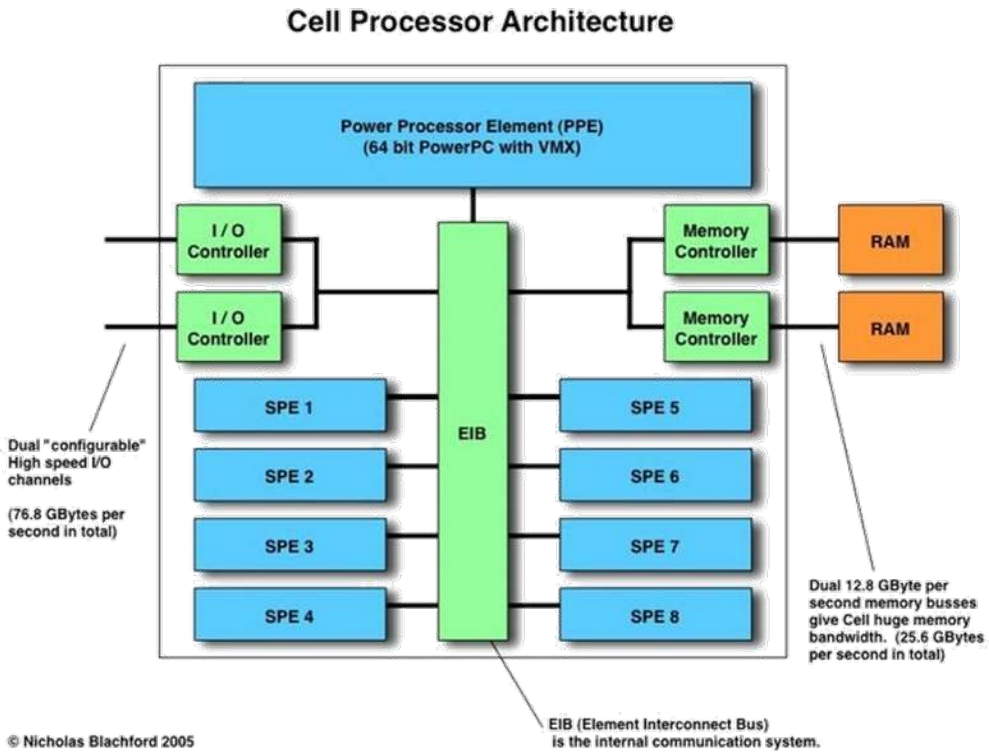


Cell broadband engine

Cell

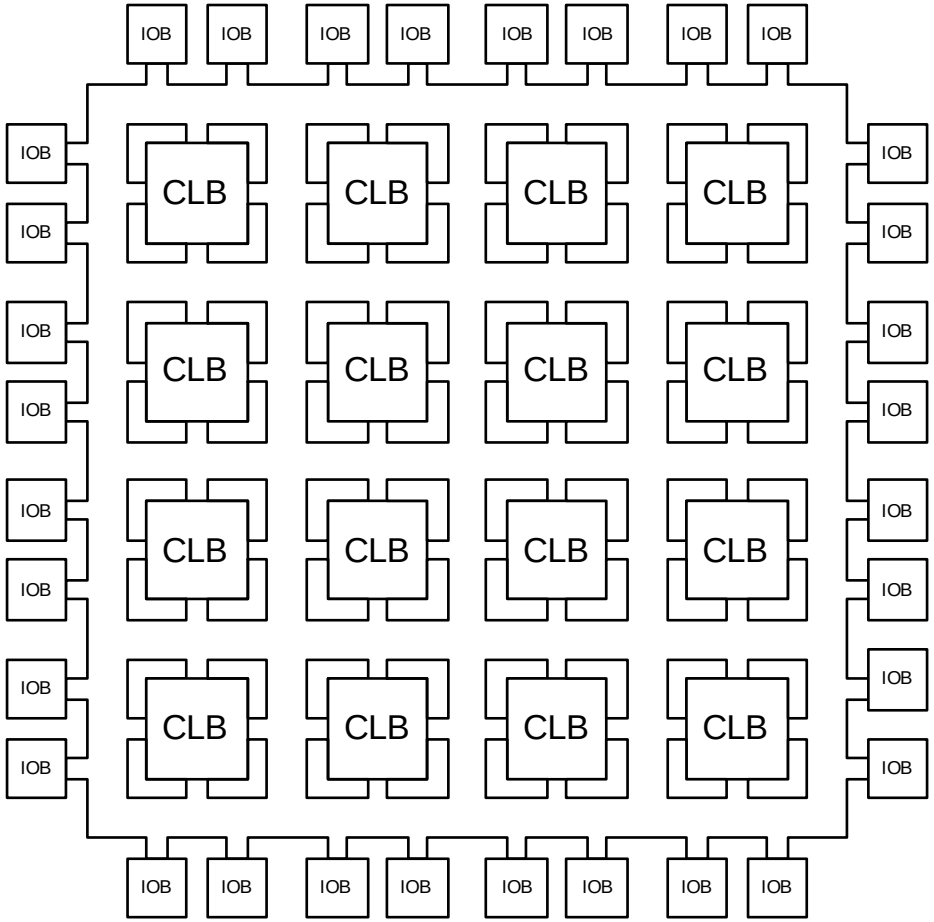
- 1 PPE
  - Two threaded RISC
    - L1 and L2 cache
  - Operating system and supervising SPE-s
  - One of SPE
    - 128 bits SIMD, RISC
    - 256 memory,
- Circular data bus

Relatively low power consumption

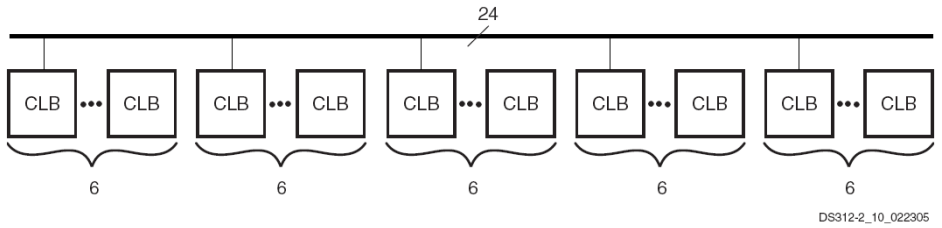
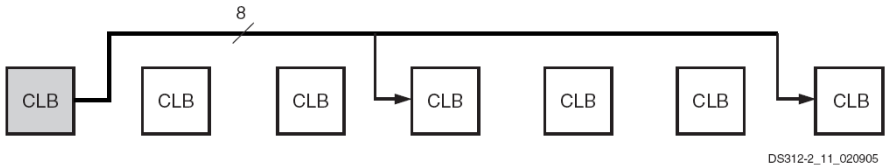
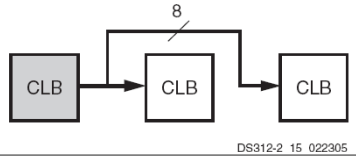
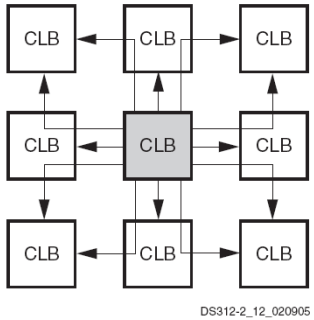


FPGA

- Configurable Logic Block (CLB)
  - Look-up table (LUT)
  - Register
  - Logic circuit
    - Adder
    - Multiplier
    - Memory
    - Microprocessor
- Input/Output Block (IOB)
- Programmable interconnect



Interconnection types

Horizontal and Vertical Long Lines (horizontal channel shown as an example)	
Horizontal and Vertical Hex Lines (horizontal channel shown as an example)	
Horizontal and Vertical Double Lines (horizontal channel shown as an example)	
Direct Connections	

Cellular automaton

- Cells in a regular grid
- Every cell may be in one of a finite set of states
- The grid may be arbitrary dimensional
- The time is discrete
- Every cell operates with the same rules
- Generation
  - After each cell performed the state transition governed by the rule a new generation of cells (state)

Cellular automaton – Rules

- One dimensional automaton:
  - Each cell has two neighbors
  - The output of the cell is based on the current output of the neighbors and the cells current output:
  - For example current pattern: 000, output 1
  - These local rules can be written as tables:

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Cellular automaton – Rules

- Rule table

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

- The first row is constant
- The second rule can be interpreted as a binary representation of a number therefore this sequence: 01101110 can be interpreted as 110.
- Thus the name of the rule is Rule 110.

Cellular automaton – Rules

- Rule 124

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	1	1	1	0	0

- Rule 30

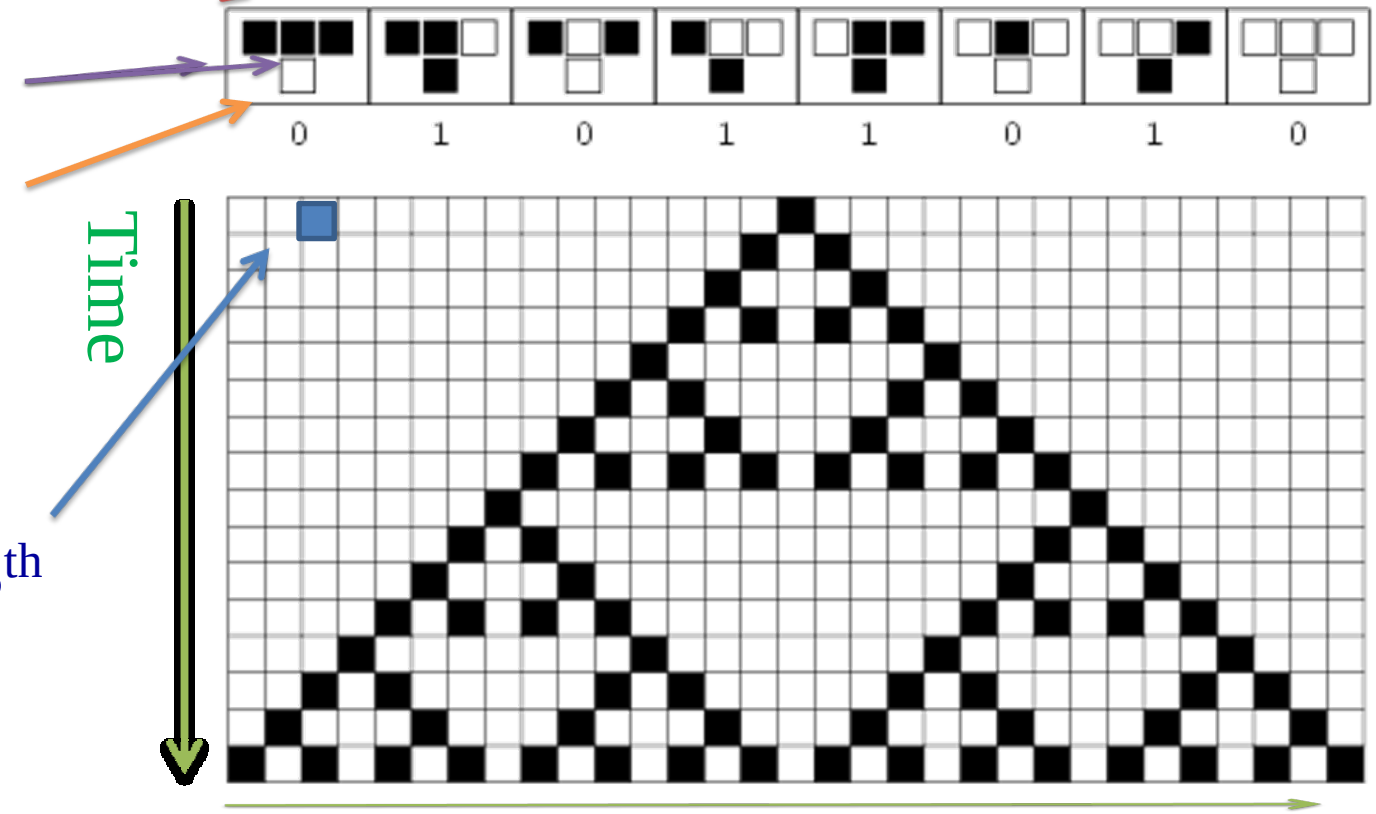
current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0

Cellular automaton

Previous state of cell

Previous state of neighbor cells

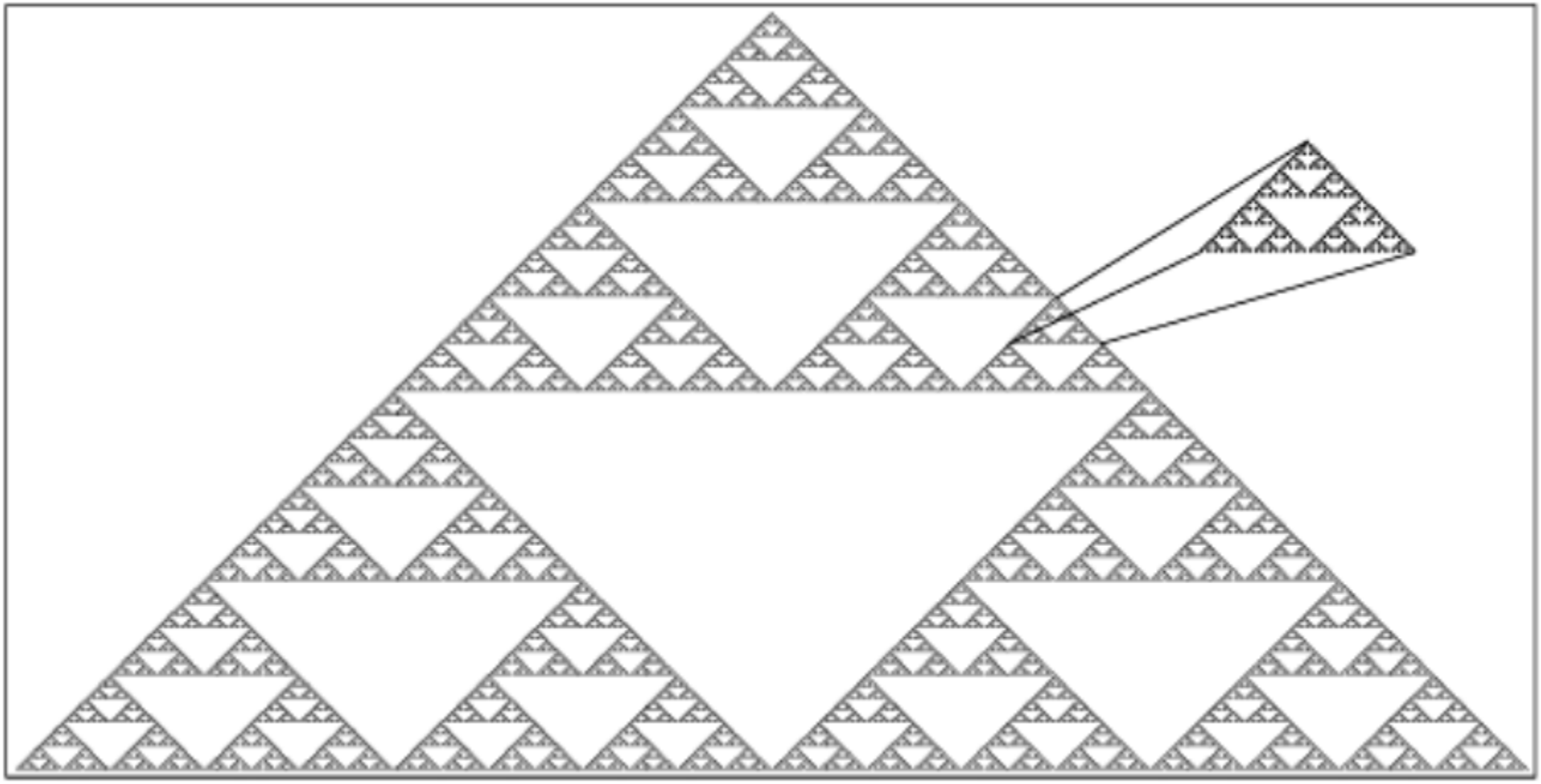
New state of cell



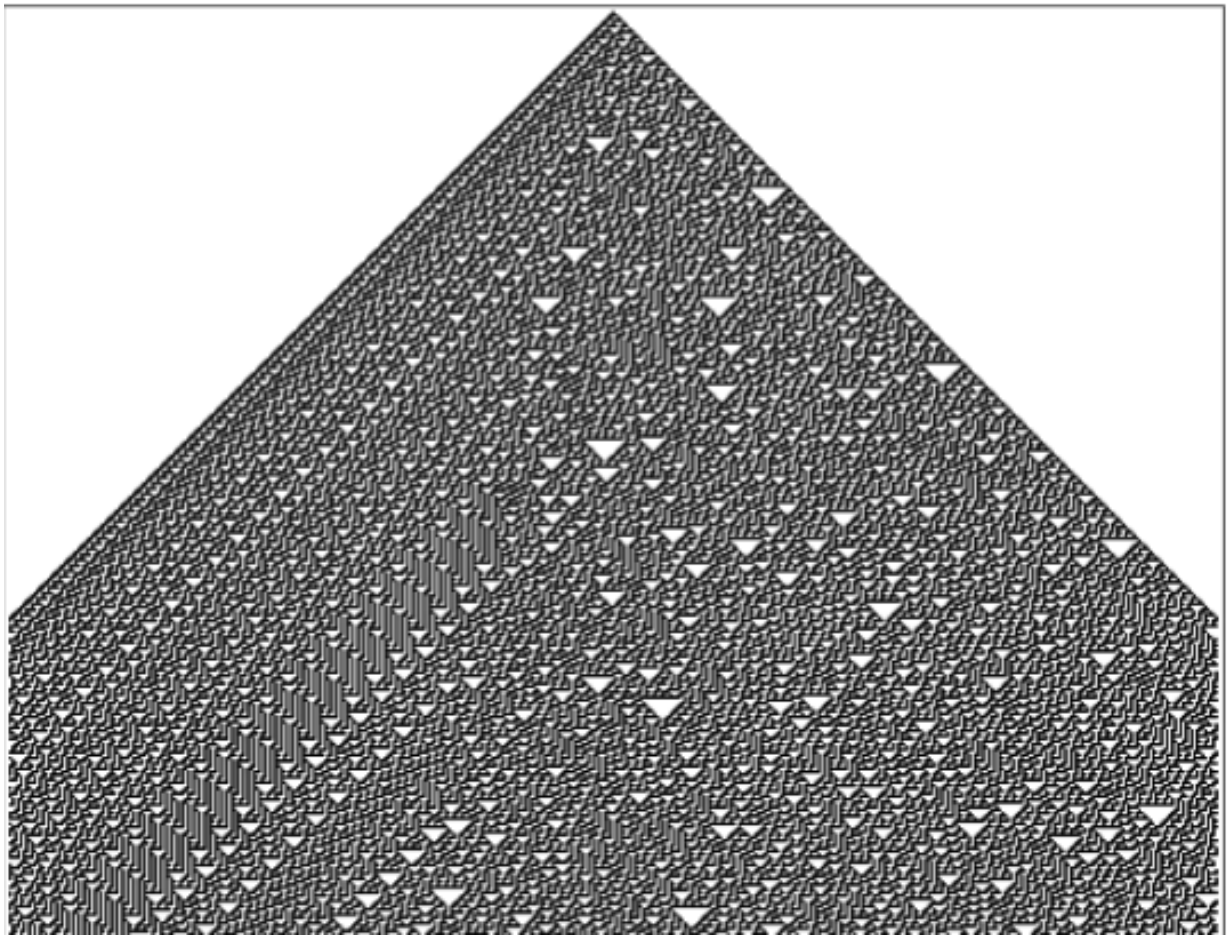
Output of 3th cell at time instant 1

Space – 1 row = 1D array of cells

Cellular automaton – Rule 90



Cellular automaton – Rule 30



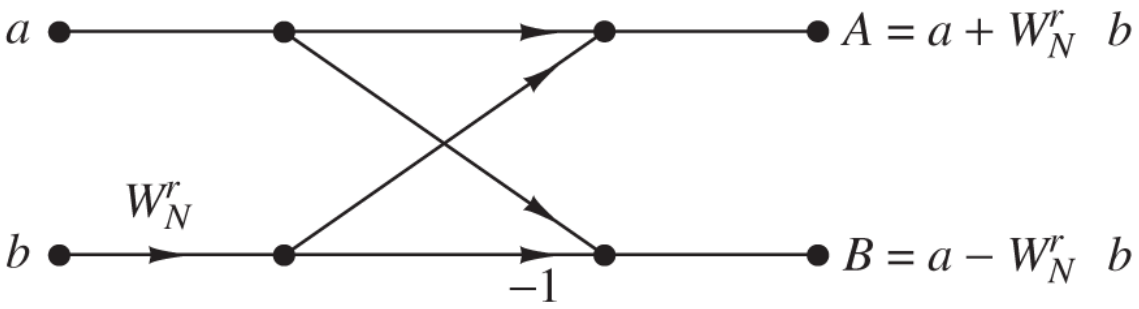
Cellular automaton – Parallelization

- This automaton can be easily parallelized on FPGA
- The speedup obtained is extremely huge
  - Simulation
    - The cells are evaluated each after each
  - Hardware implementation
    - 640 cells are evaluated simultaneously on a chip in a 2D array when the size of array is 640x480

As previous example indicates the parallelized version outperforms the simulated linear implementation

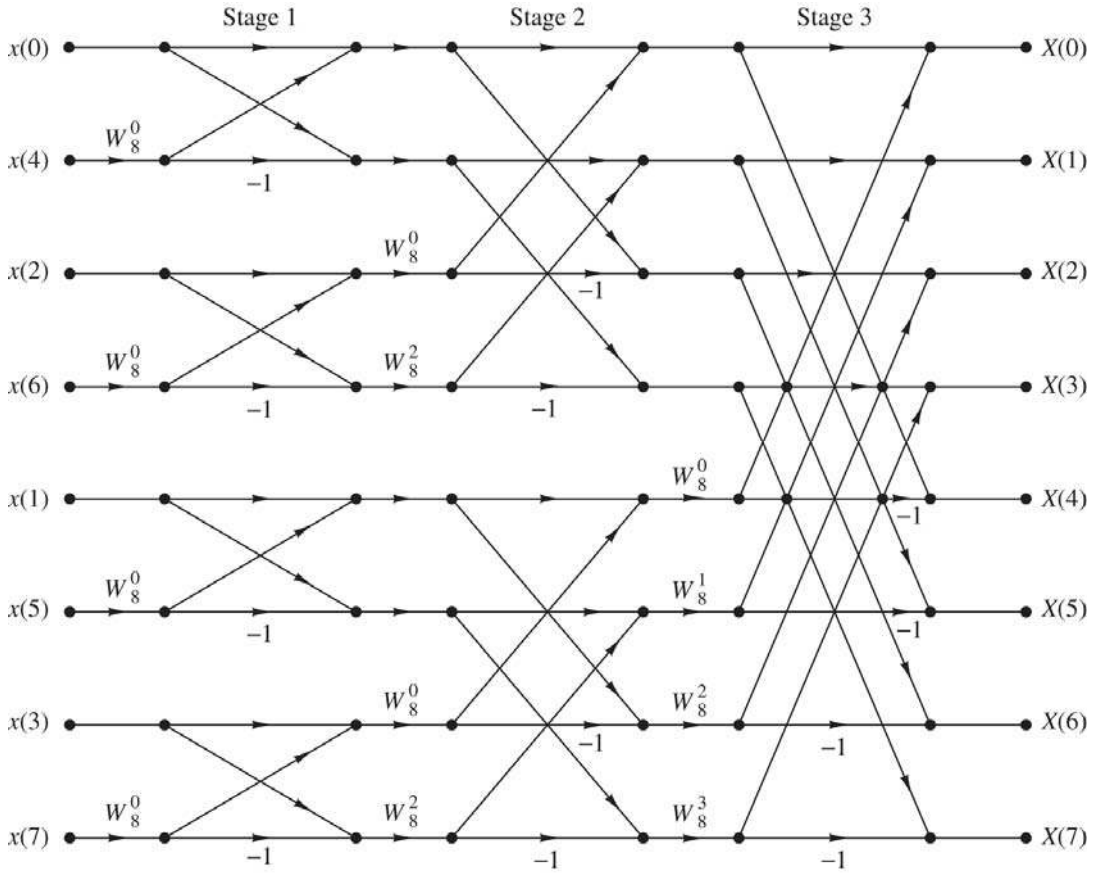
FFT (Fast Fourier Transform)

- Recall
  - FFT is used to compute a signals DFT (Discrete Fourier Transform) in short time
  - The FFT algorithm eliminates a great number of calculation of the DFT algorithm
  - „Butterfly”



FFT (Fast Fourier Transform) – Parallelization

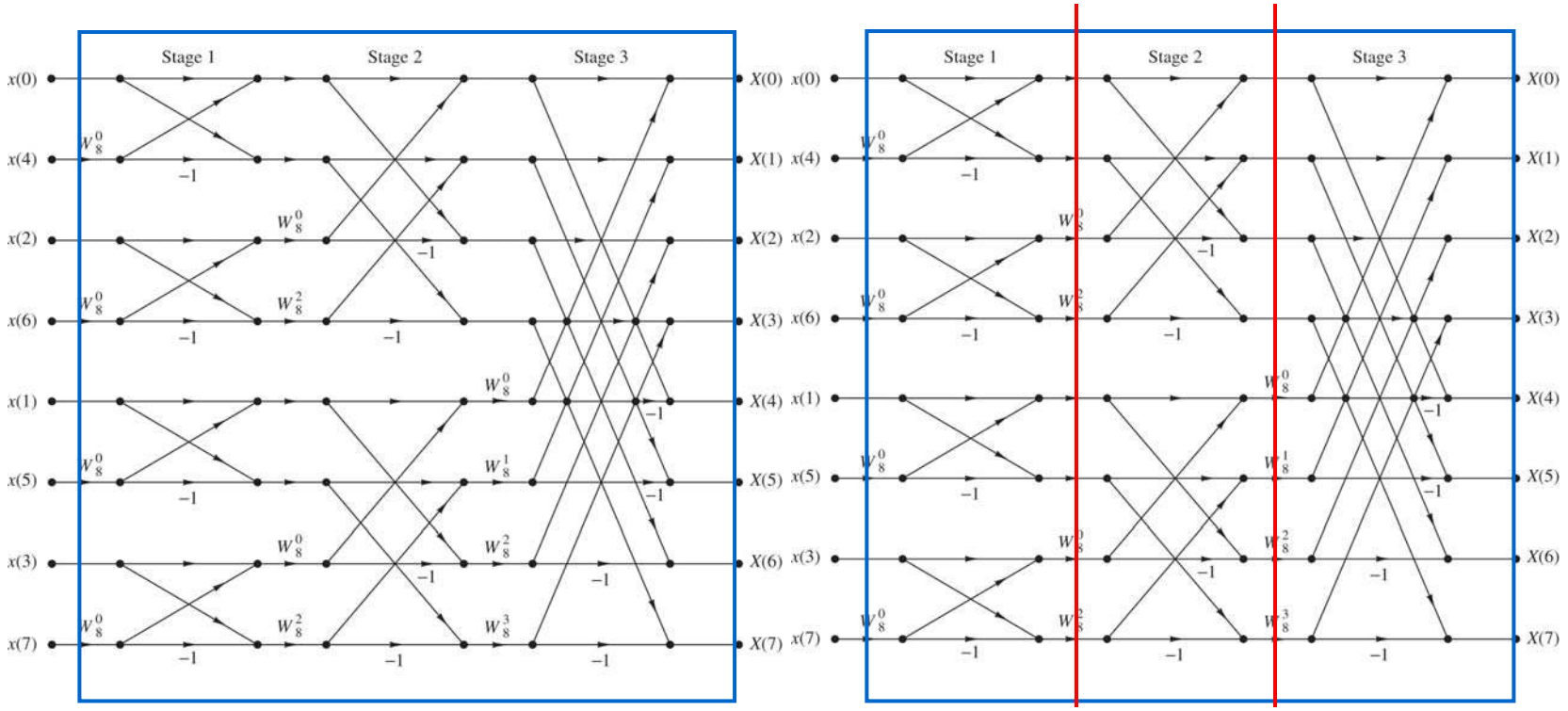
- Recall
  - More butterfly
  - Complex, bigger FFT
  - This can be easily parallelized



FFT – Parallelization, pipeline

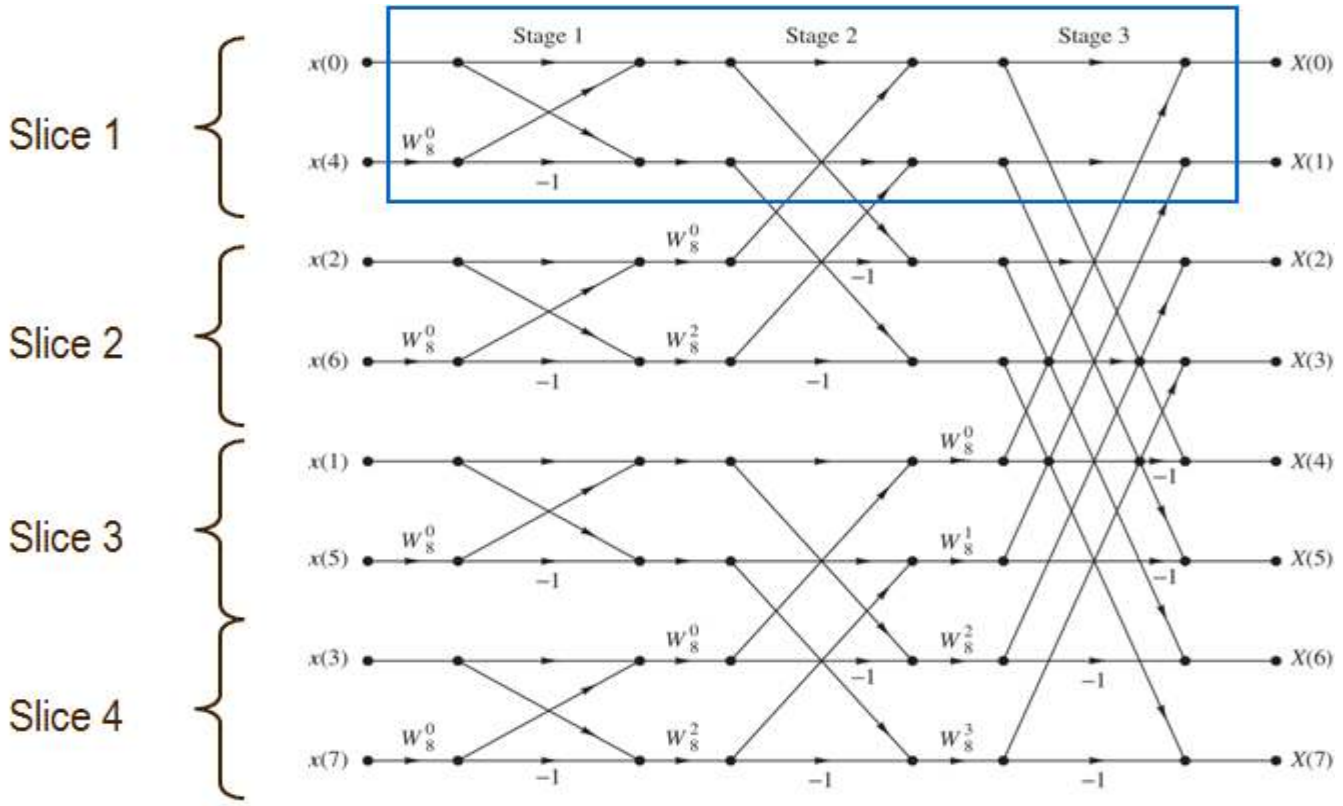
- Original

Pipeline stages



FFT – Parallelization, pipeline

- Parallel – Pipeline FFTs

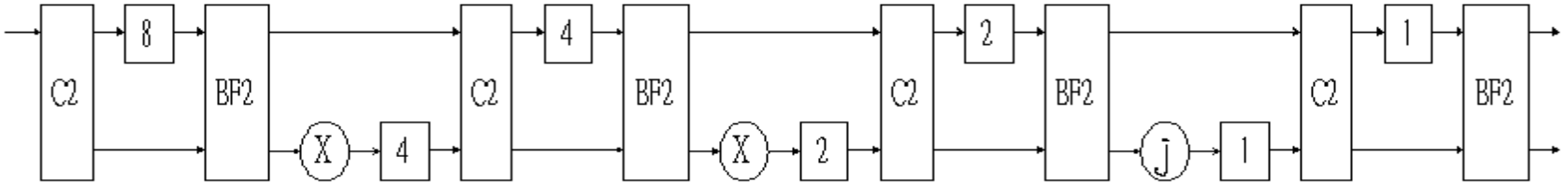


FFT – Parallelization, pipeline

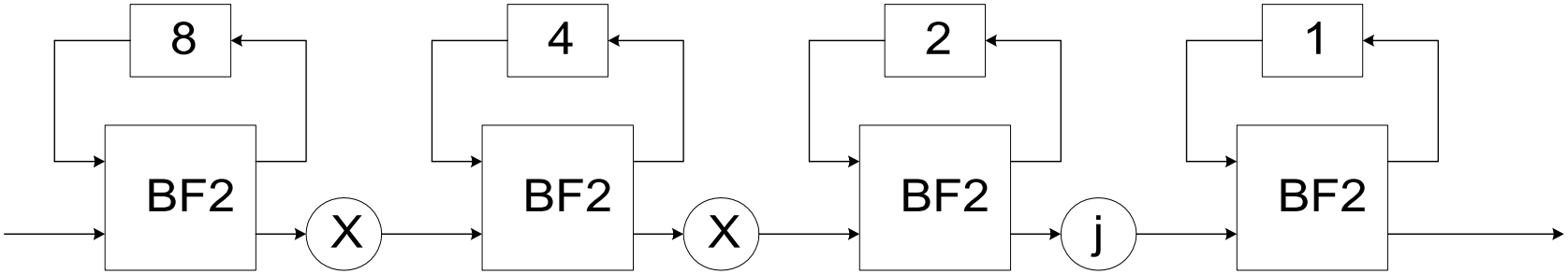
- Parallel – Pipeline FFTs
  - Pipeline FFT is very common for communication systems (OFDM, DMT)
  - Implements an entire "slice" of the FFT and reuses-hardware to perform other slices
  - Advantages: Particularly good for systems in which x(n) comes in serially (i.e. no block assembly required), very fast, more area efficient than parallel, can be pipelined
  - Disadvantages: Controller can become complicated, large intermediate memories may be required between stages, latency of N cycles (more if pipelining introduced)

FFT – Parallelization, Circuits

Radix-2 Multi-path Delay Commutator: R2MDC



Radix-2 Single-path Delay Feedback: R2SDF



FFT – Parallelization, Circuits

Hardware Resource Requirements

Following table gives a short overview about the hardware needs of previously specified hardware elements

	Complex Multipliers	Complex Adders	Memory	Control Logic	Comp. Efficiency	
					add/sub	Multiplier
R2SDF	$\log_2 N-2$	$2\log_2 N$	N-1	Simple	50%	50%
R2^{2}SDF	$\log_4 N-1$	$4\log_4 N$	N-1	Simple	75%	75%
R2MDC	$\log_2 N-2$	$2\log_2 N$	3N/2-2	Simple	50%	50%

FFT

- Depending on the use of the FFT algorithm the FFT could be implemented in various methods
- The parallelized method could improve the performance of the traditional Fast Fourier Transform
  - Hardware implementation
  - Furthermore software implementation on
    - Nvidia CUDA platform
    - DSP chips
- The advantages is obvious in a real-time system

Conclusions

- Motivation
- Multicore systems
  - Definitions
- Survey of multicore systems, architectures
- Applications
  - Cellular automation demonstration
  - FFT implementation