Programmable Cellular Automata

End-User Parallel Programming (EUPP) to Map Naturally Parallel Applications → Massively Parallel Computer Hardware Architectures

Dr. John-Thones Amenyo

jta@york.cuny.edu

Dept. Math & CS

July2009

Programmable Cellular Automata Intermediate Formalism

Map: Application Trends → Architecture Trends

Application		Architecture
Distributed Sensor Networks	⟹	Parallel Computers
Distributed Robot Systems		Distributed Computers
Natural Computation		Multi-core Computers
Autonomic Computing		Many-core Computers
Parallel Computing		

"Bronto" Scale Computing Spectrum & Gradations:

Exa-Scale + Peta-Scale + Tera-Scale + Giga-Scale Computing

Programmable Cellular Automata Architecture

	P: Processing *IPS, FLOPS*	M: Memory *Bits, Bytes*	S: Switching Comm.*Bps*	C: Control Coord. *Cops*?	IO: Input / Output *Pixels, Acels etc.*
K (kilo) (10^3)	KIPS	KBytes	Kbps		Kpixels
M (Mega) (10^6)	MFLOPS	MBytes	Mbps		**Mpixels**
G (Giga) (10^9)	GFLOPS	GBytes	**Gbps**		Gpixels
T (Tera) (10^{12})	**TFLOPS**	**TBytes**	Tbps		Tpixels
P (Peta) (10^{15})	**PFLOPS**	PBytes	Pbps		Ppixels
E (Exa) (10^{18})	EFLOPS	EBytes	Ebps		Epixels

IPS – Instructions/Sec; Bps – Bits/Sec; Pixels – Picture elements; Acels – Action elements, FLOPS – Floating Point Operations/Sec; Cops – Control Operations/Sec: Via Parallel / Modular / Distributed Designs

Programmable Cellular Automata
Applications

Computational STEM; Grand Challenges; Simulations; Sci. Data Visualization (Viz.); Bioinformatics; Neuroinformatics; Computational Connectomics; Immuno-informatics; Medical Informatics; VLDB; Large-scale RFID; GIS; Digital Data, Content, Media, Multi-media; Large scale Modeling; Large scale Data Mining
HPC; Distributed Sensor Networks; Distributed Robot Networks; Sensate Media; Multi-Touch Displays; Computational Intelligence
Network Computing; Internet; Web Services; Cloud Computing; Network Storage; Grid Computing; Tera-Grid; MPI; CC++; Ct
Desktop Tera-scale computing; Antenna Arrays; SDR; Distributed Sensor / Robot Networks; Ad hoc networks; VLSI Algorithmics, FPGA, DSP, ASIC chips, Reconfigurable Computers

HPC – High Performance Computing; VLDB – Very Large Scale DB; RFID – Radio Frequency Identification;
GIS – Geographical Information Systems; MPI – Message Passing Interface; SDR – Software Defined Radio
FPGA – Field Programmable Gate Array; DSP – Digital Signal Processing; ASIC – Application Specific IC

Programmable Cellular Automata Applications

Computational Biology	Sequence Matching, Comparison, Analysis; (Virtual) Models & Simulations: Cell, Biochemical / Metabolic Pathways, Gene Expression Networks, Immune Systems, Brain; Protein Folding / Conformation / Dynamics; *In Silico* Experiments.
Computational Chemistry	Drug Design; Pharmaco-genomics; Modeling: Protein Functions; Protein Engineering
Physics & Astronomy	Search for Exo-planets; Modeling: Stellar & Galactic Evolution and Dynamics; Hydrodynamics, Fluid dynamics; Molecular Dynamics
Earth Science; Oceanography	Signal Processing: Remote Sensing, Radar, Lidar, Sonar; Modeling: Climate Change; Weather dynamics, prediction/forecasting
Env. / Ecol. Science	Modeling, Simulations, Signal Processing
Economics	Modeling: Econometrics; Financial; Global / Regional / National Econ.
Engineering/ ICT/IT	"@Home" Projects, PC Harvesting; Data Mining; Distributed Robotics; Cryptography; Nano-materials modeling, Desktop supercomputing.
Military	Distributed Sensor Clouds; Battlefield Management Systems; Logistics
Consumer	Games; Image / Multi-media / Digital Content Processing; Lifetime Data Proc., Entertainment

Programmable Cellular Automata Applications

IT Consumers: Home & Institutions: Desktop Supercomputing, Image, Photo, Picture, Video, Media, Multimedia Processing, Biometrics Processing

Factories, Stores: Distributed Sensors; RFID Data Collection / Storage

Ships, Carriers, Tankers: Signal Processing, Image Processing

Aircraft: Signal Processing, Image Processing

Spacecraft: Signal Processing

Battlefields:

Env., Fields, Plots of Land, Airspace, Waterbodies: Env. Monitoring; Flora & Fauna tracking; Pollution management

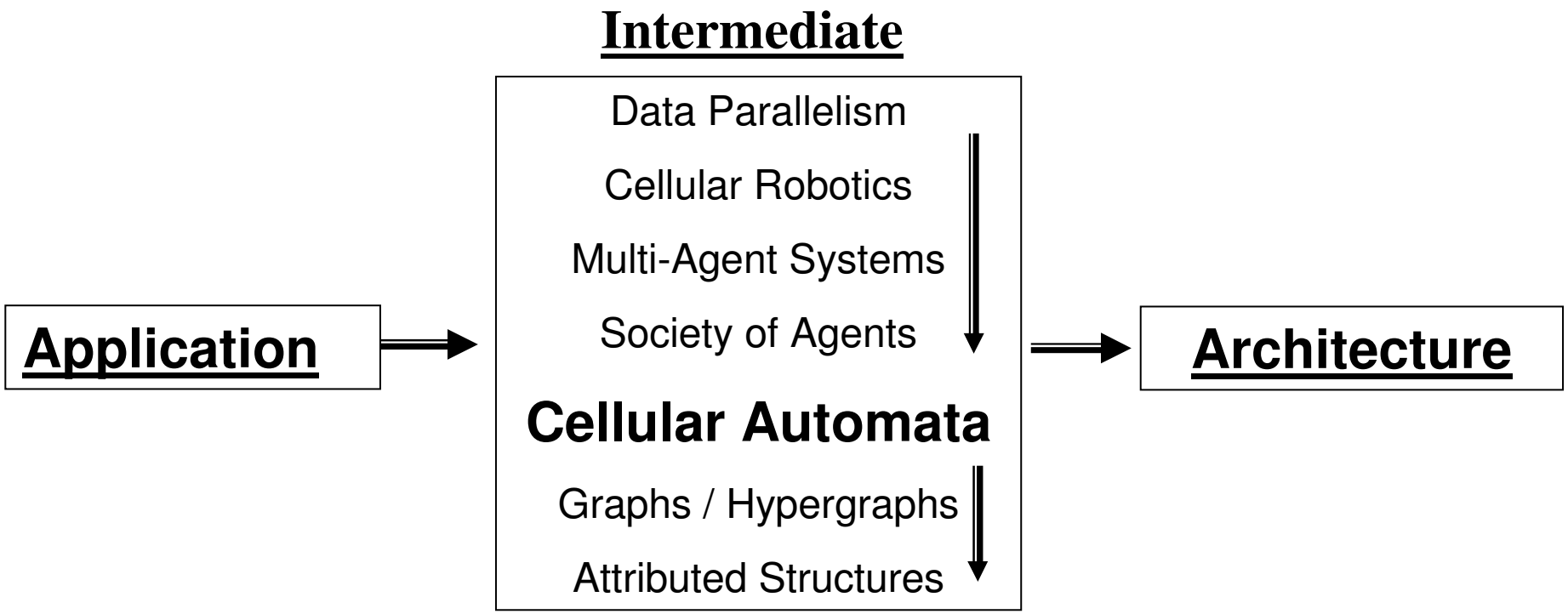
Sea, Ocean Segments:

Programmable Cellular Automata

Map: Application → Architecture
What Maps Into What? Why Does It Matter?

Application		Architecture
Data		Processors
Data Structures		Co-Processors
Functions		Processor Clusters
Operations	→	Memory, Storage
Routines, Scripts, Behaviors		Memory Clusters
Procedures, Algorithms		Input / Output Processors
Tasks, Transactions		Communications Processors
Processes		Switches
Threads		Synchronizations

Programmable Cellular Automata Intermediate Formalism



Programmable Cellular Automata

Distributed & Parallel Programming

Ian Foster's Methodology:

1 Application Tasks/Process Structure/Architecture Modeling

2 Intercommunication structures → Ad-hoc networks

3 Aggregation, containerization

→4 Mapping, allocation, assignment to computation execution structures ←

Map: Computational Algorithmic Structures → Computer Architecture Structures

Extend to: Automatic Programming, Generative Programming

Programmable Cellular Automata

Distributed & Parallel Programming: Lots of Traditional Metaphors Available

Array Processing	Matrix Algebra, Calculus	Signal Processing	Applicative Prog.
Vector Processing	Vector Algebra, Calculus	DSP, VHDL	Functional Prog.
Image Processing	Numerical Analysis	Spectral Analysis	Dataflow Prog.
Picture Processing	Diakoptics, Space Filters	List Processing, LISP	Systolic alg., arch
Video Processing	Network Analysis	Sequence Analysis	HPC
Graphics Processing	Circuit Analysis	Graph Algorithms	HAS
Pattern Processing	Tuple Programming	Transforms: FFT, Wavelet	MPI, OpenMP
Computational STEM	Data Flow Programming	Graph transformations	BSP
Computational Physics	CSP, CCS, CPP	Array-based Languages	Massively Parallel Proc.
Computational Chemistry	Process Calculi	Computational Chemistry	Data Parallelism
Computational Biology	Process Models	Computational Biology	Task Parallelism
Computational Math	OCCAM, Estelle, LOTOS	Computational Math	Instruction Level Parallel.

Programmable Cellular Automata

Programming for Natural Parallelism

Computation = Meeting (in Space-Time) of Data & Manipulations =

Computation = < data, function / operation> - Encounters

➔Data, Function or Both Flow to Site / Place of Computation

Manipulation of Collections of: Numbers, Codes, Symbols, Icons, Signs, Representations, Structures, Patterns, Images, Signals, Pictures, Waveforms, Tables, Arrays, Graphs, Hyper-graphs, Networks, Nested Structures, Metadata, Annotated / Tagged / Colored / Attributed Structures

Alternative View: Computation is Managed by <u>Operation System</u> (OS)

Threads, Multi-Threading, Concurrent Programming

Manage: Service, Application, Job, Task, Process, Thread, Fiber (micro-Thread), Instruction, micro-Instruction.

Programmable Cellular Automata

Data Parallelism

Algorithmic Skeletons/Kernels/Patterns

SISD

SIMD: Map, ApplyToAll

SPMD

MIMD

Reduce: Insert

Kronecker Product (KP)

Algorithmic Combinatorics

Compositions

Orchestration computing

Quipu/Khipu computing

Stream Processing

MapReduce (Google)

Barrier Synchronization

Inner Product, Outer, Vector, Norm, Exterior, Distance, Metric

Applicative Programming (Functional) Style: K. Iverson, J. Backus, T. More ("I.B.M") vs.

Recursive (Modern Functional Programming Style: Haskell, ML, Miranda, etc.)

Programmable Cellular Automata

Data Parallelism

p = (p1, …, pM), q = (q1, …, qN).
A = ((a11 … aM1) … (aM1 … aMN)). B = ((b11 … b1L) … (bK1 …bKL))

g = Compose(f1, … fQ).

(SISD (f) (p)) = (f p) ➔ (f(p)) ← Use A. Church's Lambda notation

(SIMD (f) (p)) = (Map (f) (p)) ➔ ((f p1) … (f pM))

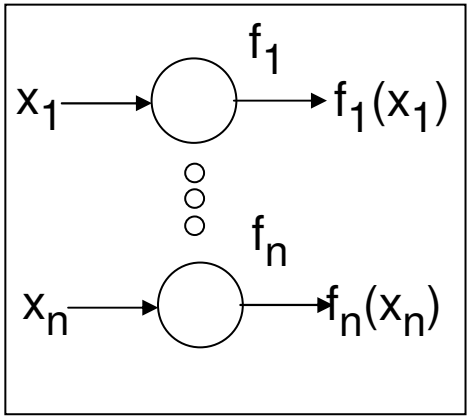
(MIMD (f1 … fM) (p)) ➔ ((f1 p1) … (fM pM))

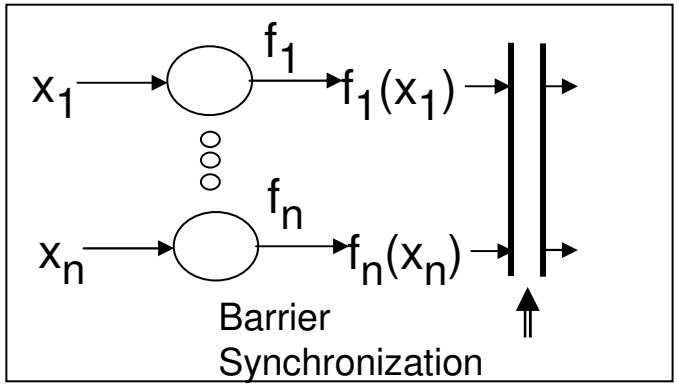
($MIMD_2$ (f1 … fM) (p)) ➔ ((f1 p) … (fM p))

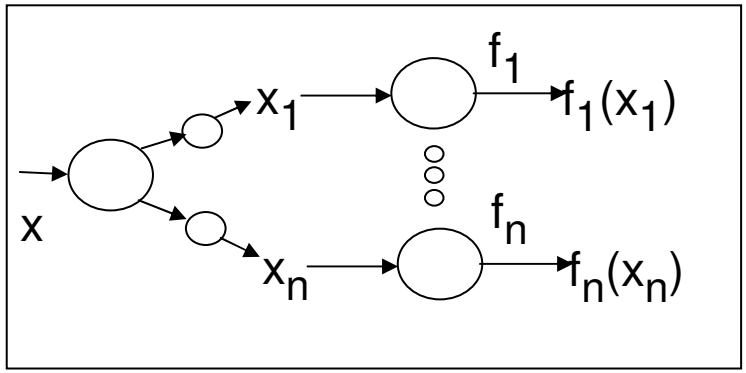
(SPMD (g) (p)) ➔ ((g p1) … (g pM))

Programmable Cellular Automata

Data Parallelism







Programmable Cellular Automata Data Parallelism

(∑∏-)Reduction Computing: - Aggregation, Accumulation

Data Collection = <x, y, a>; x = (x1, x2, …xN); y = (y1, y2, …, yM), N = M.

Task Collection = <f, g, h>; g = (g1, g2, …, gN); h = (h1, h2, …, hN).

Reduction Summation: Sx = (x1 + x2 + … +xN). Reduction Product: Px = (x1.x2…xN)

SP Reduction Product: SP(x, y) = (x1.y1 + x2.y2 + … + xN+yN)

Generalize: MISD(+, f, x) = (f(x1) + f(x2) + … + f(xN))

Generalize: $MISD_1$(f, g, x, y) = (f(…f(g1(x1, y1), g2(x2, y2)), …, gN(xN, yN)))

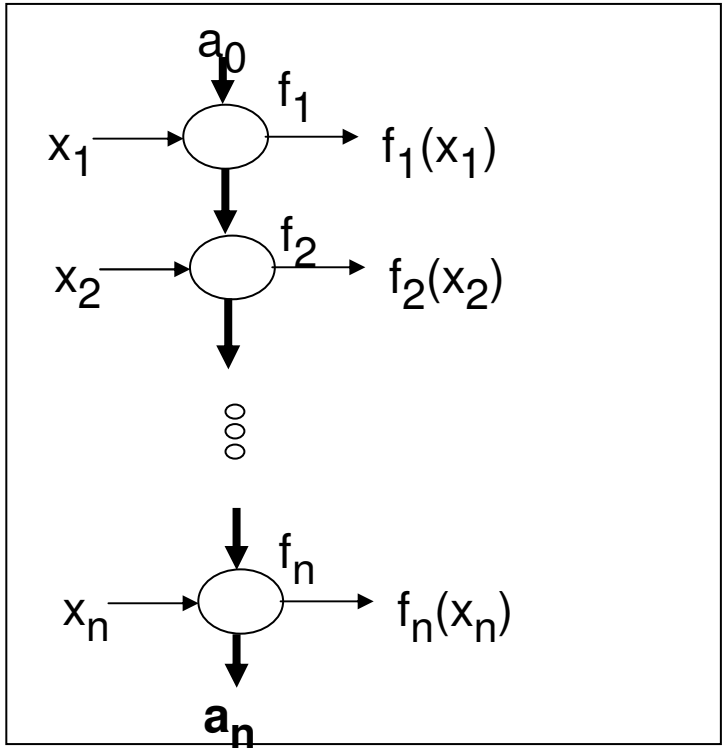
Generalize: $MIMD_2$(f, h, x, y) = (f(…f(h1(x, y), h2(x, y)), …, hN(x, y)))

J. Flynn's Taxonomy: MISD – Multiple Instruction Single Data stream; Pipelines, Tree-based computations

Uses: Scalarization, Transforms, Convolution, FFT, Outer Product, Vector Product, Cayley Product, Matrix Product, Tensor Product, Statistics: averaging, k-moments; Scans, Prefix-sums (partial/ongoing accumulations) , Max, Min

Programmable Cellular Automata Data Parallelism:

($\sum\prod$-)Reduction Computing: - Aggregation, Accumulation, Pipelines



Programmable Cellular Automata

Kronecker Product

a.k.a Tensor Product, Cross Product, Cartesian Product, Gibbs-Grassmann Combinatorial Product

p = (p1, …, pM), q = (q1, …, qN).

A = ((a11 … aM1) … (aM1 … aMN)). B = ((b11 … b1L) … (bK1 …bKL))

p X q = KP(p, q) = (((p1 x q1) (p1 x q2) … (p1 x qN)) … ((pM x q1) (pM x q2) … (pM x qN)))

A X B = KP(A, B) = (((a11 x B) (a12 x B) … (a1N x B)) …((aM1 x B) … (aMN x B)))

Watch Out! Extensive Use Ahead! For Everyone!

KP Overdo Computations ➔ Filter / Mask / Extract / Project those of interest

➔ Symbolic (Lazy) Computational Until On-Demand of Some Calc

Programmable Cellular Automata

MapReduce

Compose(Map/SIMD, Reduce/Reduction)

p = (p1, …, pM), q = (q1, …, qN). (f). g = Compose(f1, … fQ). (h), h is a binary (2-adic) operator

A = ((a11 … aM1) … (aM1 … aMN)). B = ((b11 … b1L) … (bK1 …bKL))

Recall: (Map (f) (p)) ➔ ((f p1) … (f pM)). **(Map (g) (p)) = ((f p1) … (f pM)).** (SIMD)

Recall: (Reduce (h) (p)) ➔ (p1 h p2 h p3 h… h pn) ➔ (h(pM, h(… h(p3, h(p2, p1))…)))

(MapReduce (f, h) (p)) ➔ (Reduce (h) (Map (f) (p))) ➔ (Reduce (h) ((f p1) … (f pM)).

Programmable Cellular Automata

Composition: Quipu Orchestration

Compose(Map/SIMD, Reduce/Reduction)

p = (p1, …, pM), q = (q1, …, qN). (f). g = Compose(f1, … fQ). (h), h is a binary (2-adic) operator

A = ((a11 … aM1) … (aM1 … aMN)). B = ((b11 … b1L) … (bK1 …bKL))

[Compose({SISD, SIMD, SPMD, MIMD}, {SISD, SIMD, SPMD, MIMD})]*

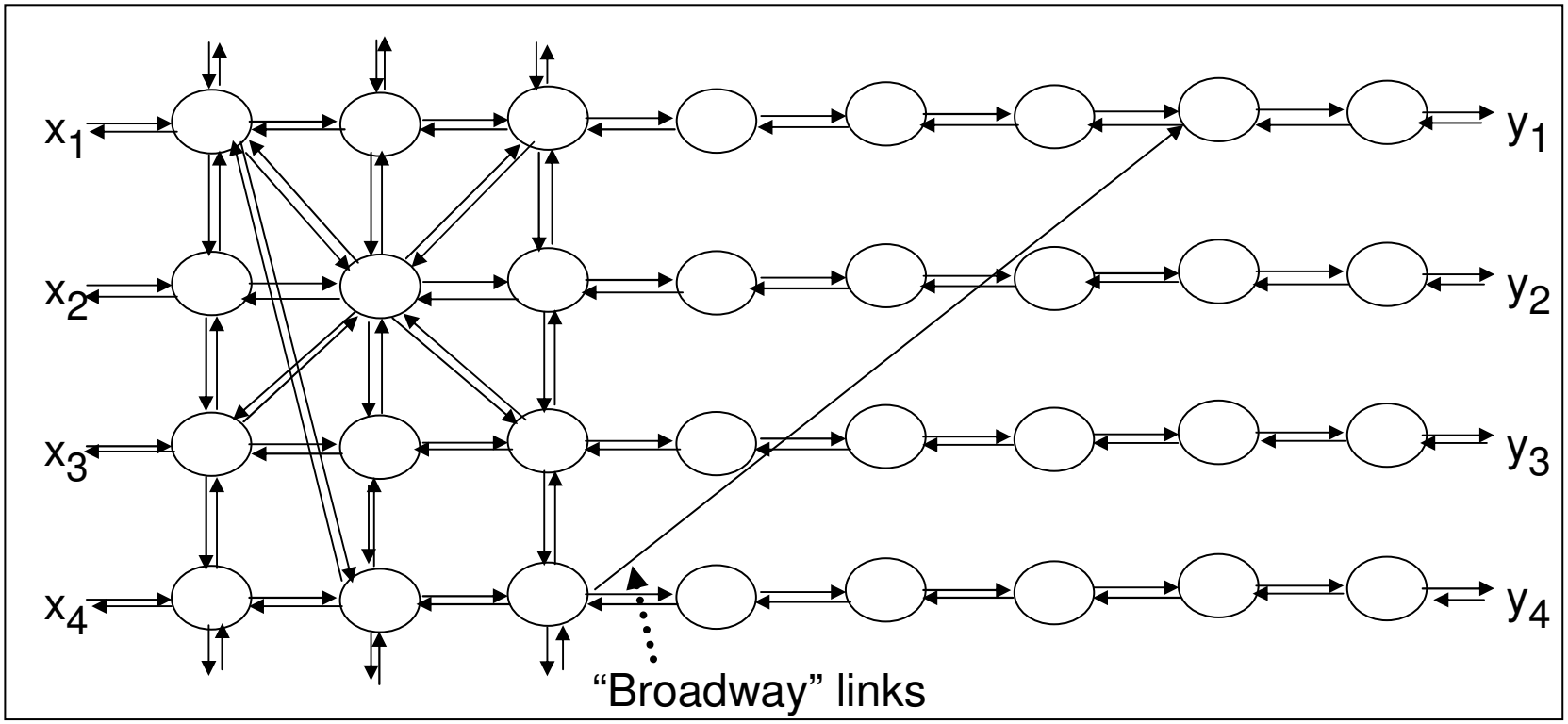
= [{SISD, SIMD, SPMD, MIMD} →IntereXchange → {SISD, SIMD, SPMD, MIMD}]*

IntereXchange:

- **Flow & Inter-Communication Patterns: pt-2-pt, pt-2-mpt, mux/demux, broadcast, multicast, scatter, gather, partition, bipartite, multi-partite, MIN, switching**
- **Synchronization & Inter-coupling Patterns: barrier, local, global**

Programmable Cellular Automata
CA Computing:

Computing over Spatial & Topological Structures of Compute Nodes



Programmable Cellular Automata
CA Computing:
Computing over Spatial & Topological Structures of Compute Nodes

Cellular Automata (CA)-like Computing: - Computing over Spatial & Topological Structures of Compute Nodes.

Node Collection = <n>; n = (n1, n2, …nK); Each node n has a neighborhood N(n) – set of direct (1-hop) neighbors. Could use Incidence Matrix & Logical Connectivity Graph repr. instead. At any time (assume discrete), each node n has a "state" <Mn, Xn, Yn > = <Internal/Memory, Input, Output>, and associated state transition rule(s).

Simplest Case (Markovian): At any time, the next state of a compute node depends on its own current state and the most recent states of the nodes in its neighborhood. Use the task collections to calculate the state transition elements, e. g., via Embarrassingly Parallel or Reduction Computations.

Generalize (non-Markovian): Neighborhood topologies of individual nodes change over time.

Generalize: Each node n has a distribution of "distance-based" multiple neighborhoods: $N_{d1}(n),\ldots,N_{dk(n)}(n)$. "Mesh-of-trees" paradigm

Generalize (non-Markovian): State transitions at a node are memory-sensitive, depending on the history of the state transitions at the nodes and/or at its neighborhood(s). Systolic algorithms

Uses: Hidden Markov Models; Artificial Neuro-networks (ANN); Attribute evaluation in parse tree / syntax trees of Attribute Grammars; Cellular automata models; data flows on n-cubes, hypercubes, routing networks, multi-commodity flow networks; Traversal of topological, spatial structures and lattice structures. Data dependency models of spreadsheets. Task Parallelism; Shifts and Rotations of data; Protein folding models? Transputer/OCCAM/CSP models.

Programmable Cellular Automata
CA Computing

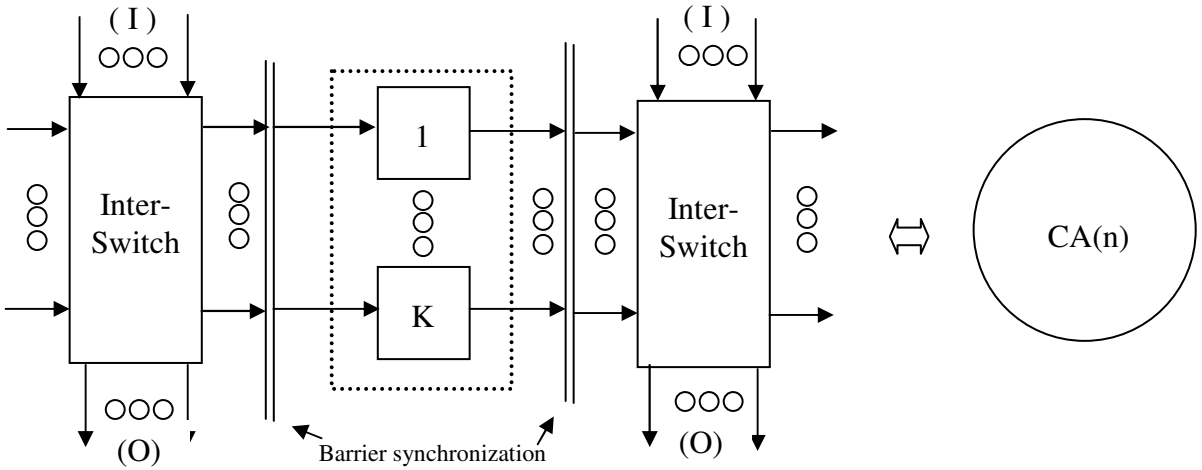


Fig. 2: (Locally) Synchronous cellular automaton (CA) stage [CA(n)], at time epoch n

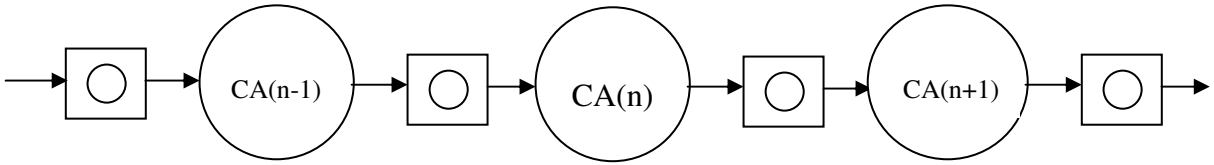
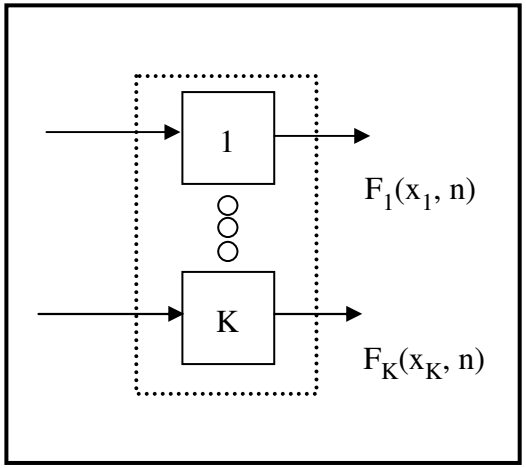


Fig. 3: (Chaining) Sequential composition of cellular automaton stages



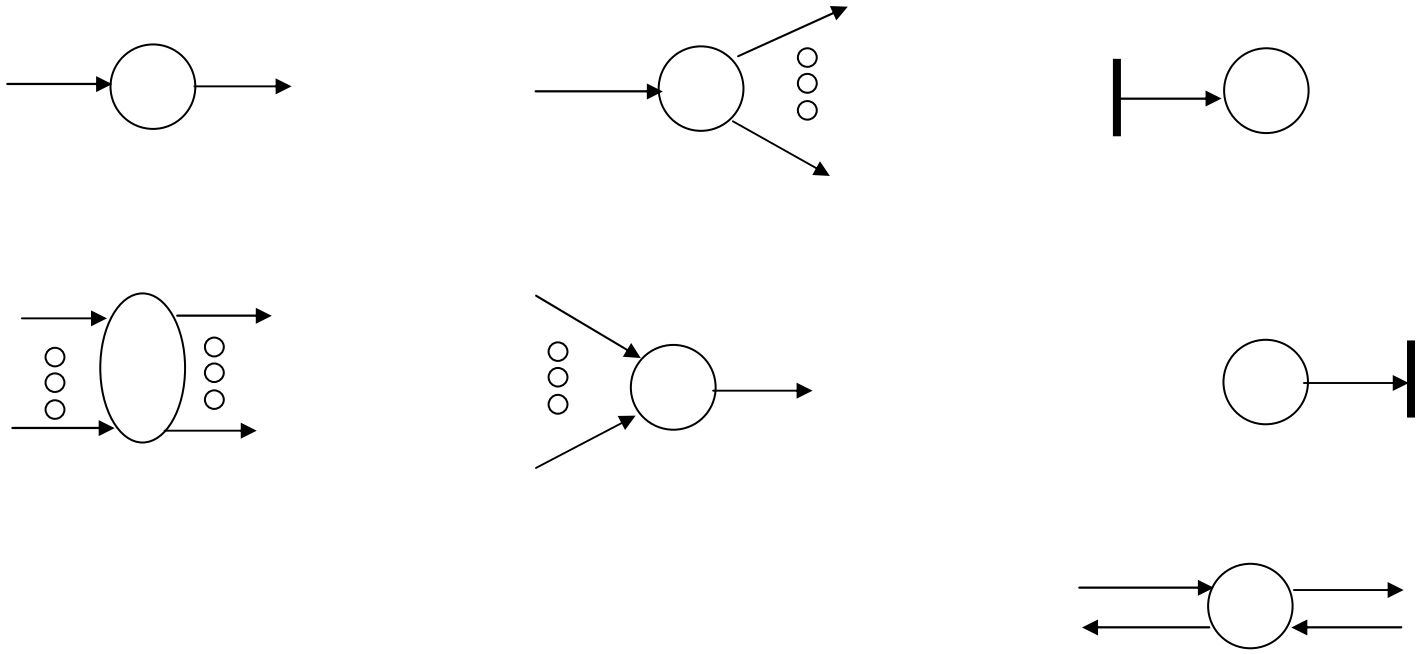
Programmable Cellular Automata Flow Manipulations

x → (S) → y	Relay, Sender/Receiver, Transceiver, Transponder, Bridge, Gateway, Connector, Inter-link, Inter-coupler, Portal
(x…xm) → (S) → (y1…yn)	Switch, Rendezvous, Mailbox, Exchange, Clearinghouse, ?Bank, Router, Permutation, Re-combinations
x → (S) → (y1…yn)	Broadcast, multicast, Expander, Dispatch
x → (S) → (y1…yn)	Demux, Scatter, Unpack, Dispatch, Queuing
(x1…xm) → (S) → y	Gather, Broadcatch, Aggregate, Superimposition, Mux, Concentrator, Pack, Marshall, Read-All, Read-Any, Cyclic service
(x,v) ←→ (S) ←→ (y,u)	Synport, duplex / half-duplex channel, parallel bus

Distinguish task (functional & relational) building blocks (P,M,C,IO)-modules, from flow control building blocks (S,C)-modules. For flow, tackle issues of inter-flow concurrency, coordination and synchronization.

State-of-the-art: MPI and OpenMP;

Programmable Cellular Automata Flow Manipulations



Programmable Cellular Automata

Other Intermediate Programming for Natural Parallelism

Interacting Units – Style 1

CA: Cellular Automata

AN: Automata Networks

SAN: Stochastic Automata Networks

CNN: Cellular Neural Networks

ANN: Artificial Neural Networks

DSN: Distributed Sensor Networks

DRN: Distributed Robot Networks

MAS: Multi-Agent Systems

SOM: Society of Mind, Bots, Agents, Bots

Data Dependency Models (Spreadsheets)

Swarms, Computational Ant Intelligence

Interacting Units – Style 2

OOPP: Object-oriented Parallel Programming

CSP, CCS, CPP

OCCAM structures

Process Calculi

Petri Nets, Transition Networks

ATN: Augmented Transition Networks

Programmable Cellular Automata

End-User Parallel Programming (EUPP): What to Expect In the Next Few Years

- Human Programmers (Non-Professional, Occasional, "End-Users")

+

- Computer Programming Assistants / Aides / Guides

  Human → User-oriented → Intermediate → Machine-oriented→Arch.

- Automatic Programming: Intelligent Program Translators and Transformers: Mappers, Optimizers, Tuning, Adaptors
- Affinity Computing (Anticipatory Computing)
- Transitions: Push → Pull (On Demand) → Affinity
- Cognitive Software / Affective Software, Meta-Enabled Applications
- DSL (Domain Specific Languages) vs. DSEL (DS Embedded Languages)

Programmable Cellular Automata:

Summary

Objects/Concepts→ Components+Containers → Agent/Ants/Bots → Processes/Tasks
Map: Nodes / Vertices → Bots / Processes
Map: Edges / Hyper-edges → Bots / Processes

(EUP) End-User Programming: Professional / Non-Professional\

(DSL: Domain-Specific Languages)

↓

Automated / Computer-Assisted Translation / Mapping

↓

Computation Execution / Implementation

Programmable Cellular Automata:

End-User Parallel Programming (EUPP) to Map Naturally Parallel Applications Into Massively Parallel Computer Hardware Architectures

Thank You