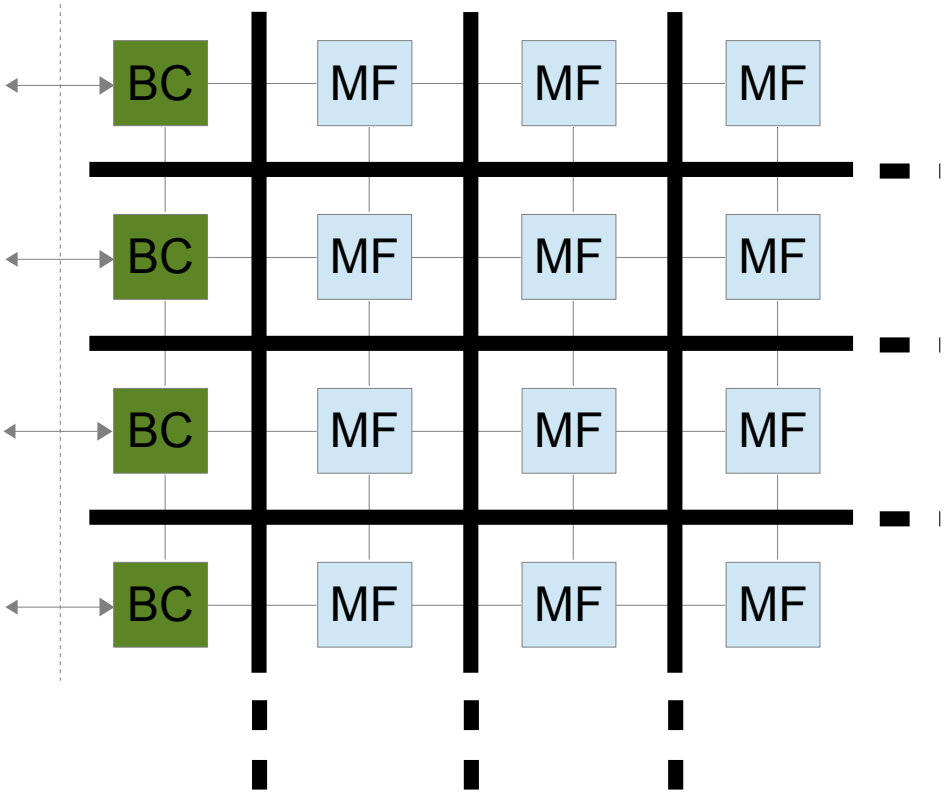
A Paradigm for Building and Programming Scalable FPGA based Numeric Processors

Fritz Mayer-Lindenberg, TUHH

Configurable 'logic': the FPGA



= single-chip integrated reservoir of many* simple logic cells

multi-function MF cells have individually configurable funktions, no sequential control

border BC cells are configurable interfaces to the external pins

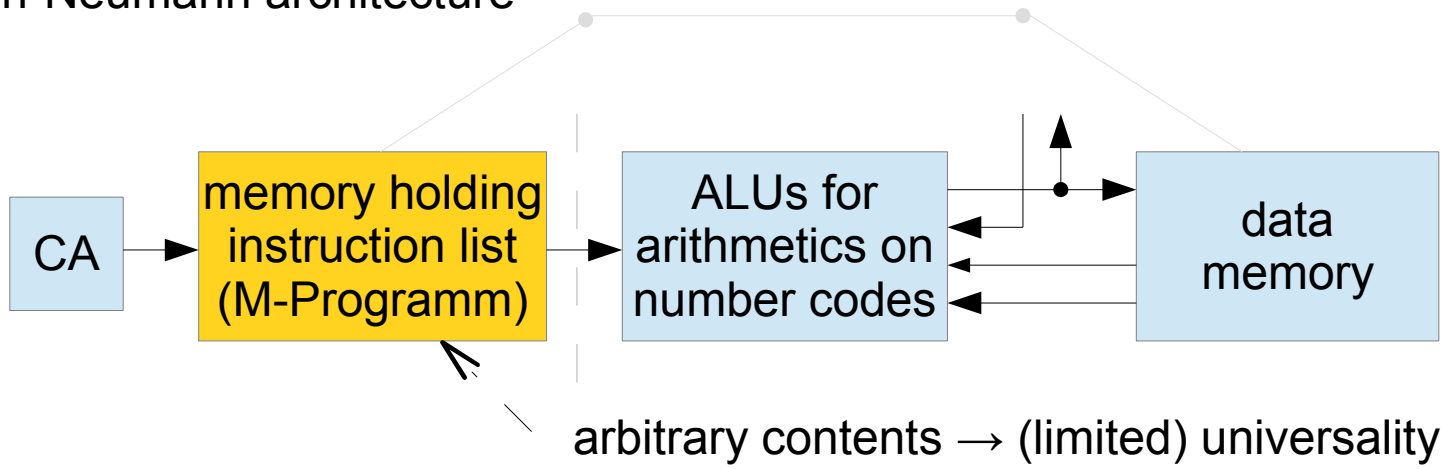
elektronically configurable wiring network,
separate clock signal networks

at a lower density: complex arithmetic functions, memory blocks, fast serial interfaces

*..10k...100k..

The FPGA cells can be connected via the wiring network into compute circuits (ALUs) for arbitrary number codes and be equipped with control circuits to become programmable 'soft' processors.

Von-Neumann architecture



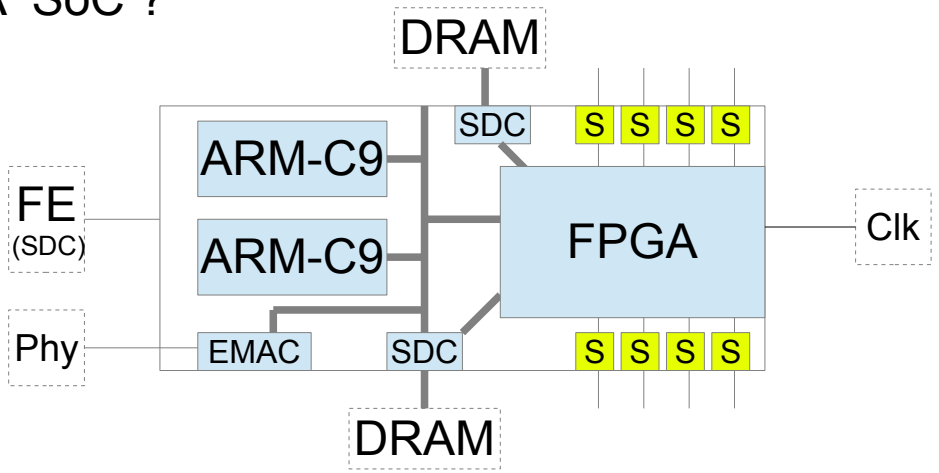
- von-Neumann CPU = von-Neumann-system w/o memories
- all components highly integrated, fixed ALU resources
- example: ARM CPU, contains ALU for integer and floating point codes
  hierachic memory: data registers – caches – main memory
- von-Neumann architecture and FPGA architecture are both architectures for 'universal' computers programmable/configurable for many applications
  can 'emulate' each other

What is in a low-to-mid size ARM+FPGA SoC ?

- 2 Cortex-A9 Cores
  each with FPU + 'Neon' cop
  Caches, RAM, DMA, DRAM-C

- standard interfaces Uart, CAN
  Ethernet-MAC (ext. Phy)
  SDC-Interface, USB (ext. Phy)
  Interface to FPGA section

- FPGA, depending on chip size:
  > 70k Cells, low skew clocks
  > 200 Multiplier/ALU 18-bit .. 2000
  > 500 DP-RAM blocks 1kx10
  separate DRAM controllers
  high-speed serial interfaces .. 0..8..16..



DRAM, FE, Phy, ClkGen are external

Latest generation SoC chips have more cells, RAM, multipliers, higher speed clocks and interfaces, and more and faster ARM cores

Documentation: (nearly 10k pages)
- SoC/FPGA HW
- ARM HW
- SoC/FPGA tools
- ARM tools

ARM: 600-800 MHz (cheap)
FPGA: 100-200 MHz (expensive)

A 'pure' FPGA can implement SoCs, too; (low end) FPGA performance ~ ARM.

Focussing on numeric applications (DSP, robotics, HPC, cryptography, graphics …)

- not on general purpose computing, symbol processing, dynamic data allocation

→ → :

Paradigm Part I: - exploit the arithmetic components to configure many ALUs
various number codes, complex, fused operations, SIMD

- extend them by sequential controllers to soft CPUs
use simple controllers and internal memory blocks

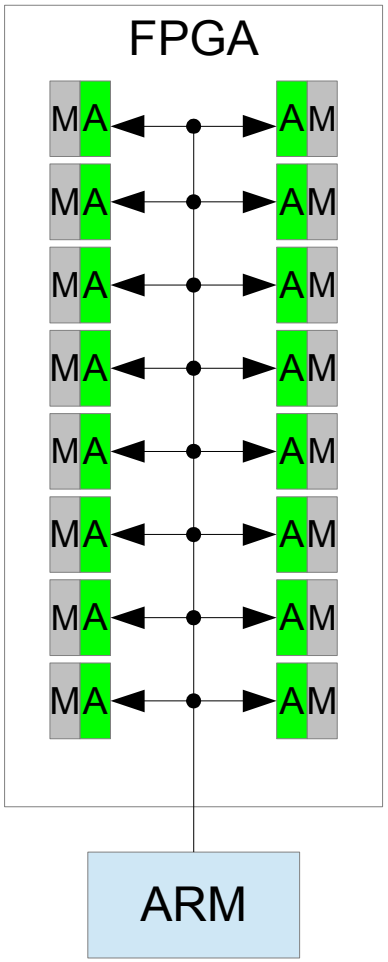
- connect them by data interfaces (→ MIMD+SIMD network)
need data only interfaces, point-to-point or buses

Back to primitive µctrl networks ??

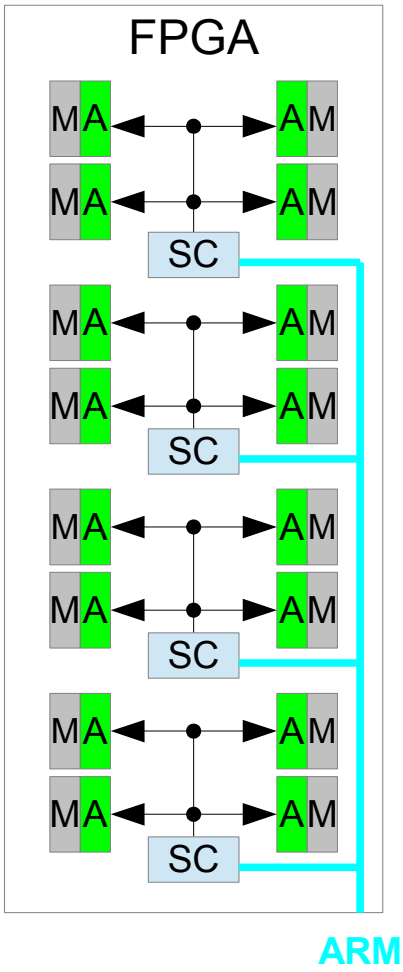
Reasoning: Every ALU must be used at a high rate in order to use the FPGA efficiently. This implies that there has to be sequential control for each ALU to select the data and to process them. Both controllers and interfaces should be fast and simple to save resources for more ALUs. Will need debugging/download support and external interfaces.

Control schemes for FPGA based ALUs

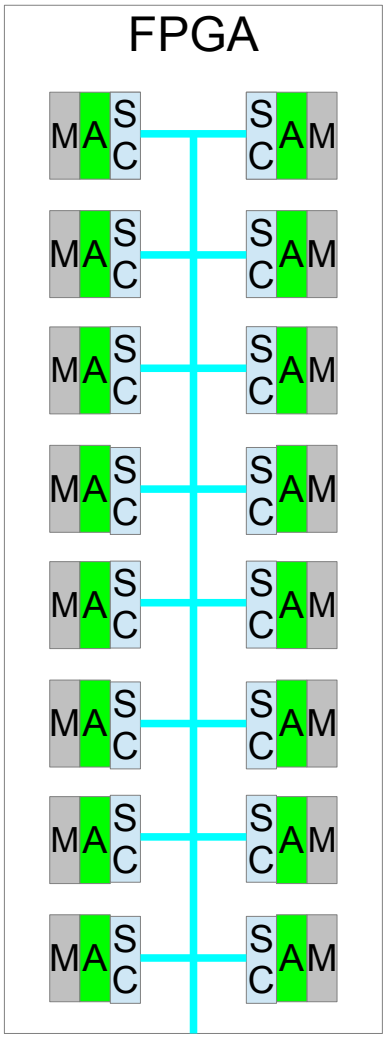
.. data I/O path not shown



SIMD, external control by ARM



SIMD/MIMD, ctrl by soft controller



MIMD, ctrl by soft controllers

Paradigm Part II: - simplify application programming by building on a HW OS predefining all standard functions, leaving the ALUs only

- **external memory interfaces**
- **host interface for debugging, downloads and I/O**
- **interfacing the ARM subsystem of an SoC**
- **sequential controllers**
- **external networking including routing switch function**

- **use external interfaces for FPGA networking and for I/O protocols w. low overhead required, implemented by the OS supporting ARM only SoC networks**
  use DMA-supported memory-to-remote-memory transfers
  need to route the host interface through the external network

Enforce the use of non-standard soft processors and NoC/ext.NW ???

Reasoning: The soft CPUs will implement various number codes, but can use the same sequential controller adapted to the OS and becoming part of it as it is standard then. The network will operate stand-alone connecting the memories. Processors move data in and out of them only. The use of the OS is beneficial for non-numeric applications, too.

Paradigm Part III: - **build scalable systems as networks of FPGA/SoC nodes**

using the available high-speed interfaces for data

and the special interfaces to the external memory chips

use low to mid size SoC nodes

- **configure the FPGA nodes from precompiled bitstreams in a local Flash memory**

  partial system reconfiguration by fully reconfiguring individual nodes

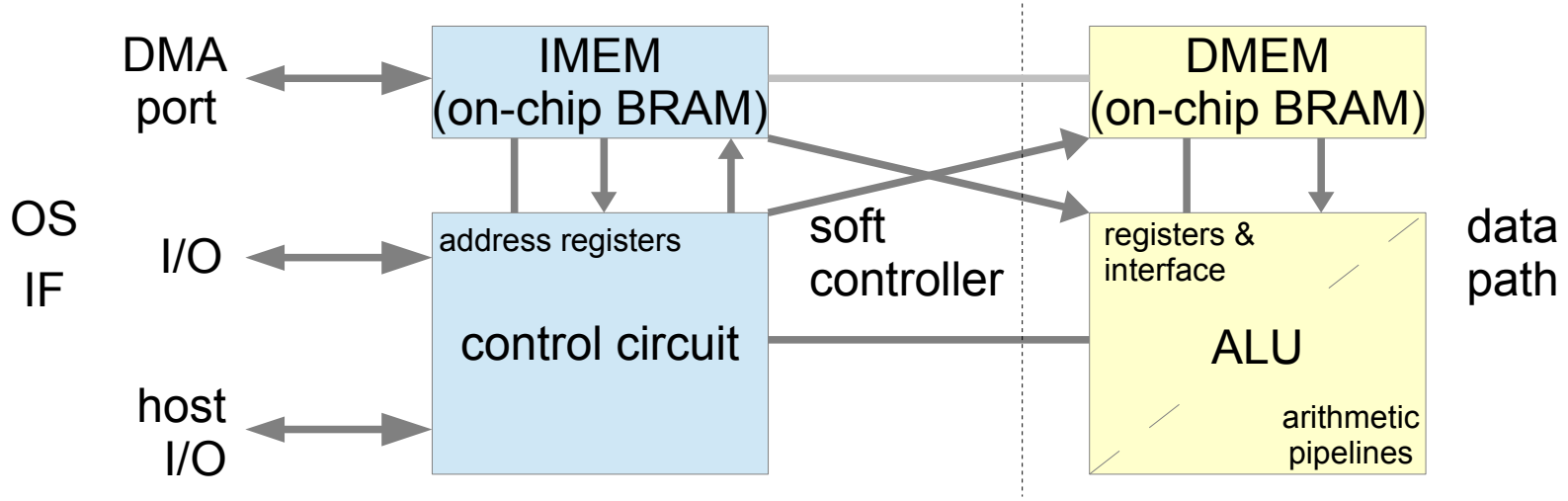
  application programming reduced to the parallel programming of the processors using some HLL; if some new ALU or non-numeric function is needed, this design task is handled separately

Not using partial FPGA reconfiguration, not using the same HLL for circuit design???

Reasoning: Low to mid size ease the compilation of new bitstreams, speed up configuration and reconfiguration, provide a higher memory bandwidth per processor, and may even result in cheaper systems. Low-to-mid size FPGA and SoC nodes are avilable with 8 to 16 High-speed interfaces allowing for fairly high node degrees and graphs of low diameters and communication overheads.

Soft controller design / modular soft processors

Required to: support non-standard ALUs, wide data codes
provide maximum ALU efficiency, parallel CF and mem.acc.
.. support multiple threads
be a low complexity circuit to allow for large MIMD sub nets
.. have a simple memory architecture



- controller performs instruction sequencing, memory control and I/O, 4 threads
- ALU data word size independent on controller address/index word sizes
- VLIW type instructions for ALU ops. executed in parallel with controller ops.
- no memory bus, no cache/MMU, DMA supported I/O to ext.mem.(SW caching)
- controller design adapted to FPGA resources (18/20-bit word size)

20-bit instruction set of the controller .. Altera version 'CPU1A', extended from 18-bit Xilinx version

- implements
  - control flow (conditional jumps) and code compression (calls)
  - manages 4 instruction threads w. register banks, c. switches
  - memory control (IRAM and DRAM)
  - I/O with ALU, MC, host

- operates itself on 20-bit integer/index/address data .. 6 lvl pipeline

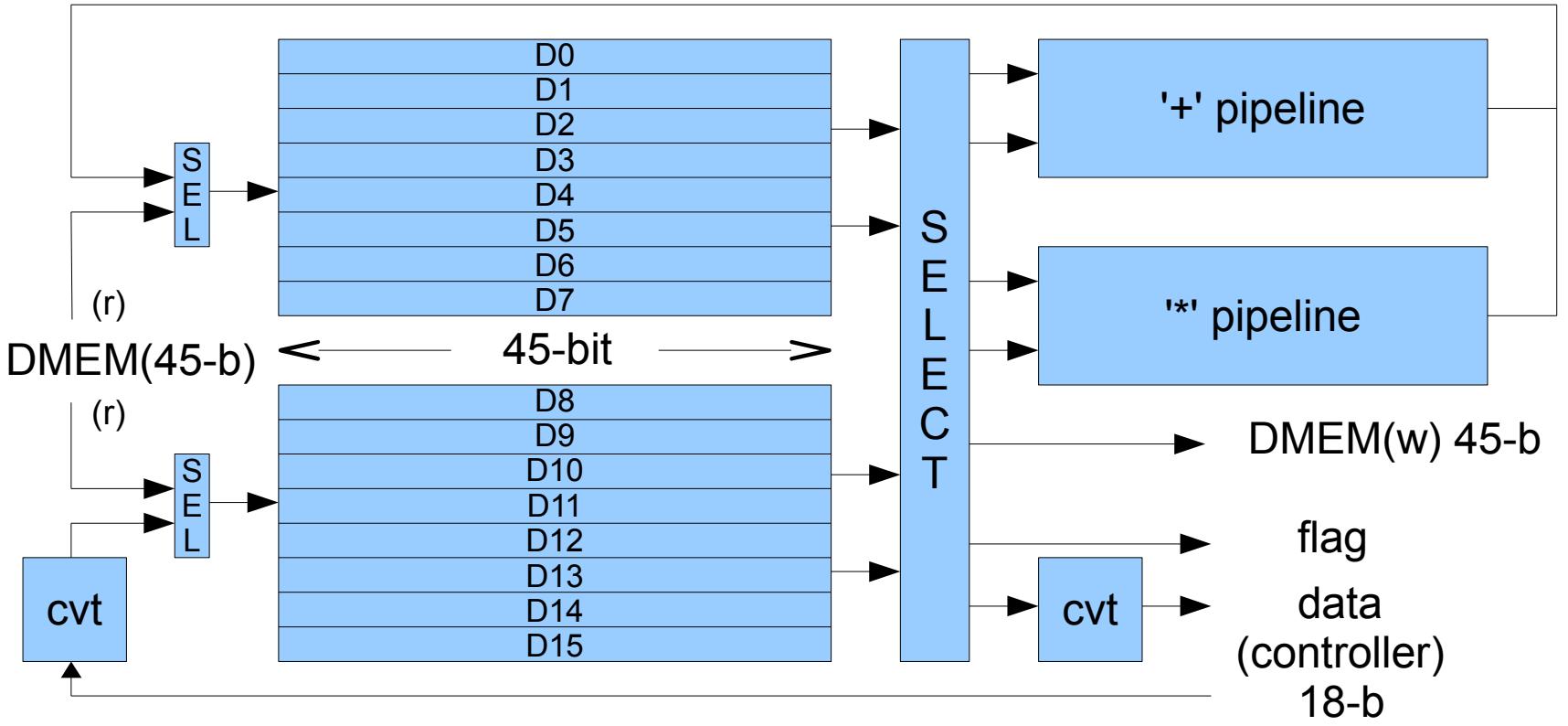
- uses internal IRAM/DRAM only, small memory spaces, SW caching, DMA channel

- code mapping (CPU1A part only, ALU receives extra instruction word): .. easy decoding

x 0 0 i(17)	.. jump instructions	x: allow context switch
x 0 1 0 i(16)	.. memory access instructions	
x 0 1 1 0 i(15)	.. operations of the CPU1A (address) ALU	
x 0 1 1 1 i(15)	.. internal and external IO, DMA control	
0 1 0 0 d(16)	.. read data block from MC	
0 1 0 1 d(16)	.. send block from MC via network	
0 1 1 z d(16)	.. write data block to MC (z: empty block, no DMA)	
1 1 d(18)	.. register mapping instruction	

Example: Floating point ALU / data path attaching to soft controller (Spartan-6)

45-bit number codes: 34-bit mantissa+sign, 9-bit exponent, no non-normals, round → 0
supporting parallel chained +/* operations and dual memory accesses (effic. dot product)
registers and data RAM are 45-bit wide, data RAM with one 'rw', one 'r' port, 4 threads



V144 vector ALU

- operates on 144-bit wide data words composed of 4 36-bit components
  components are unsigned 34-bit fixed point magnitudes, a sign bit, and a spare bit

  |s_---- m(34) ----|s_---- m(34) ----|s_---- m(34) ----|s_---- m(34) ----|

- data words encode
  - real 3- or 4-vectors
  - complex 2-vectors
  - real or complex numbers or quaternions

- CPU handles and computes block exponents for individual data words or vectors
  - exponents stored separately from the vectors in controller memory

  ALU performs 4 multiply and 4 add operations in parallel

  CPU is ≈12% of circuit overhead only – can use nearly all multipliers of the 150T
  → double peak performance (block fp) (e.g. 12+6 processors, multiplier limited)

Quaternions (IH): a 4D associative algebra over the reals with the basis 1, i, j, k

$$q = (a,b,c,d) = a + i\,b + j\,c + k\,d$$

Multiply table

*	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

(non commutative)

Conjugation $\quad \text{conj}(a,b,c,d) = (a,-b,-c,-d) \qquad (a,b,c,d)*\text{conj}(a,b,c,d)=a^2+b^2+c^2+d^2$

IH contains { a+ib }, is 2D complex vector space, right multiplies complex linear

$$x \rightarrow x*q$$

... IH contains { a+jc }, { a+kd } as well ...

unit quaternions form a group under '*', used to parametrize 3D rotations

Special cases for the quaternion product using 4 multiplies each:

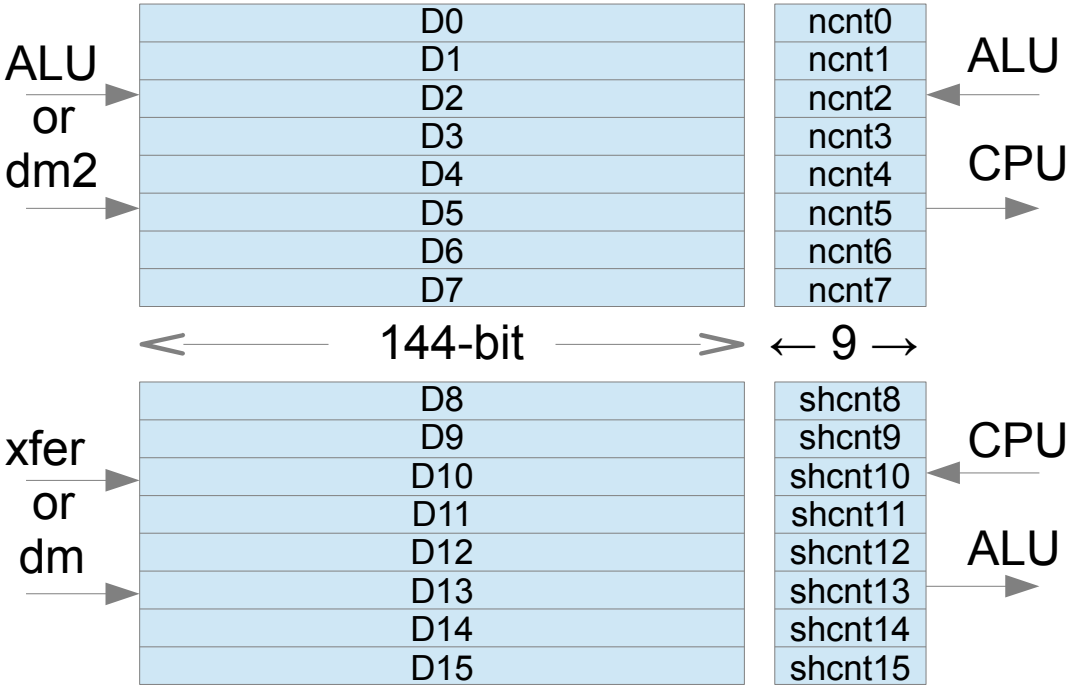
$$(a + ib)*(a' + ib') = (a*a' - b*b') + i(a*b' + b*a') \quad \leftarrow \text{ complex product}$$
$$(a + ib)*(jc' + kd') = j(a*c' - b*d') + k(a*d' + b*c') \quad \leftarrow \text{ completes } z*v$$
$$(jc + kd)*(a' + ib') = j(c*a' + d*b') + k(d*a' - c*b')$$
$$(jc + kd)*(jc' + kd') = -(c*c' + d*d') + i(c*d' - d*c')$$

(full quaternion product obtained by adding up the special cases)

ALU registers* and instructions

* separate for each context

fits into SLX9 (12 mpy, not fused, w/o butterfly)



Arithmetic instructions

dr=½(dtl+dth)	
drl=dsum(dt)	
dr=ds*dt	SIMD product
dr=ds+dt	SIMD sum
dr=dt–ds	SIMD difference
drl=dtl*dsl	complex product L
drh=dtl*dsh	complex product H
drh=½dth*dsl	quaternion product L
drl =½dth*dsh	quaternion product H
dr=dtl+/-w*dth	butterfly, w from ROM table

Transfer instructions (parallel)

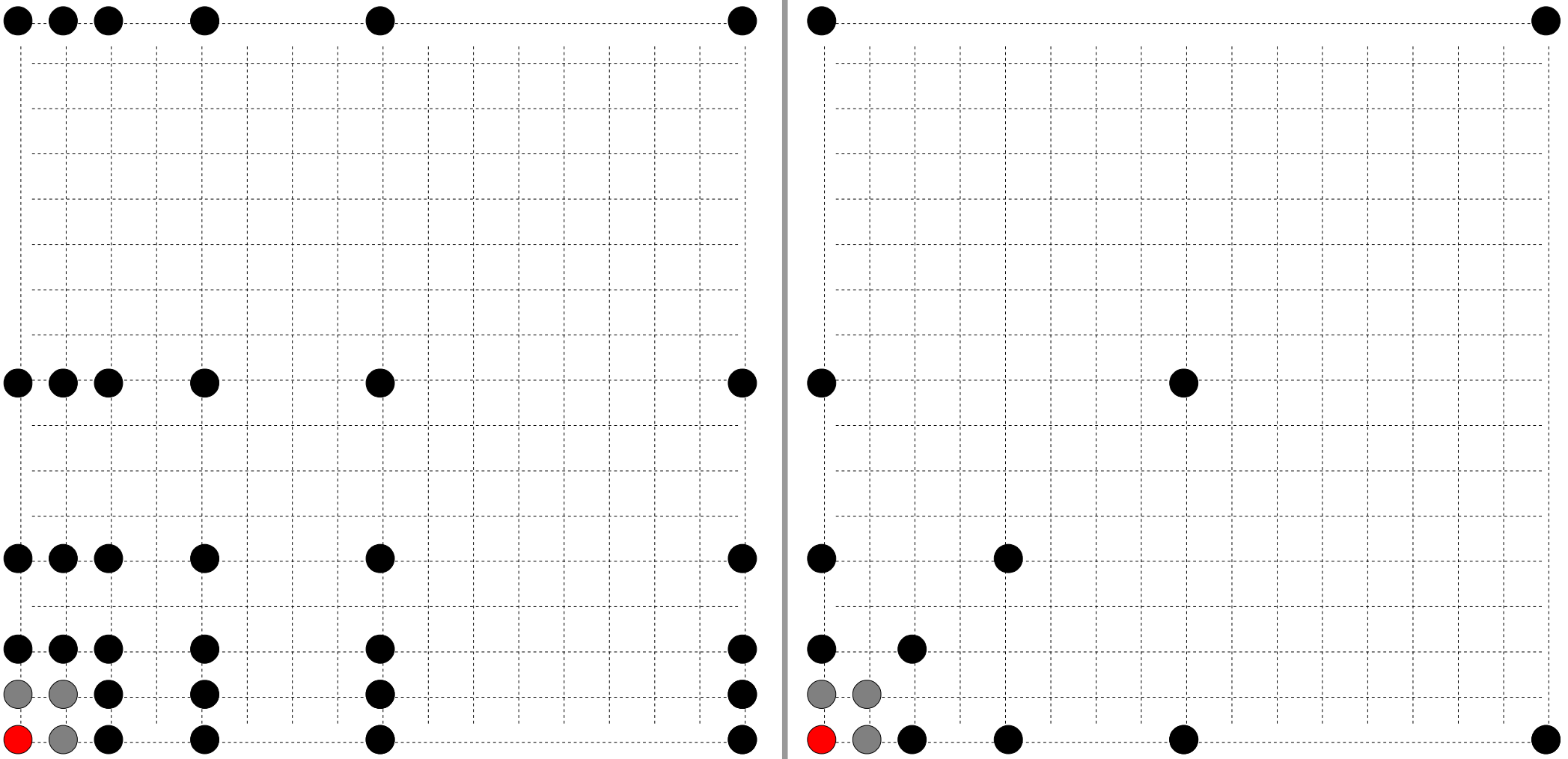
shift dr	use registered shift count
dr=dt	register move
dr = conj(dt)	complex/quaternion conjugation
rsh	shift with CPU
shc	shift count with CPU

3 more dot product instructions w/o parallel xfers

butterfly uses extra FFT address generator

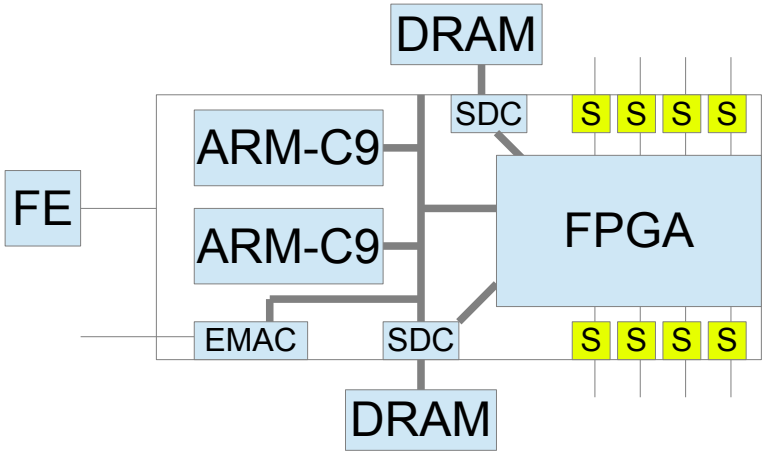
code distribution FP

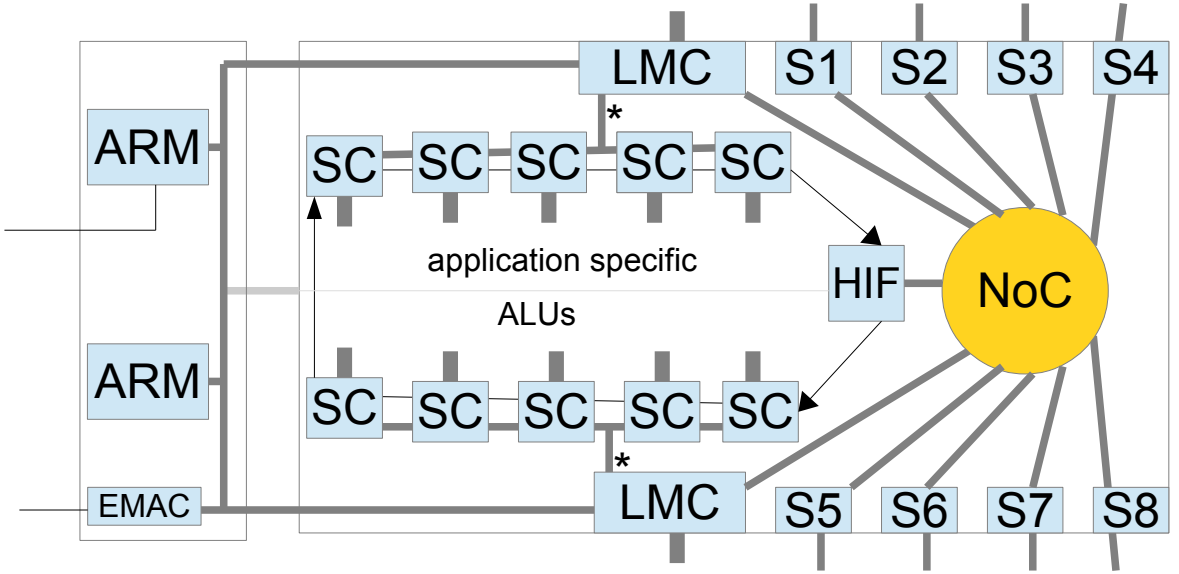
BFP



Rem.: In a normalized BFP vector only 1 component needs to be normalized.

HW-OS (SoC version)



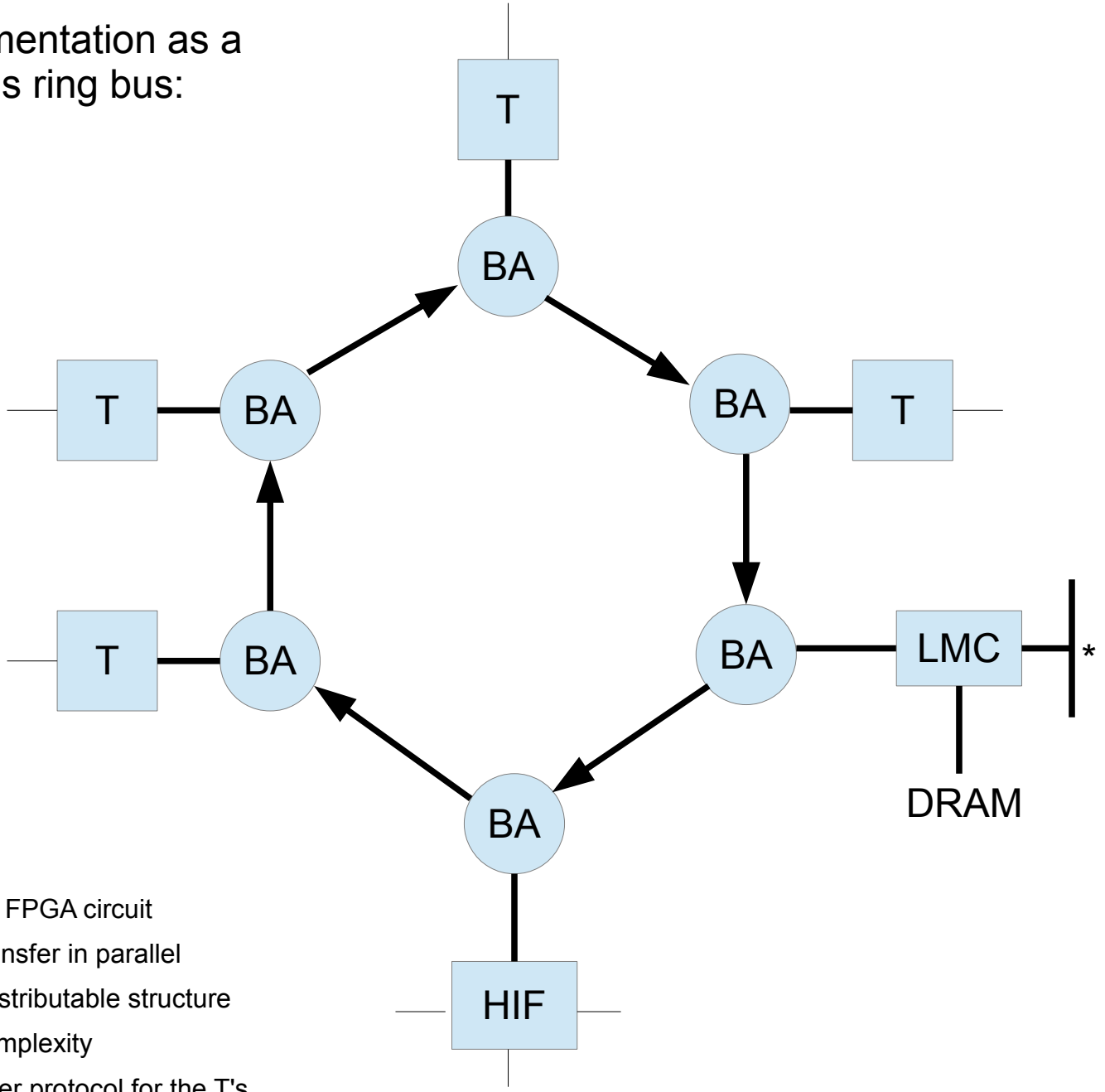


+ more interfacing support

- not all of the SCs need to be Connected to a LMC via its sub network (*)

- the application part can add more interfaces between SCs

NoC implementation as a synchronous ring bus:



synchronous FPGA circuit

segments transfer in parallel

distributed/distributable structure

moderate complexity

simple transfer protocol for the T's

Communications via the HW-OS

– The ARM in an SoC can access the host interface and take over the host role. It also configures the FPGA from Flash memory and provides an LAN interface.

– communications on command of an ARM or an SC to an LMC

local block address – target node/path – target LMC – target block address

block transfers occurs as a DMA transfer w/o further help by the processor
the routing path for a transfer is defined in the header word of the message

– P-P communications further use DMA cache transfers P↔M

data transfer P→M – block transfer – data transfer M→P

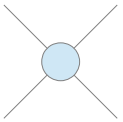
P↔M transfers occur in response to special commands to the LMCs
.. also for the caching of instructions

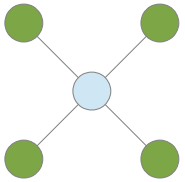
– FIFO transfers and handshakes are linked to special block addresses
communications can be extended to internal memory blocks of the FPGA

– protocol support for temporarily isolating the FPGA for reconfiguration

– the HW-OS defines a logical network of LMC nodes each with a sub network linking it to small, mutually disjoint groups of ARMs and SCs

Modeling network architectures:





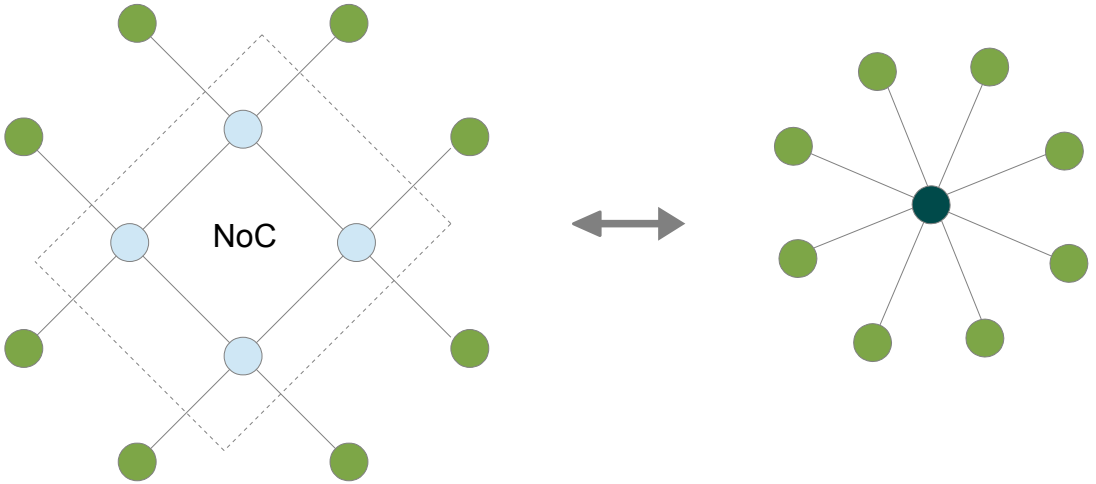
a scalable heterogeneous HW architectures is

- defined by a selection of types of system building blocks

  building blocks may compute or not node attributes and edge attributes

- systems/networks of the architecture are those built from blocks of these types, open interfaces being allowed

Hierarchic refinement/abstraction

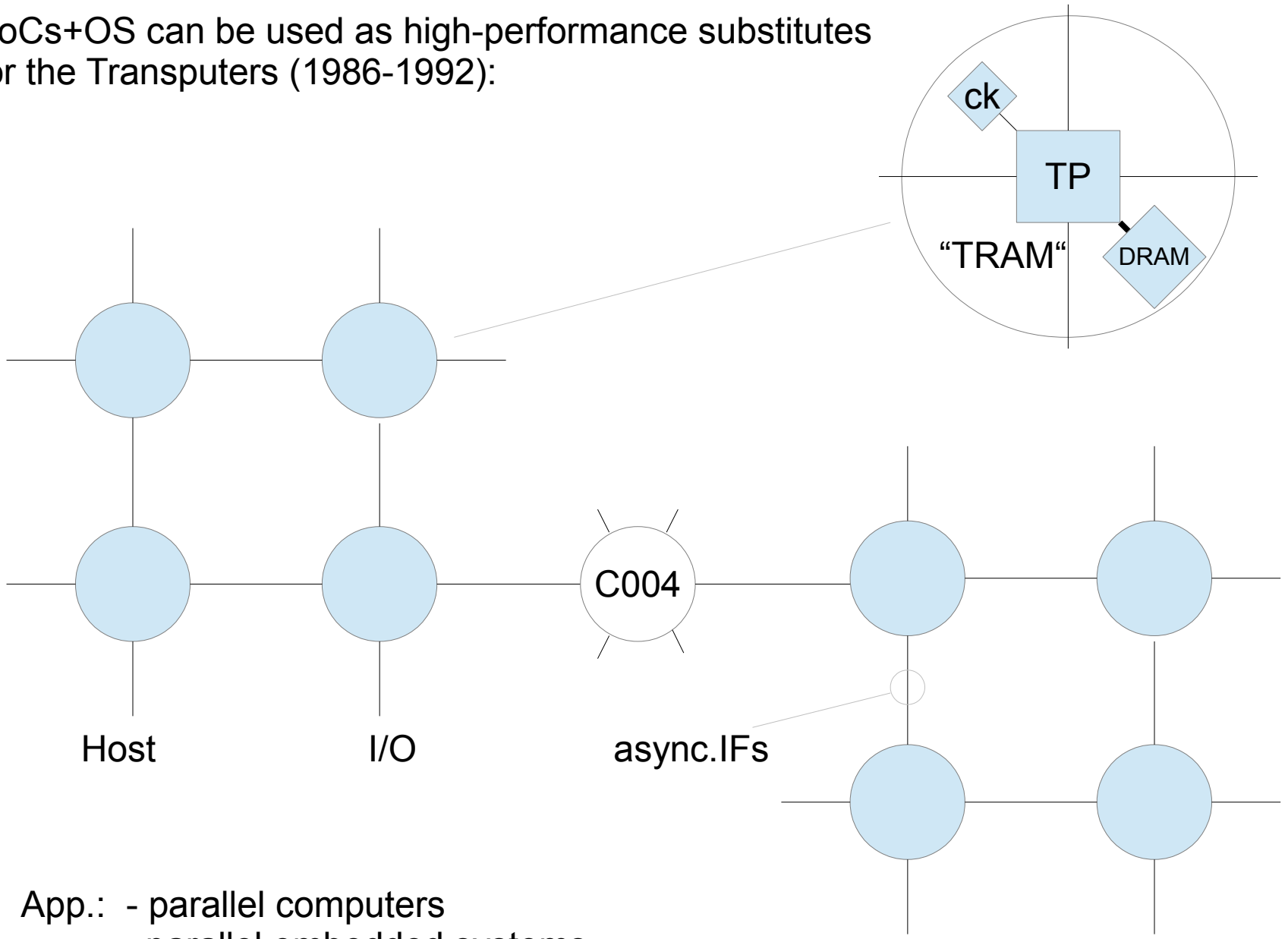


Example:

The scalable architecture of SoC nodes. Components are the SoC node type with 8 interfaces and optionally some more types representing other processors or system components like buses and memory.

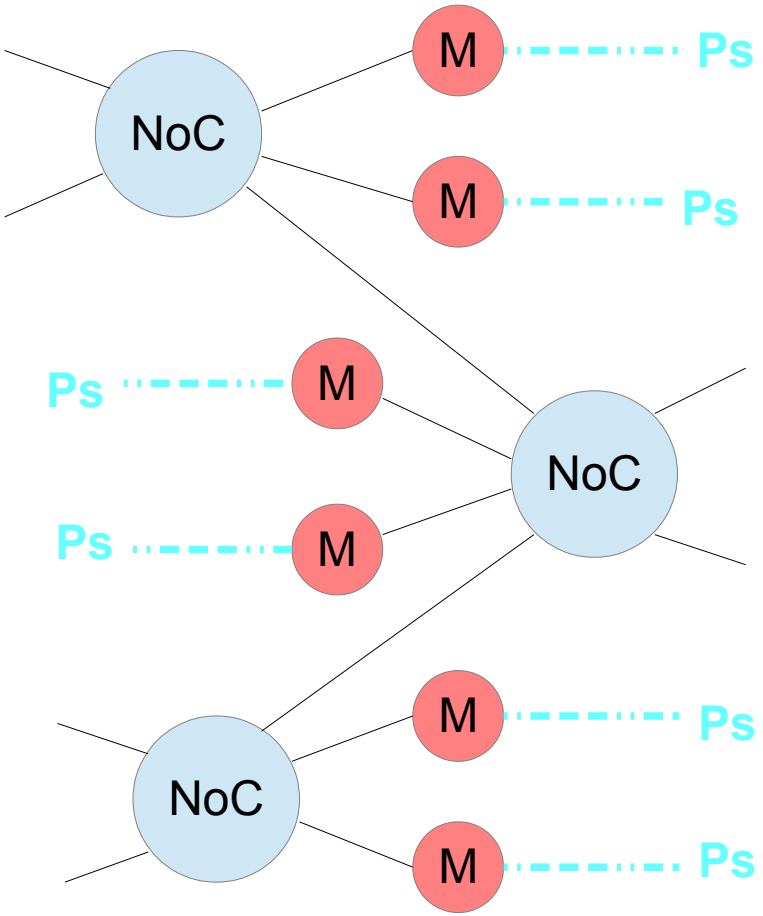
Refinement can be used to expand the sub networks of soft processors and the HW-OS.

SoCs+OS can be used as high-performance substitutes for the Transputers (1986-1992):



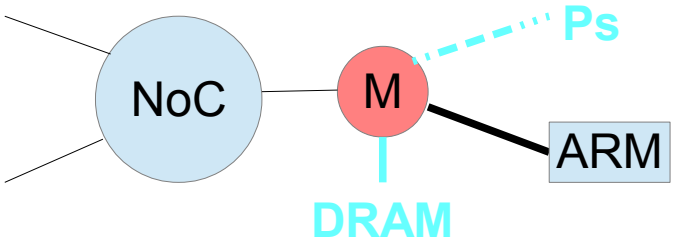
App.: - parallel computers
- parallel embedded systems

Partial function of the HW-OS for networking the memory nodes and ARM processors:



- the M nodes have a bridging function between the NoCs and Ps
- every DRAM-chip is controlled by an M node attached to an NoC, every M node has memory
- P's only access the M nodes they Are connected to on a single SoC
- M nodes store blocks of data, The network sends these as Messages to other M nodes
- processors send/receive blocks word-by-word or via DMA

Simplest case:



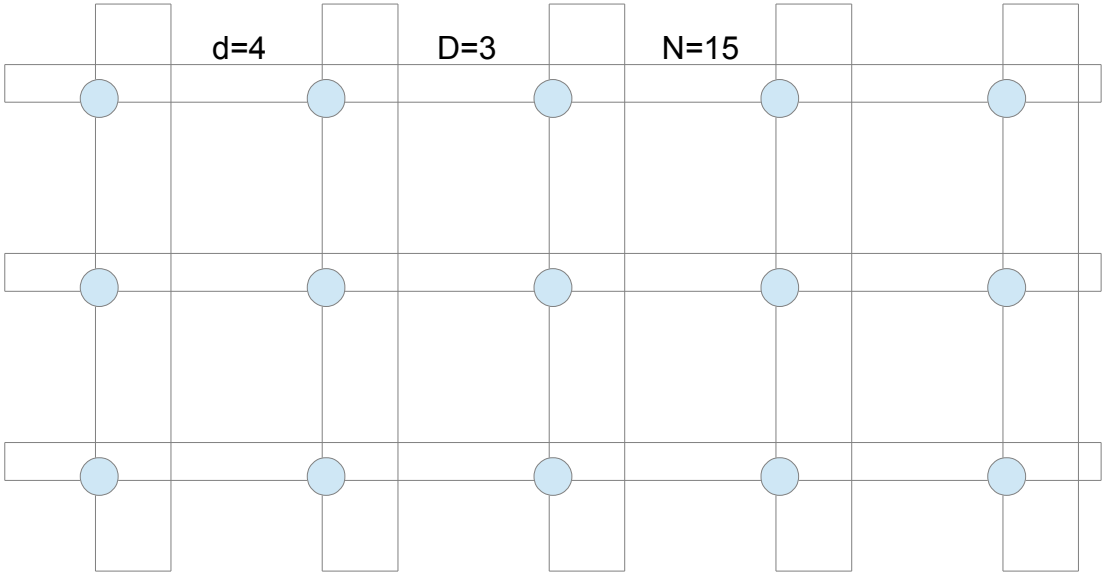
Component (processor) networks with internal links only 'are' finite graphs (V,E):

V node set (processors), $E \subset V \times V$ edge set (directional links)

Assume connectivity and symmetry, $(x,x) \notin E$, $(x,y) \in E \Leftrightarrow (y,x) \in E$ (bidirektional links, 'undirected graph')

Some metrics:

N	-	the numbers of elements of V	(processors)
d ('degree')	-	maximum number of neighbors over all nodes	(Schnittstellen)
h(x,y) for $x,y \in V$	-	minimum of the path lengths from x to y	(routing-Kosten)
D ('diameter')	-	maximum of the h(x,y) over all x,y	(max. routing costs)



Famous problem:
find graphs with largest N for given (d,D)

Symmetries of (V,E):
$f:V \to V$ bijektive, $(x,y) \in E \Leftrightarrow (f(x),f(y)) \in E\}$
S(V,E) group of all symmetries of (V,E)
$=\{\pi \in S(V) |\ \pi A = A\pi\ \}$ f. A = adjacency matrix

How many processors can there be for given d, D>1 in a network?

$$N \leq 1 + d + d(d-1) + .. + d(d-1)^{D-1} \qquad \text{(Moore bound)}$$

(V,E) is a 'Moore graph', if $N = 1 + d + d(d-1) + .. + d(d-1)^{D-1}$.

Satz (H-S): For D=2 the only Moore graphs up to isomorphy are

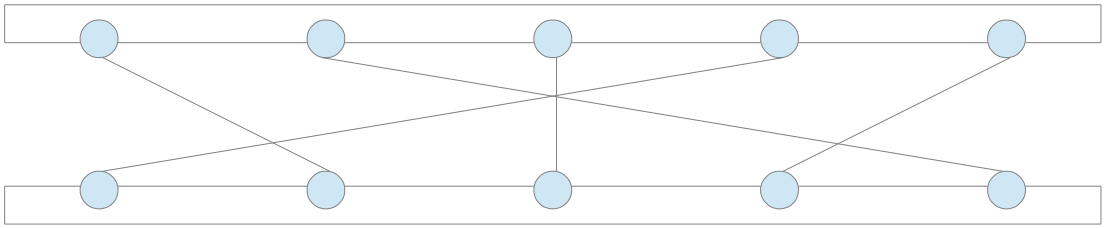
the cycel graph C5 with	d=2	and	N = 1+2+2*1 = 5
the Petersen graph with	d=3	and	N = 1+3+3*2 = 10
the HS-Graph (s.u.) with	d=7	and	N = 1+7+7*6 = 50

and perhaps (!)

one other Graph with	d=57	and	N = 1+57+57*56 = 3250.

Petersen-Graph:

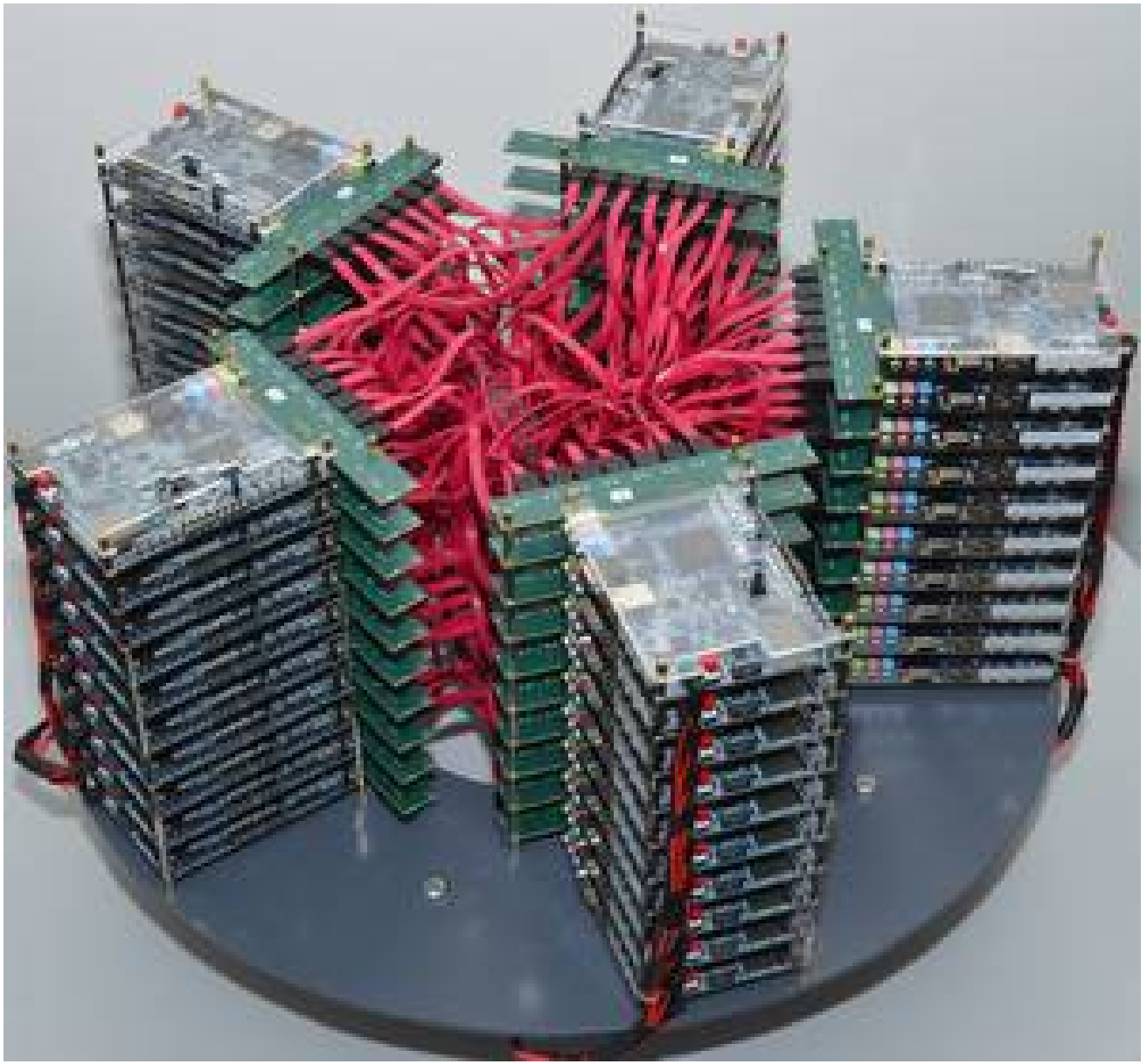
- ist not a Cayley graph
- has no Hamilton cykle



d=3, D=2

D=1: fully connected graphs, d=N-1
D>2: cycel graphs, d=2, N=2D+1

The ER4 experimental computer at the TUHH



- a general purpose parallel computer for research/education easy partitioning, IO extensions, rewiring

- 50 nodes linked via SATA cables

- nodes contain

  2 ARM cores
  FPGA
  2GB DRAM
  16GB flash

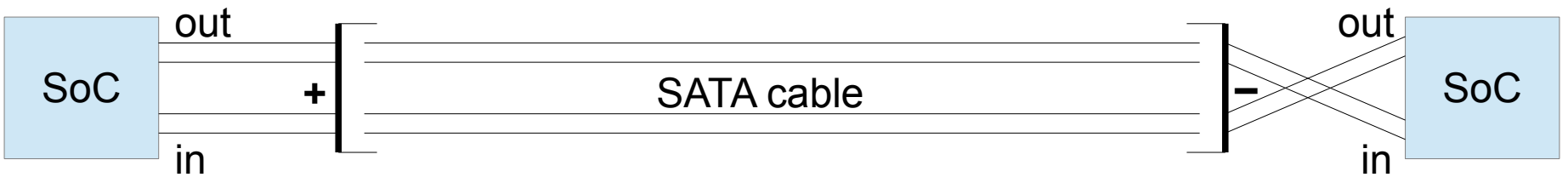
  small display, audio/vga

- host PC or terminals connect to any node

- about 800 soft processors plus 100 ARM cores

- 10 node columns are Petersen graphs

Funny detail: want to use standard SATA cables for the links between SoC nodes



- problem: SATA cables connect one-to-one, yet need IO crossing between transceivers

- solution: provide both '+' connectors on some transceivers, '-' connectors otherwise, specifically, 4 '+' and 4 '-' connectors for 8 transceivers on each SoC node

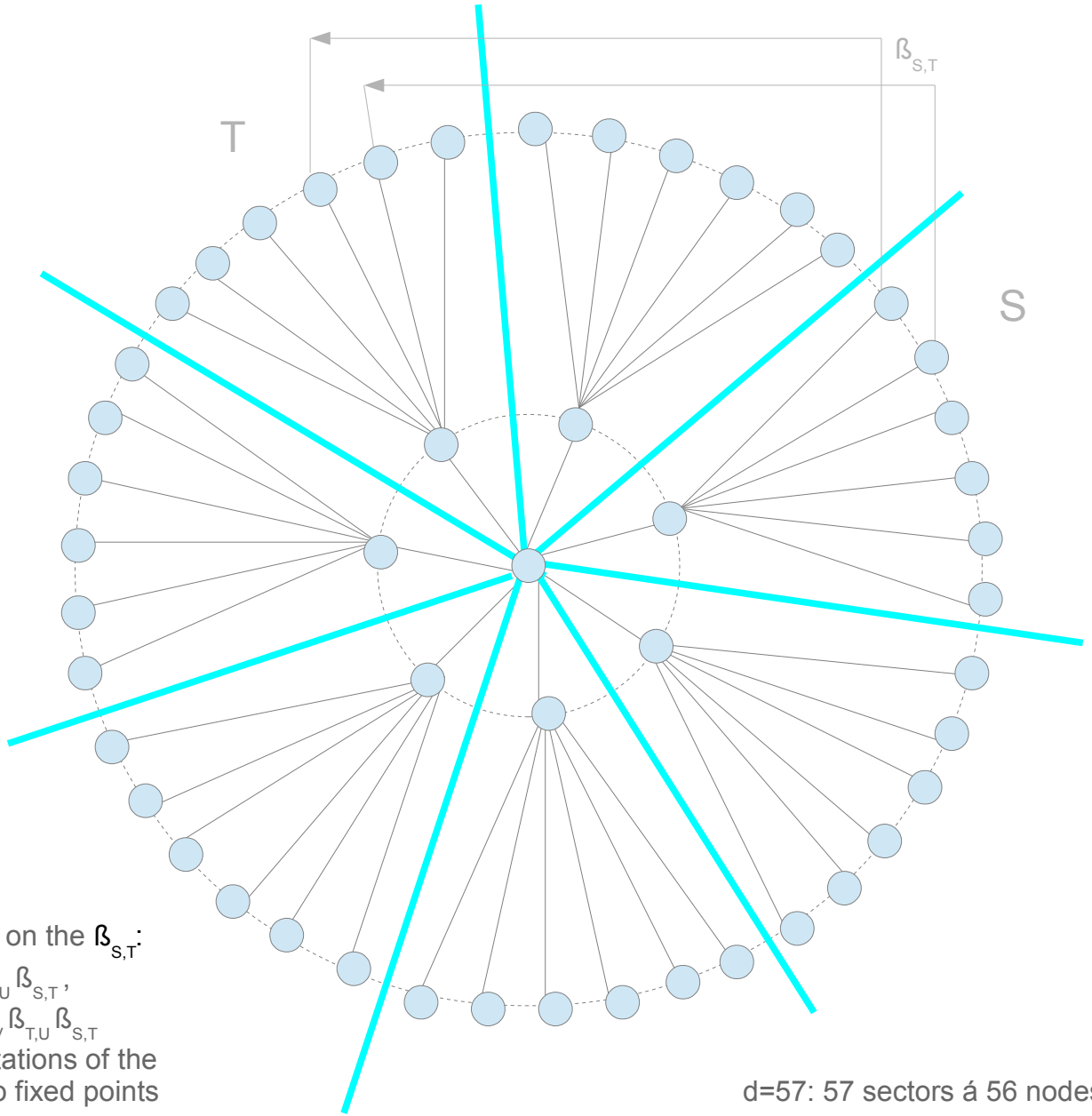
  interconnection rule: cables allowed between '+' and '-' connectors only

Can every interconnection graph of degree up to 8 be wired up this way ??

Lemma: Yes.

Proof: Wire up one by one longest paths of still unwired links.

The Hoffman-Singleton 50-node Moore graph (HS-Graph) - as a tree from a starting node:



1) The edges not shown are between the outer leaf nodes only (at d=2)

2) There are no cycles of less tha 5 nodes

3) Leaf nodes in the same sector are not connected

4) A leaf node is not connected to more than one node in the same sector
→ Bijektions ß$_{S,T}$↔Link

5) Every leaf connects to one in every other sector

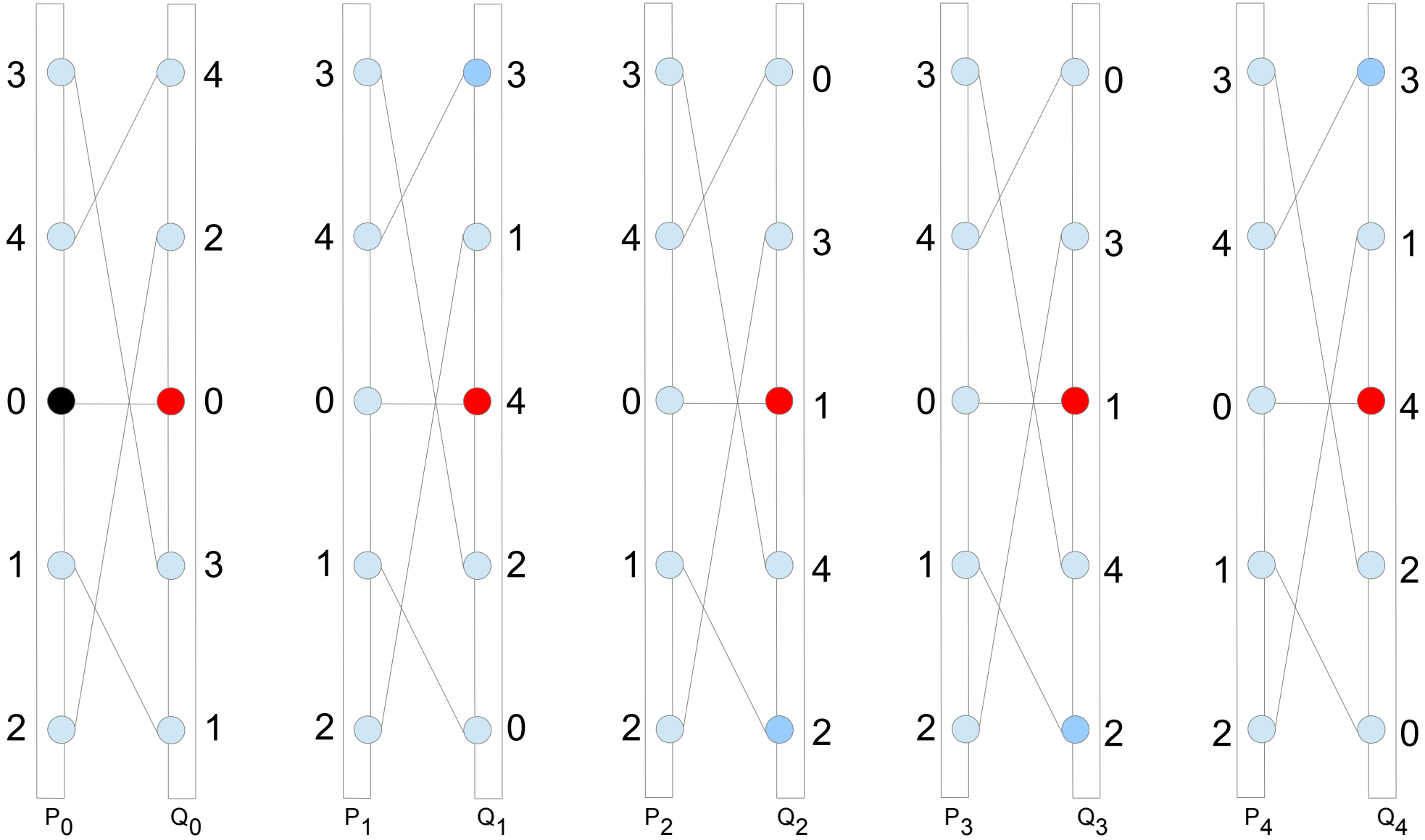
6) Every two not directly linked leaves connect via a lengh 2 path through a single intermediate sector

Conditions on the ß$_{S,T}$:
All ß$_{U,S}$ ß$_{T,U}$ ß$_{S,T}$,
ß$_{V,S}$ ß$_{U,V}$ ß$_{T,U}$ ß$_{S,T}$
are permutations of the sectors w/o fixed points

d=57: 57 sectors á 56 nodes á 56 edges .. 56! permutations.

Parametrizing the HS graph

5 Petersen blocks, 10 5-cycles



Also connect for m≠x $P_x(y) \leftrightarrow Q_m(c)$ with $y = m*x + c \mod(5)$

Parametrization has a geometric interpretation that generalizes

Petersen blocks are visible in this representation

Geometric Interpretation and Generalization (Hafner, McKay,Miller,Siran):

F number field of q elements, e.g. F={0,1,2,3,4}, w. operations +,* mod(5)

Use sub sets F_P and F_G of F with (q-1)/2 elements each and

$\{0\}\cup F_P\cup(F_P+F_P) = F$ $\{0\}\cup F_G\cup(F_G+F_G) = F$ $\{0\}\cup F_P\cup F_G=F$ $F_P= -F_P$ $F_G= -F_G$

'Plane' P of the q² 'Points' $(x,y) \in F\times F$

Let G be the set of 'Lines' parametrized by the $(m,c) \in F\times F$

Define a graph (V,E) by

$V = P\cup G$ N=2q² Elemente

E = the set of links (1) $(x,y) \leftrightarrow (m,c)$ if $y = mx+c$ (incidence relation)

d = q+(q-1)/2 (2) $(x,y) \leftrightarrow (x,y+f)$ if $f \in F_P$, d²+1 ~ 2¼q²

(3) $(m,c) \leftrightarrow (m,c+f)$ if $f \in F_G$.

Then the diameter of (V,G) is D=2.

Beispiele: q=5 d=7 N=50 .. the HS graph

q=9 d=13 N=162 (< 170) .. not Moore but nearly!

Sub graphs of HS

1260 * C5, 126 * 10 disjoint C5s

H-S ist Hamilton'sch

525 * Petersen

$v \in V \Rightarrow$ the sub graph of leaf nodes $\{x|h(x,v)=2\}$ is Hamiltonian

Symmetries of HS:

S(HS) = 252.000 = 50*5040 (Hafner), HS is not a Cayley-Graph

S(HS) is node transitive*, edge transitive, and transitive on the pairs (x,y) with h(x,y)=2

For a node $v \in V$ define $S_v = \{ s \in S(HS) \mid s(v)=v \}$ and $E_v = \{u \in V \mid h(v,u)=1\}$; $\# E_v = 7$.

S_v is a sub group of S(HS) of the size $\# S_v = 5040$.

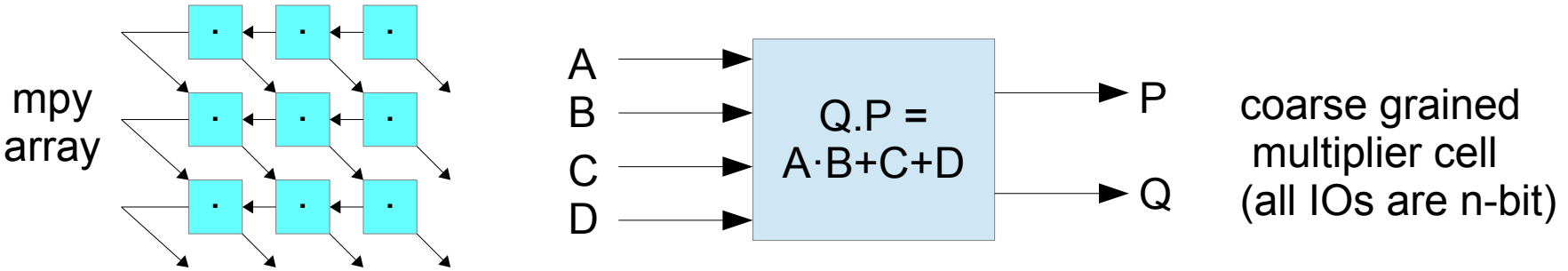
The restriction mapping $S_v \rightarrow S(E_v)$ is surjektive, hence bijektive (5040=7!)
A symmetry fixing a point and all of its neighbors must be the identity.

* can talk from host PC to every ER4 node

Hypothetic FPGA

The HW-OS uses up a significant amount of resources of the SoC in the ER4 (about 25%). As it supplies standard functions usable by almost every application it would make sense to hardwire it (including the soft controllers and the NoC with the routing function. This would simplify the FPGA programming, set free FPGA area for the arithmetic circuits, and raise the performance of the OS functions. Large FPGA components could simply be build by housing several smaller ones internally linked by serial interfaces. Small components can use simpler clock nets.

If numeric applications remain focused the mix of fine grained bit level and course grained arithmetic building blocks might be changed to further reduce the overheads for configurability and to raise performance while maintaining the flexibility to use non-standard number codes at a better performance than standard processors deliver.



Coarser grained FPGA architectures have been discussed by several authors. The 'Migcop' architecture defined in a research project at the TUHH e.g. proposes an FPGA cell performing 4-bit multiply and add operations and combining into multipliers and adders of arbitrary sizes. It also proposes the integration of hard-wired controllers and some special support for pipelined reconfiguration.