UNIVERSITY OF
DERBY

Simulating Discrete-Event Systems on HPC: Sleptsov Net Case Study

Dmitry A. Zaitsev

https://www.derby.ac.uk/staff/dmitry-zaitsev

Discrete-Event System (DES)

- A discrete event system is a dynamic system with discrete states, the transitions of which are triggered by events; its state evolution depends entirely on the occurrence of asynchronous discrete events over time.
- Two key features of a discrete event system are: i) its dynamics is event driven as opposed to time driven, i.e. the behaviour of a discrete event system is governed only by occurrences of different types of events over time rather than by ticks of a clock; and ii) its state variables belong to a discrete (not necessarily finite) set.
- Deterministic and stochastic event-based models.
- Discrete state and discrete time (sequence of steps).
- Simulation: time driven vs event driven.

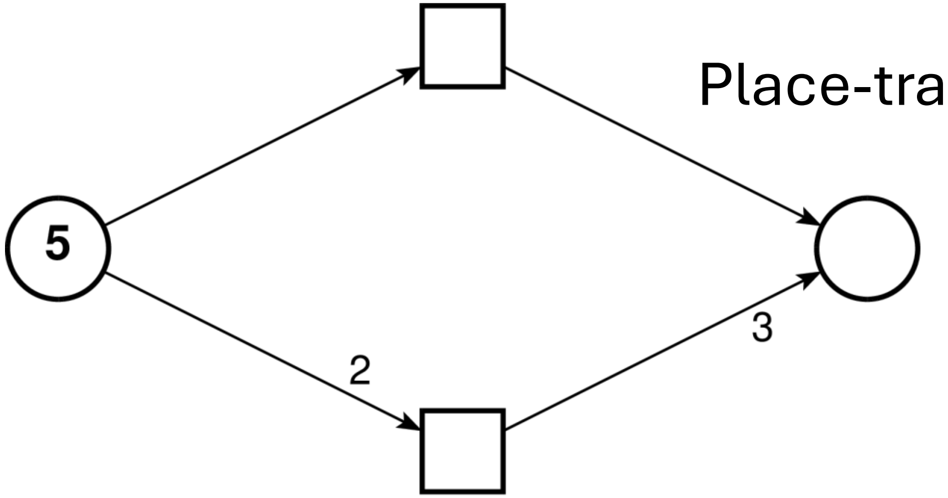
Examples of DESs

- **Formal DES systems:**
    - Finite automata
    - Cellular automata
    - Communicating Sequential Processes (CSP)
    - Labelled transition system
    - Place-transition net – Petri net
- **Informal DES systems:**
    - Manufacturing systems
    - Traffic control systems
    - Legal systems

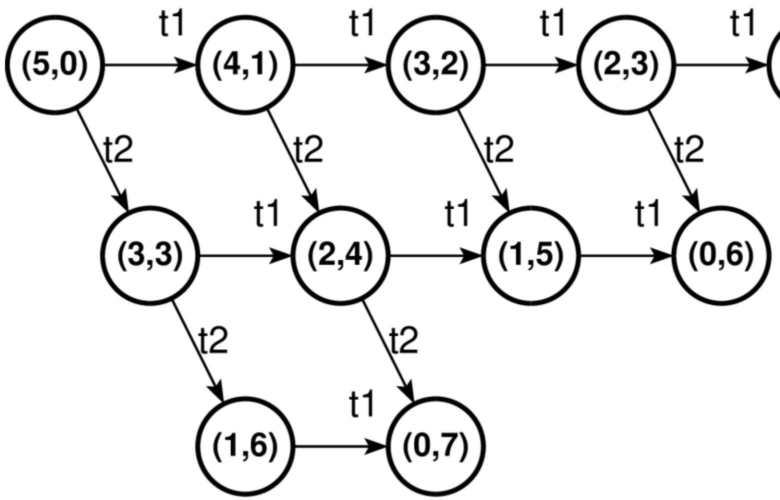
Sleptsov Net (SN)

- Bipartite directed graph – place-transition net
- Generalizes a Petri net for multiple firing a transition at a step
- Vertices: places (circles, ovals) and transitions (bars, rectangles)
- Tokens (dots, numbers) inside places – a marking of net
- Weight (multiplicity) of arcs – numbers inscribed on arcs
- Marking changes as a result of transition firing
- Turing-complete system
- https://dimazaitsev.github.io/snc.html

Sleptsov Net – Multiple Firing

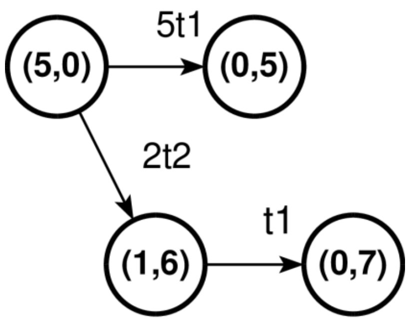


Place-transition net



Reachability graphs

Petri net



Sleptsov net

Inhibitor Priority Sleptsov Net

$N = (P, T, A, R, \mu_0)$

Places $p \in P$

Transitions $t \in T$

Arcs $A : P \times T \to \mathbb{Z}_{\geq 0} \cup \{-1\}, T \times P \to \mathbb{Z}_{\geq 0}$

Transition priority arcs $R : T \times T$, R^{+} is a strict partial order

Marking $\mu : P \to \mathbb{Z}_{\geq 0}$

- $A(p, t) = 0$, $A(t, p) = 0$ no arc;
- $A(p, t) > 0$, $A(t, p) > 0$ regular arc of specified multiplicity;
- $A(p, t) = -1$ inhibitor arc.

Transition Firing Rule

Firing multiplicity of arc:

$$c(p,t) = \begin{cases} \mu(p)/A(p,t), & A(p,t) > 0, \\ 0, & A(p,t) = -1 \wedge \mu(p) > 0, \\ \infty, & A(p,t) = -1 \wedge \mu(p) = 0. \end{cases}$$

Firing multiplicity of transition:

$$c(t) = \min_{A(p,t) \neq 0} (c(p,t)).$$

Firing transition choice:

$$(c(t') > 0) \wedge (\forall t \in T, t \neq t', c(t) > 0 : (t,t') \notin R^{+}).$$

Next marking:

$$\mu^{\tau+1}(p) = \mu^{\tau}(p) - c^{\tau}(t') \cdot A(p,t) + c^{\tau}(t') \cdot A(t,p), \;\; p \in P.$$

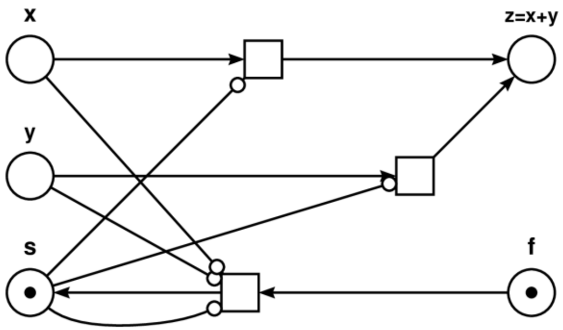
Simulation vs State Space

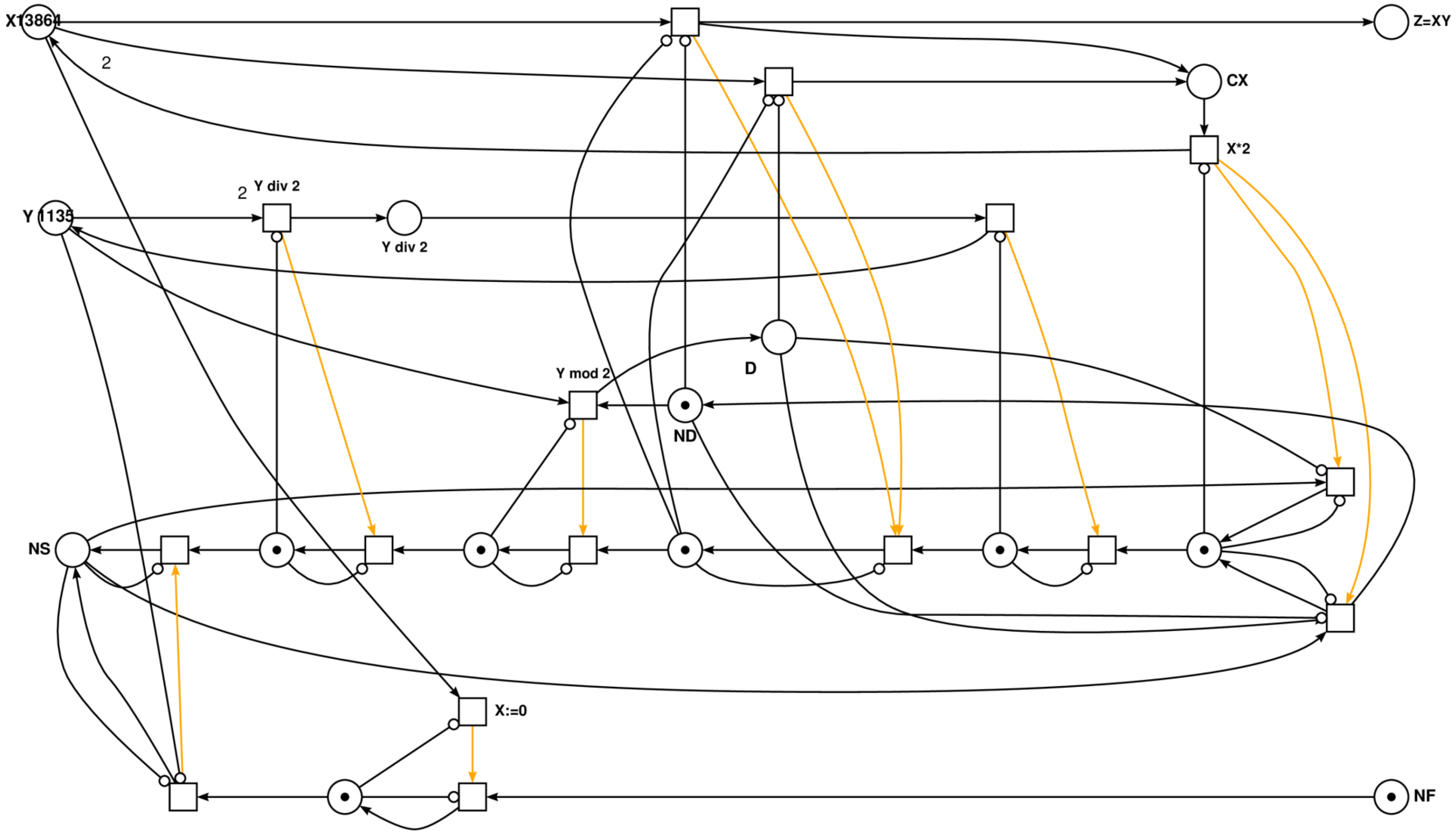
- State Space – specification of all valid states and all valid traces
- Simulation – a single, possibly very long, valid trace of firing transitions; just a valid path within State Space
- Simulation of a DES – a sequential process – a sequence of steps – a potential obstacle to implement on mass-parallel systems
- Composing State Space – fire all transitions, firable in the current state – employs mass-parallel system without bootlenecks
- An obstacle – possible infinite State Space – restricted finite constructs – graph of coverable markings for an SN/PN

Simulation as Computation

- Turing-Complete DES ***computes***
- Alternative traces are possible
- Composition rules of SN program preserve invariance of the obtained result with respect to a valid trace
- We can run a trace which is easier to follow – the first fireable transition instead of random choice at a step
- Sleptsov Net Computing (SNC): graphical SN language for concurrent programming; purity – no textual info save comments
- Refs for SNC: https://dimazaitsev.github.io/snc.html
- SN simulator = SN Virtual Machine (VM)

SNs for Addition and Multiplication





Recent Advances

- Dmitry A. Zaitsev, Tatiana R. Shmeleva, Alexander A. Kostikov, Computing and Communication Structure Design for Fast Mass-Parallel Numerical Solving PDE, Parallel Processing Letters, May 9, 2025.
- Zaitsev, D. A., Ajima, Y., Bartlett, J. F. C., & Kumar, A. (2025). 3D multicore CPU vs GPU on sparse patterns of Sleptsov net virtual machine. International Journal of Parallel, Emergent and Distributed Systems, 1–21. Published online: 16 Apr 2025.
- Zaitsev, D. A., Zhang, Z., Liu, D., & Shmeleva, T. R. (2025). Notation for mass parallel algorithms: computing Petri net state space on GPU case study. International Journal of Parallel, Emergent and Distributed Systems, 40(2), 101–115.
- R. Xu, S. Zhang, D. Liu and D. A. Zaitsev, "Sleptsov net based reliable embedded system design on microcontrollers and FPGAs," 2024 IEEE International Conference on Embedded Software and Systems (ICESS), Wuhan, China, 2024, pp. 1-8.

Developed tools

- ***SN VMs*** for HPC – multicore CPUs and GPUs – and embedded systems – microcontrollers
- ***SN state space generator*** for GPU
- ***SN compiler*** to FPGA for embedded systems
- ***Synchronous SN compiler*** to FPGA – integer approximation solver of PDE for embedded systems
- ***Linker of high-level SNs***
- Open access: https://github.com/dimazaitsev/SNCtools
- Using modelling system Tina, LAAS, France for drawing and verification of SN programs: https://projects.laas.fr/tina/index.php

How to run essentially sequential system on a mass-parallel computing device?

- **Mass-parallel implementation of a step**
  - It seems that using up to $|P| \times |T|$ threads, we can run a step in a constant time $O(1)$
  - Intrinsically sequential operations – binary operations (min) and sequential choice restrict us to the logarithmic complexity
- **Merging the source steps, firing a few fireable transitions at a step**
  - Works directly for conflict-free nets only
  - For spatial SNs on multidimensional lattices – speed-up is times the lattice size

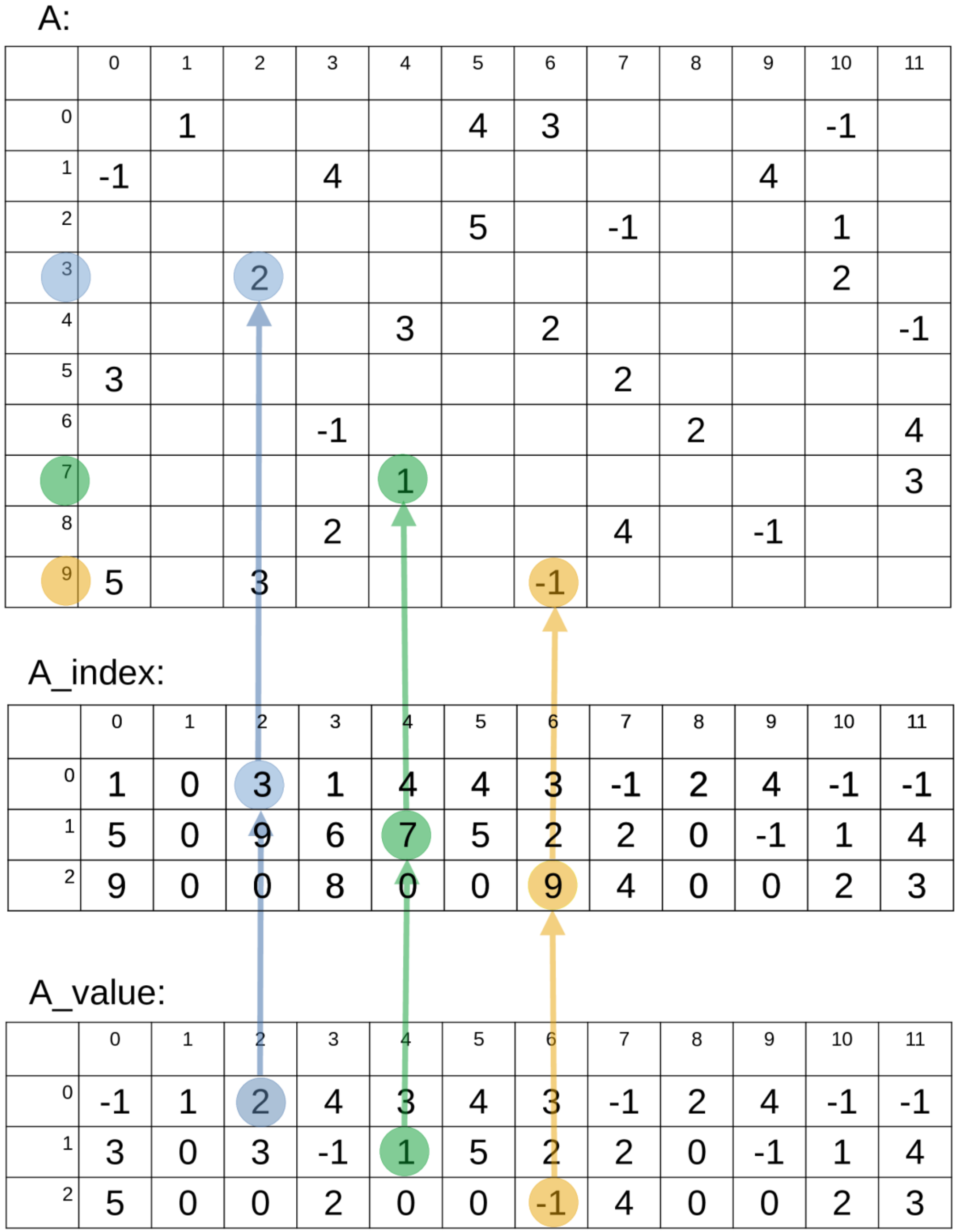
SN Step Algorithm

- Matrix representation of SN: B – $m \times n$ matrix of incoming arcs of transitions, D – $m \times n$ matrix of outgoing arcs of transitions, μ – m-vector of current marking, m – number of places, n – number of transitions
- **SN step algorithm:**
  - For an incoming (regular) arc of a transition: $y_{p,t} = \mu_p / b_{p,t}$
  - For a transition: $z_t = \min_p y_{p,t}$
  - <Filter vector $\bar{c}$ on priority lattice – for priority nets>
  - Choose a firable transition: $f, z_f > 0$
  - Fire transition f:

$$\mu_i^{k+1} = \mu_i^k - z_f \cdot b_{i,j} + z_f \cdot d_{i,j}, b_{i,j} > 0 \ \wedge d_{i,j} > 0$$

Sparse Matrix with Condensed Columns (MCC)

- Convenient for GPU processing
- $m \times n$ matrix A is represented by a pair of $mm \times n$ matrices:
- A_index – nonzero element index in the source matrix
- A_value – nonzero element value
- mm – the number of nonzero elements over columns of A
- Comparably low overhead



A:

	0	1	2	3	4	5	6	7	8	9	10	11
0		1				4	3				-1	
1	-1			4						4		
2						5		-1			1	
3			2								2	
4					3		2					-1
5	3							2				
6				-1					2			4
7					1							3
8				2				4		-1		
9	5		3				-1					

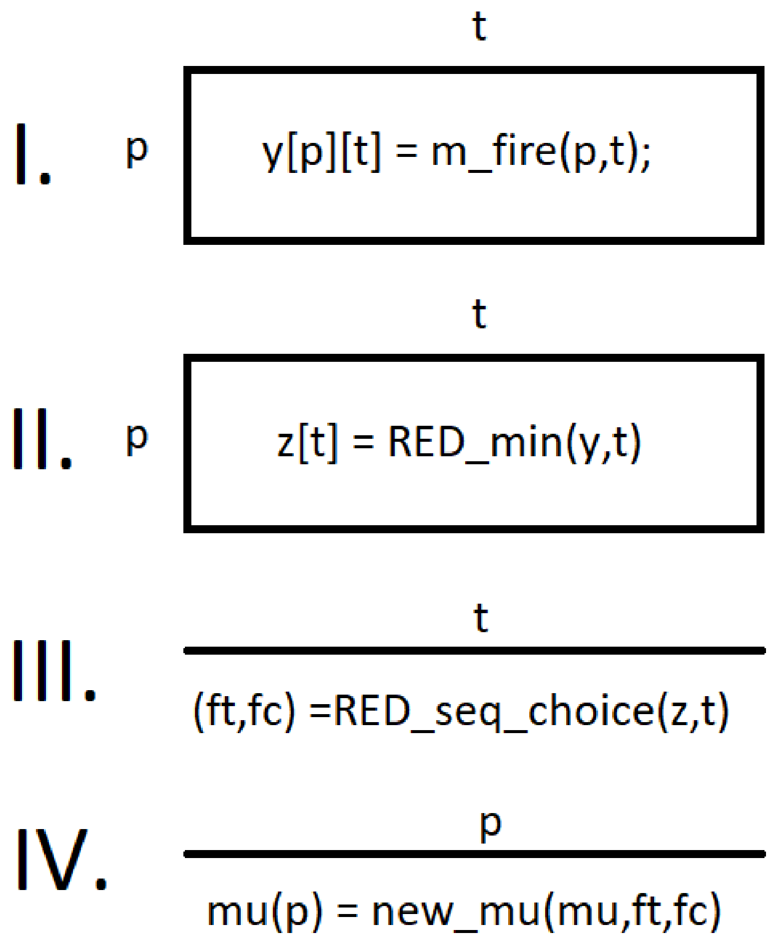
A_index:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	0	3	1	4	4	3	-1	2	4	-1	-1
1	5	0	9	6	7	5	2	2	0	-1	1	4
2	9	0	0	8	0	0	9	4	0	0	2	3

A_value:

	0	1	2	3	4	5	6	7	8	9	10	11
0	-1	1	2	4	3	4	3	-1	2	4	-1	-1
1	3	0	3	-1	1	5	2	2	0	-1	1	4
2	5	0	0	2	0	0	-1	4	0	0	2	3

Mass-Parallel Algorithm Notation



I. t p y[p][t] = m_fire(p,t);

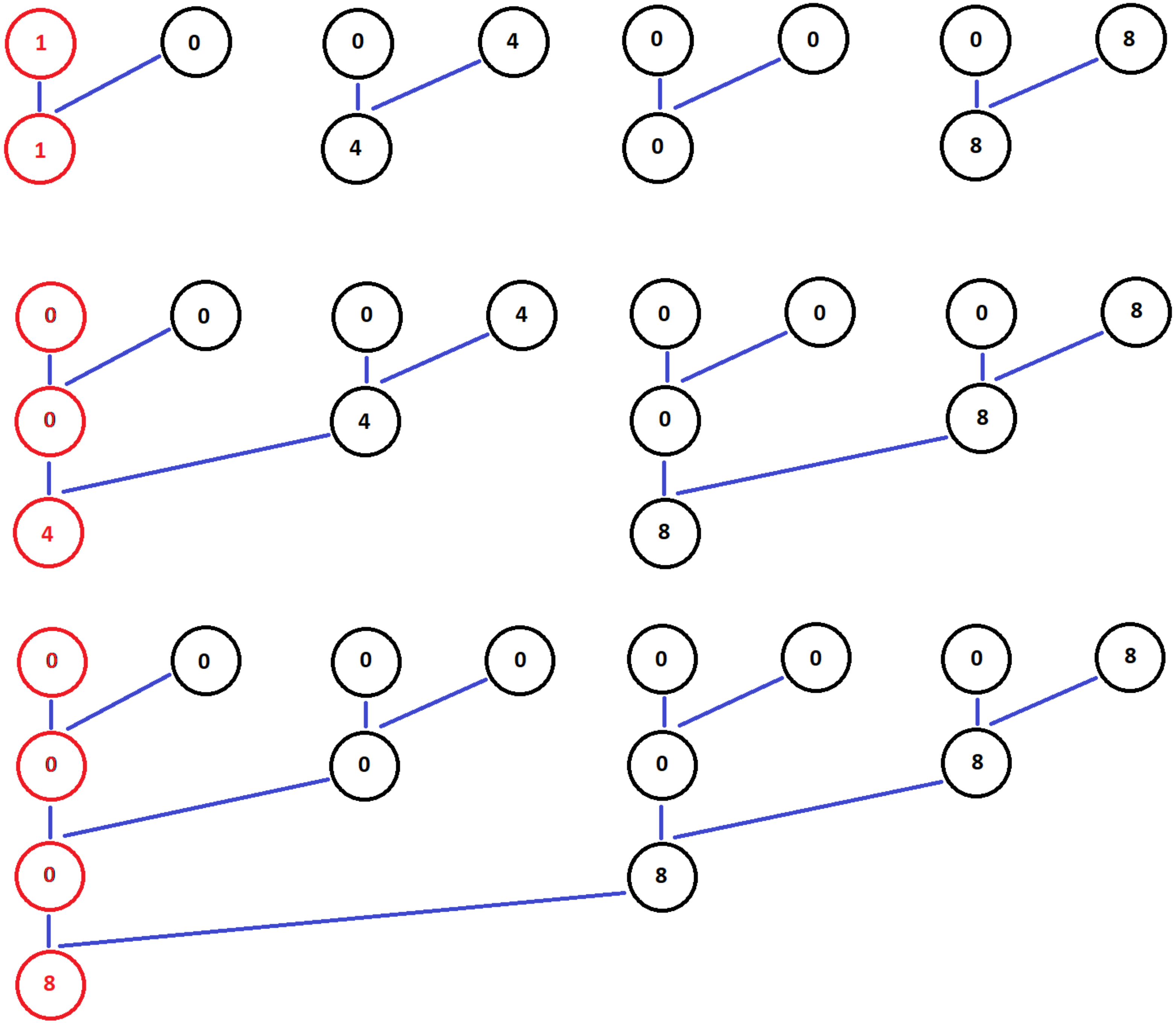
II. t p z[t] = RED_min(y,t)

III. t (ft,fc) =RED_seq_choice(z,t)

IV. p mu(p) = new_mu(mu,ft,fc)

- Priority filtering is omitted at a step
- Transitions are preliminarily reordered in the descending order of priorities
- At a step, the first firable transition choice

Reduction of Sequential Choice



Best case

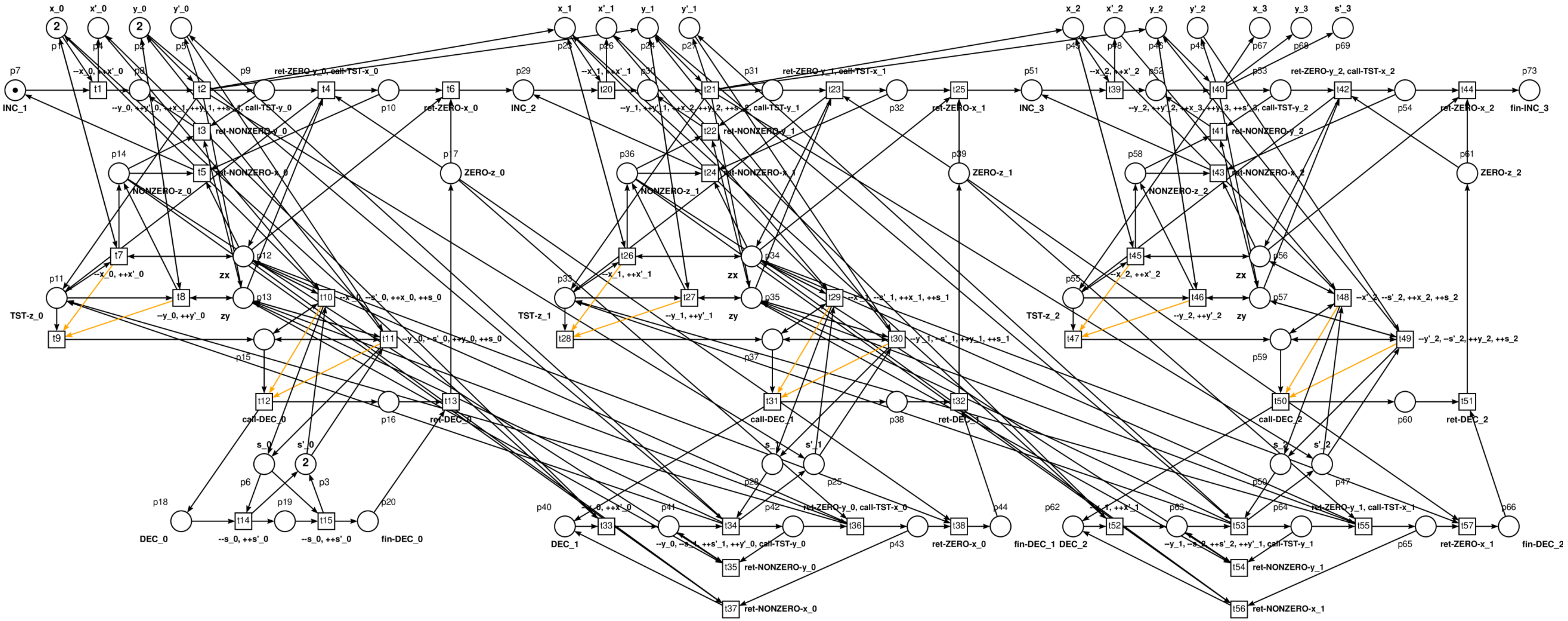
Average case

Worst case

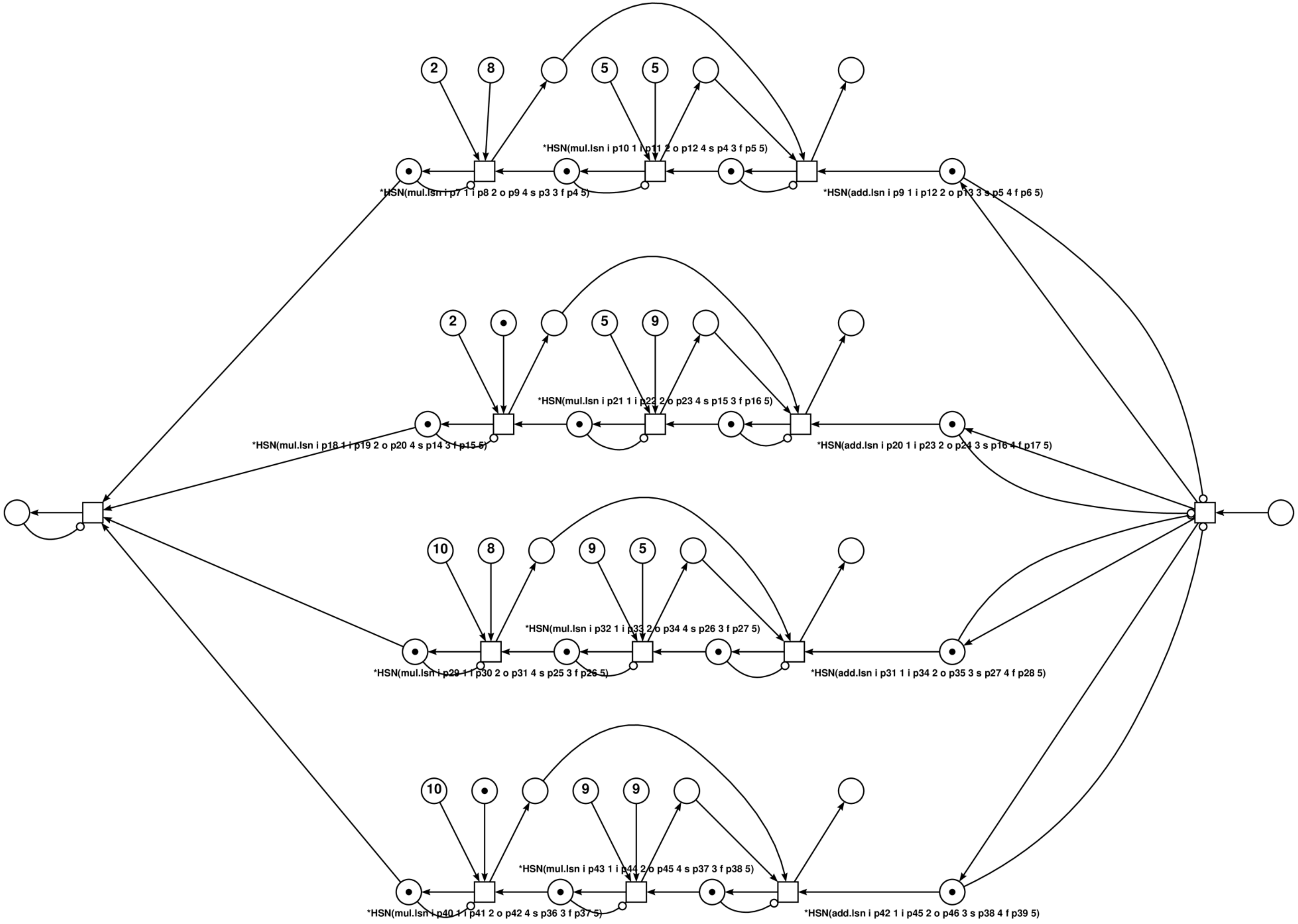
DES simulation time complexity depends upon who runs DES

- A conventional (sequential) algorithm: $O(k \cdot m \cdot n)$ improved to $O(k \cdot mm \cdot n)$ for sparse matrices
- A Universal Sleptsov Net: polynomial in k
- A mass-parallel algorithm: $O(\mathrm{k} \cdot (\log mm + \log n))$
- A zero universal SN: SN runs itself
- SN is implemented in hardware – compiled to Verilog and run on FPGA: $O(k \cdot (mm + n))$ improved to $O(\mathrm{k} \cdot (\log mm + \log n))$

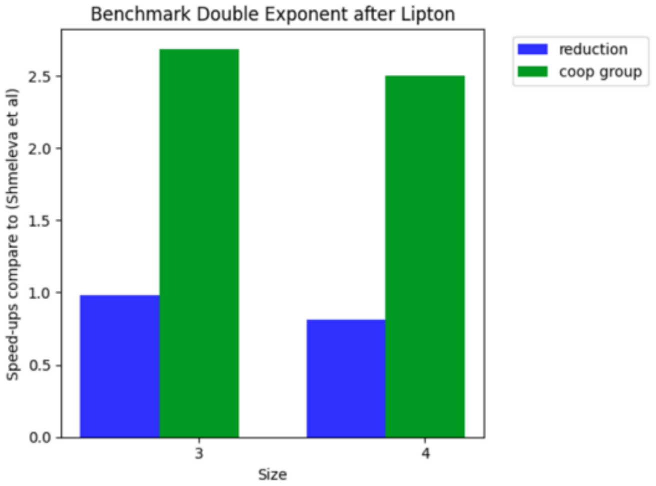
Benchmark SNs: Double exponent after Lipton



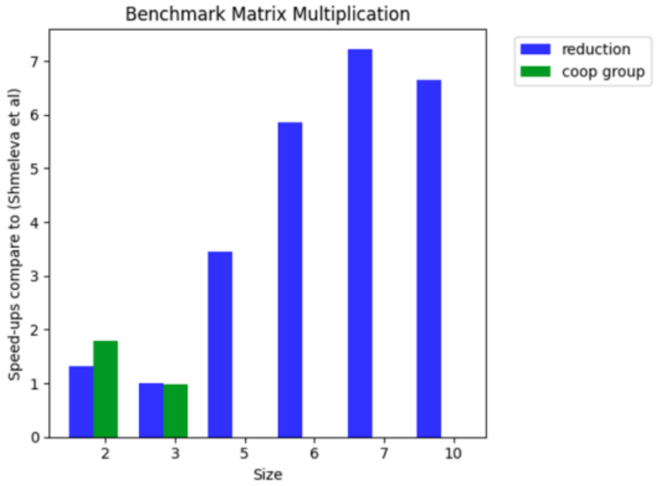
Benchmark SNs: Matrix Multiplication



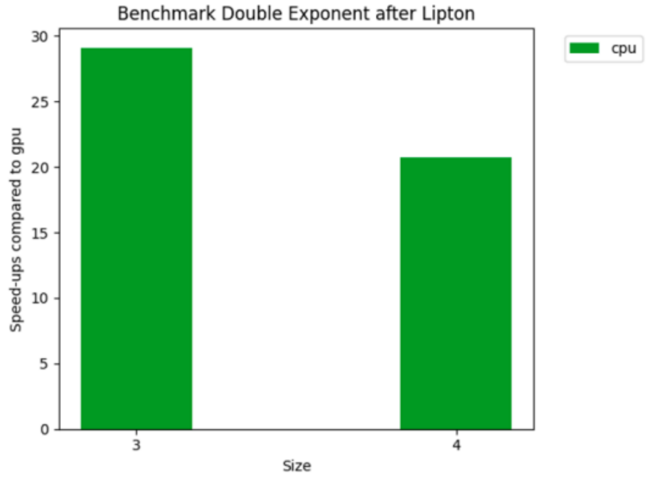
Benchmarks



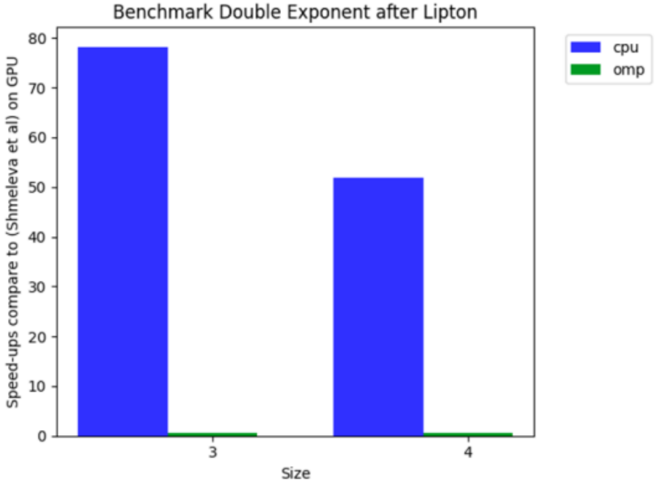
(a) double exponent on GPU;



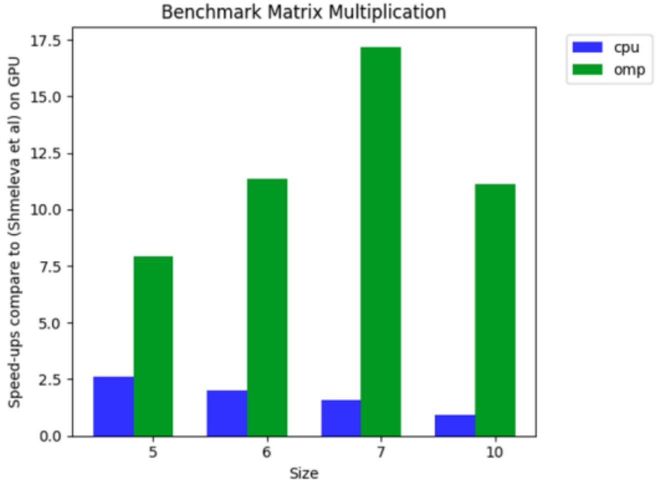
(b) matrix multiplication on GPU.



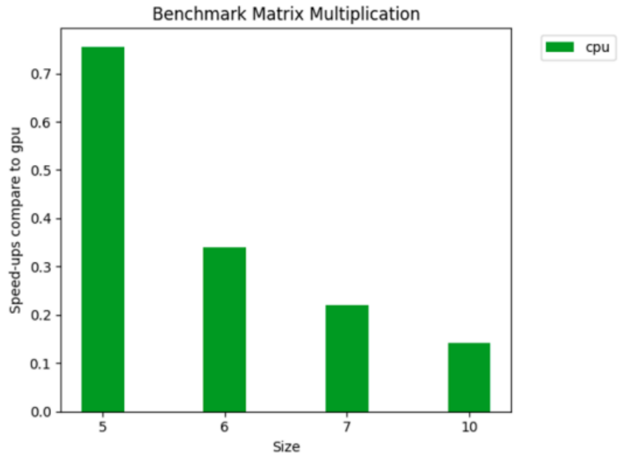
(e) double exponent on CPU vs GPU;



(c) double exponent on CPU;



(d) matrix multiplication on CPU.



(f) matrix multiplication on CPU vs GPU.

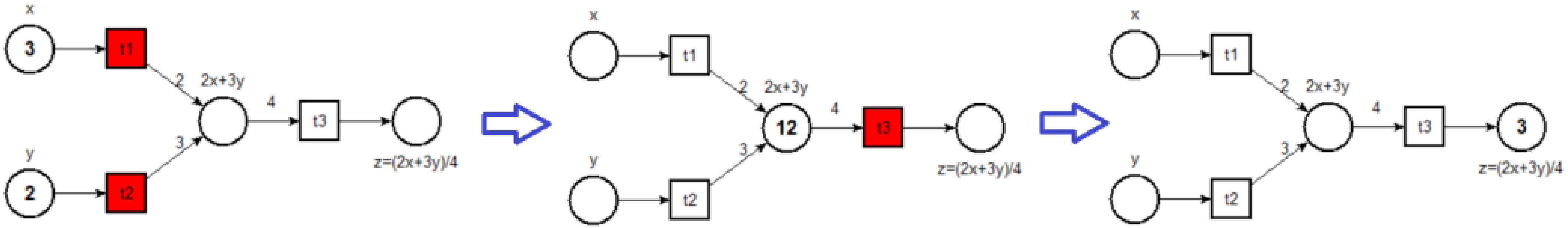
NVIDIA GPU is a clumsy device to handle

- Claimed 6D lattice of threads is considerably restricted actually
- Organized as a 3D grid of 3D blocks of threads
- Maximal grid size does not mean parallel execution of all threads; they are processed sequentially by available Streaming Multiprocessors
- Block size is limited (1024 in a sum of 3D)
- Performance considerably depends on using the block warp structure (a bunch of 32 threads)
- In general, synchronization is provided within a block
- Global synchronization over the grid with Cooperative Groups has reduced scalability limited by the actual number of Streaming Multiprocessors
- Modern and future multicore CPUs represent a good competitor

Is the maximal firing strategy a remedy?

- ***The maximal firing strategy*** = synchronous net – fire the maximal subset of fireable transitions at a step (Salwicki, Burkhard, Kotov)
- Looks like a good deal of speed-up: one step instead of many steps
- Works perfectly for (structurally) conflict-free nets
- Applied for verification and performance evaluation of nonnegative integer approximation of PDE running on FPGA using finite-difference methods
- In the general case, the step timed complexity becomes exponential in the number of fireable transitions

Sync SN Transition Firing Rule

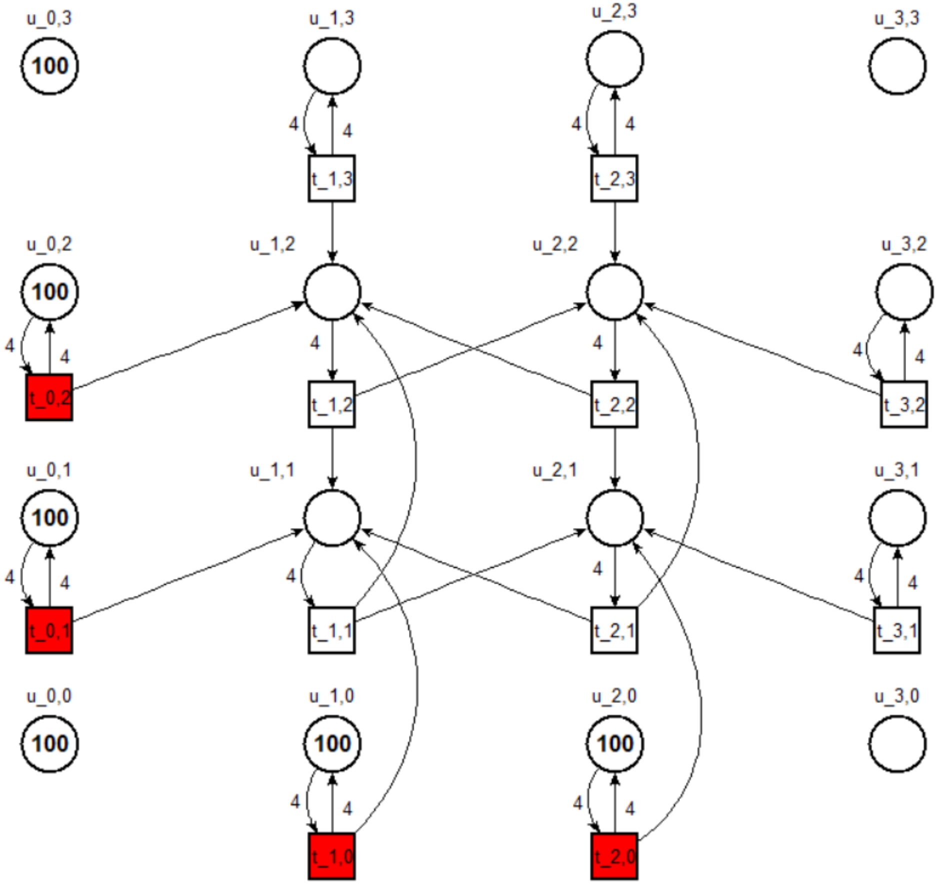


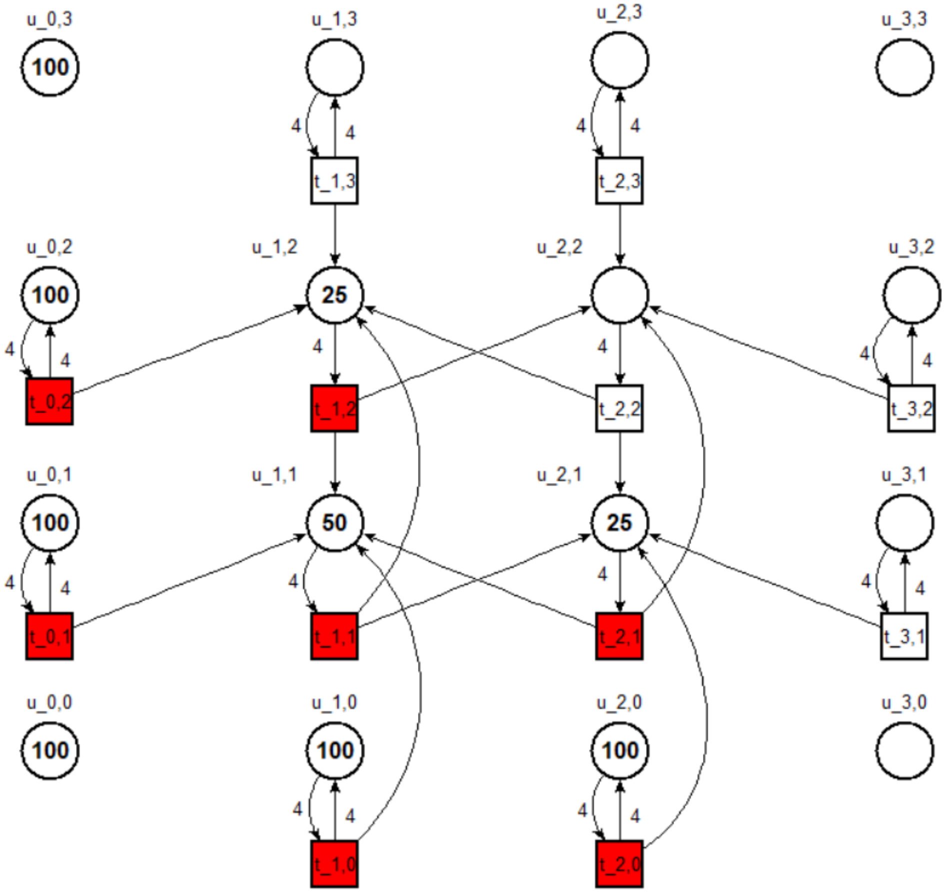
$$C(p,t) = \mu(p)/A(p,t)$$ **Firing multiplicity of arc**

$$C(t) = \min_{A(p,t)>0} C(p,t)$$ **Firing multiplicity of transition**

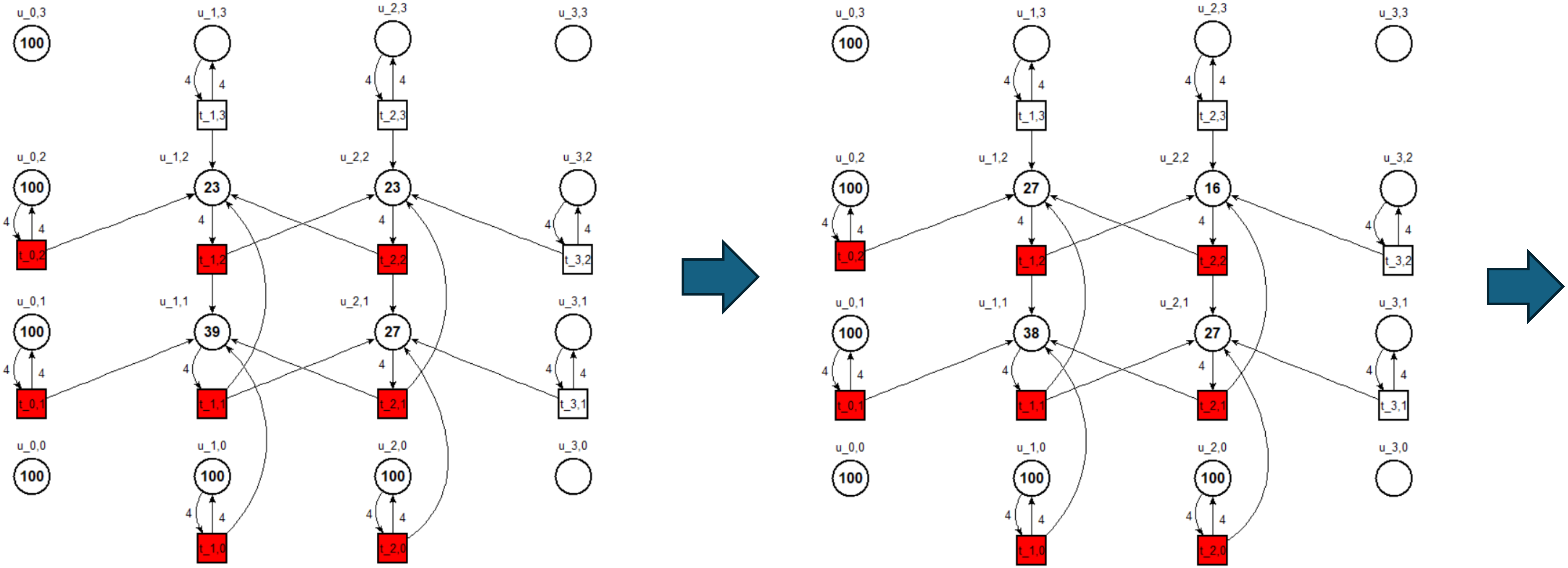
$$\mu(p)^{k+1} = \mu(p)^k - \sum_{A(p,t)>0} C(t) \cdot A(p,t) + \sum_{A(t,p)>0} C(t) \cdot A(p,t)$$ **Next marking**

Sync SN to Solve Laplace Equation, #1





Sync SN to Solve Laplace Equation, #2



Conclusions

- Speed-up DES simulation on mass-parallel devices, e.g. GPU
- ***Sleptsov Net Computing***: simulation as computation
- **Speed-up a step using:**
  - Ad-hoc sparse data format – matrix with condensed columns (MCC)
  - Reduction of sequential choice
  - Global synchronization with GPU Cooperative Groups (limited)
- **Speed-up through steps:**
  - The maximal firing strategy (Salwicky, Burkhard, Kotov) – fire a few fireable transitions at a step
  - Speed-up for conflict-free nets, numerical solving PDE on FPGA
  - Exponential complexity of transition choice in general case