

Sleptsov Net Computing

Dmitry Zaitsev

<https://dima.zaitsev.github.io>



Most powerful computers: <http://Top500.org>

System	Year	Vendor	Cores	Rmax (PFlop/s)	Peak (PFlop/s)	Efficiency (HPCG)
<u>Supercomputer Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11</u>	2021	HPE	8,730,112	1,102.00	1,685.65	0.8 %
<u>Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D</u>	2020	Fujitsu	7,630,848	442.01	537.21	3.0 %

Jack Dongarra Turing Award Talk

Top 500 Supercomputers in the World as of November 2022								
Rank	Site	Computer	Country	Cores	HPLMax [Pflop/s]	Top 500 Rank	HPCG [Pflop/s]	% of Peak
1	RIKEN Center for Computational Science	Fugaku, Fujitsu A64FX 48C 2.2 GHz, Tofu D, Fujitsu	Japan	7,630,848	422	2	16.0	3.0%
2	DOE / SC / ORNL	Frontier, HPE Cray EX235a, AMD 3rd EPYC 64C, 2 GHz, AMD Instinct MI250X, Slingshot 10	USA	8,730,112	1,102	1	14.1	0.8%
3	EuroHPC / CSC	LUMI, HPE Cray EX235a, AMD Zen 3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11	Finland	2,174,976	304	3	3.41	0.8%
4	DOE / SC / LBNL	Summit, AC922, IBM POWER9 22C 3.7GHz, Dual-rail Mellanox FDR, NVIDIA Volta V100, IBM	USA	2,414,592	149	5	2.93	! 5%
5	EuroHPC/CINECA	Leonardo, BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 40 GB, Quad-rail NVIDIA HDR100 Infiniband	Italy	1,463,616	175	4	2.57	1.0%
6	DOE / SC / LBNL	Perlmutter, HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10	USA	761,856	70.9	8	1.91	2.0%
7	DOE/NNSA/LLNL	Sierra, S922LC, IBM POWER9 20C 3.1 GHz, Mellanox EDR, NVIDIA Volta V100, IBM	USA	1,572,480	94.6	6	1.80	1.4%
8	NVIDIA	Selene, DGX SuperPOD, AMD EPYC 7742 64C 2.25 GHz, Mellanox HDR, NVIDIA Ampere A100	USA	555,520	63.5	9	1.62	2.0%
9	Forschungszentrum Juelich (FZJ)	JUWELS Booster Module, Bull Sequana XH2000, AMD EPYC 7402 24C 2.8GHz, Mellanox HDR InfiniBand, NVIDIA Ampere A100, Atos	Germany	449,280	44.1	12	1.28	1.8%
10	Saudi Aramco	Dahrmm-7, Cray CS-Storm, Xeon Gold 6242, NVIDIA	Saudi Arabia	22,4	20	0.88	1.64	Activate Windows Go to Settings to activate Windows.

are getting .8% of the theoretical peak performance, .8%.

Frontier



Architecture and programming tools

- Multicore processor node – shared memory, 32-64 cores – [OpenMP](#)
- Massively parallel computations – GPU – 6D structure of threads (3D grid, 3D block), ~10000 cores – [NVIDIA CUDA](#) or [OpenCL](#)
- Distributed nodes, fast (multidimensional) interconnect – [MPI](#)

Benchmarks of computer performance

- Since 1977, LINPACK – to solve a dense system of linear equations (LU factorization with partial pivoting)
- HPL (High Performance Linpack), 1991
- The High Performance Conjugate Gradients (HPCG) Benchmark, 2015 – Sparse matrix-vector multiplication and solvers

Think! 0.8% of efficiency

- LINPACK, for a half of century, shaped computer architecture to be inefficient
- Processor-memory bottleneck of Von-Neumann architecture
- Sequential programming languages
- Heterogeneous conglomerate of concurrent programming techniques: OpenMP, MPI, OpenCL (CUDA)

Dilemmas

- Why we need separate Processor and Memory?
- Write Programs or Draw Programs?
- Is heterogeneous framework inevitable?
- Correctness Proof or Testing for Concurrent Programs?

Approaching a uniform concept

- Textual programming
- Graphs loaded by textual language
- **Pure graphical programming - nothing save graphs**
- Sleptsov net – fast universal language of concurrent programming
- Massively parallel computations
- Fine granulation
- Computing memory implementation

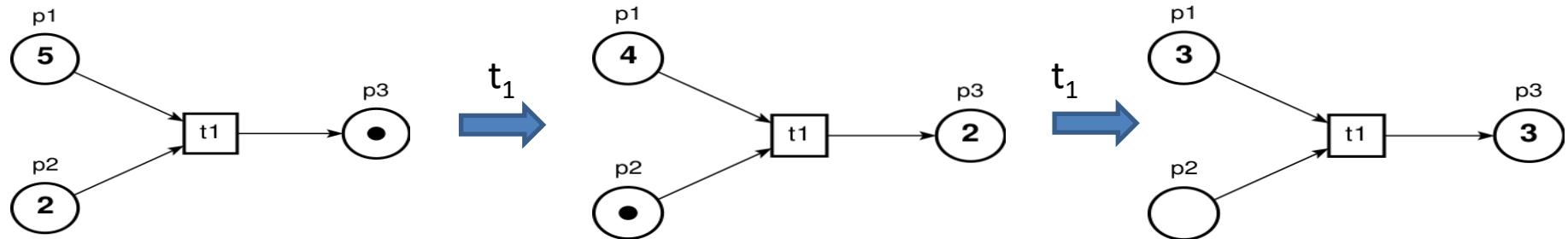
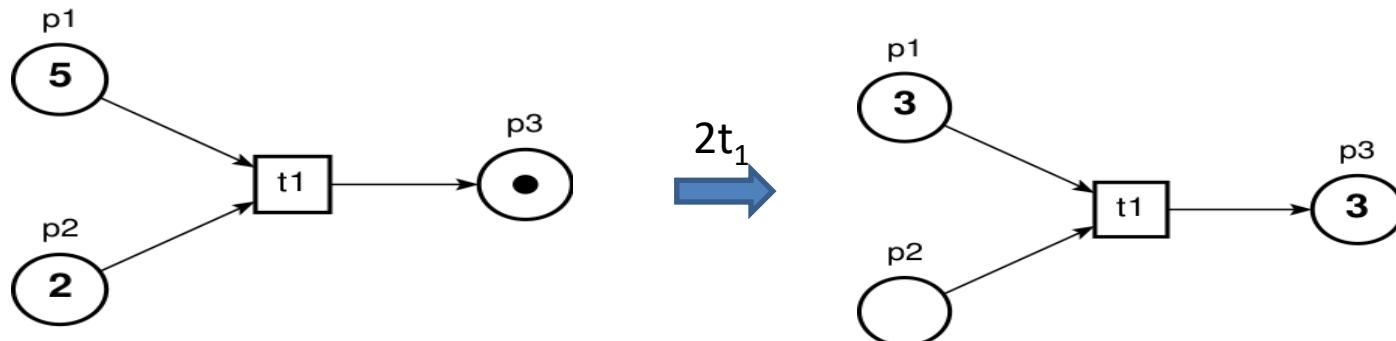
Place-transition nets

- Bipartite directed graph in CS:
- [S.Gill](#), 1958 – parallel program schemata
- [C.Petri](#), 1962 – added dynamical elements – tokens
- [T. Agerwala](#), [M.Hack](#), 1974-1976 – Turing-complete extensions: inhibitor and priority nets

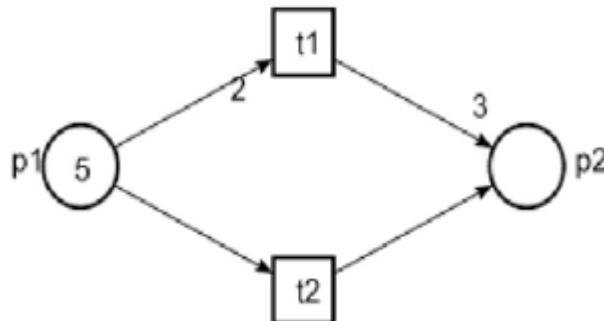
Transition firing strategy

- *Petri*
 - a single transition at a step
- *Salwicki*
 - the maximal firing strategy
- *Sleptsov*
 - the multiple firing strategy

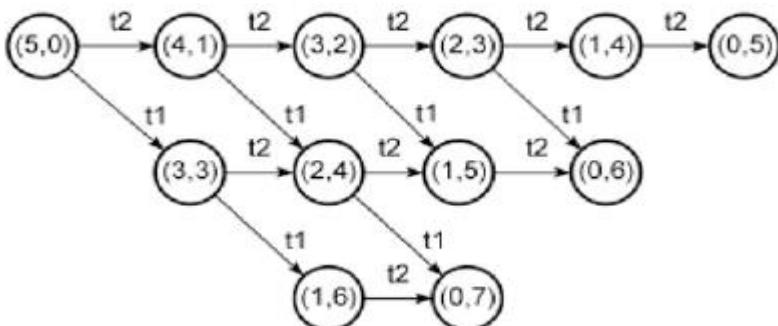
Sleptsov net vs Petri net



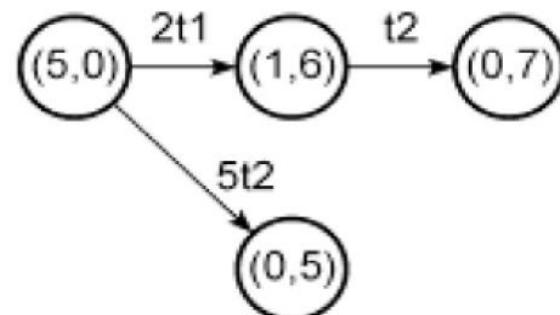
Sleptsov Net – Multiple Firing



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

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

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

- $A(p, t) = 0, A(t, p) = 0$ 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:

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

Sleptsov Nets Run Fast

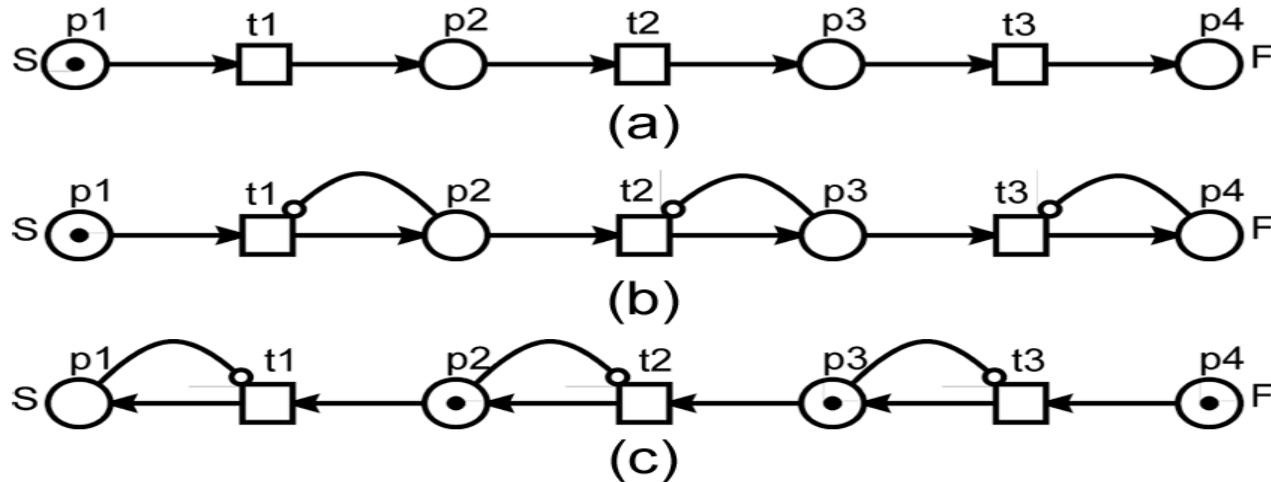
COMPARING TIME COMPLEXITIES OF OPERATIONS (LINEAR SCALE - NUMBER OF STEPS)

Operation	PN	SN
CLEAN	$x + 2$	2
MOVE	$x + 2$	2
COPY	$2 \cdot x + 3$	4
ADD	$x + y + 2$	3
SUB	$\max(x, y) + 3$	3
GT	$\max(x, y) + 3$	4
MUL	$y \cdot (2 \cdot x + 3) + x + 3$	$11 \cdot \log_2 y + 3$
DIV	$(x / y) \cdot (2 \cdot y + 2) + (x \% y) + y + 4$	$39 \cdot (\log_2 x - \log_2 y) + 19$

Concurrent programming in SNs

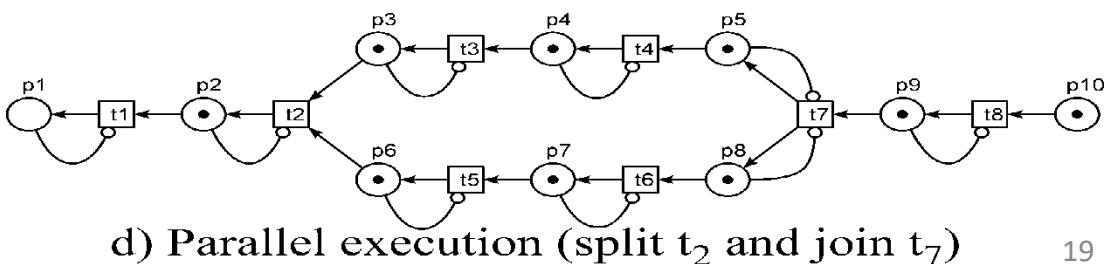
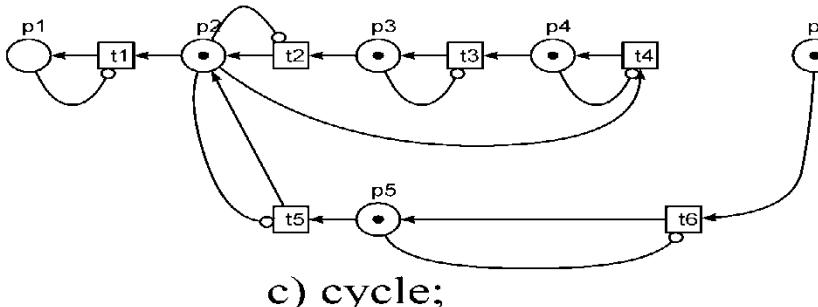
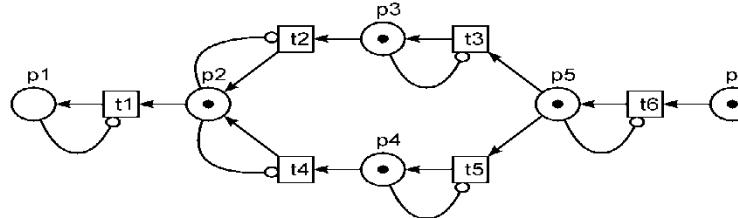
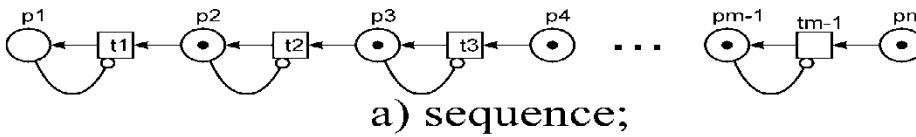
- Control Flow Approach – an explicit control flow starting-finishing transformation of data
- Data Flow Approach – start computations on readiness of data
- Hybrid Approach
- Unstructured approach

Peculiarities of programming in SNs

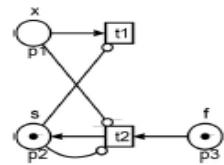


- Reversed control flow (c)
- Using inhibitor arcs to control transitions' firing
- Infinite firing multiplicity on a valid inhibitor arc

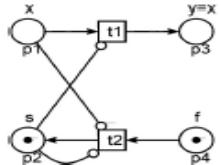
Basic statements



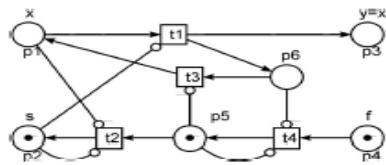
Basic subnets (subroutines)



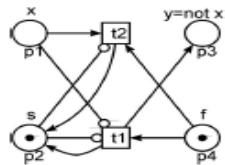
CLEAN:
 $x := 0$



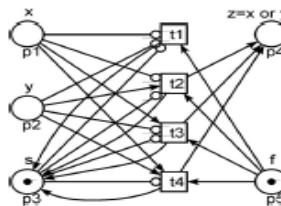
MOVE:
 $y := x, x := 0$



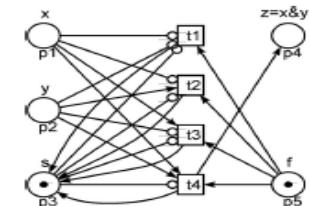
COPY:
 $y := x$



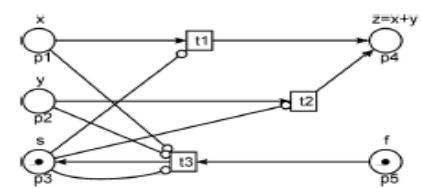
NOT(x):
 $y := -x, x := 0$



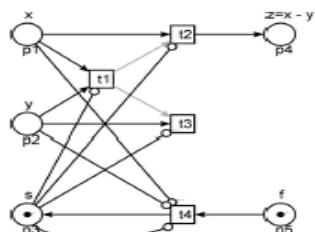
OR(x,y)
 $z := x \vee y, x := 0, y := 0$



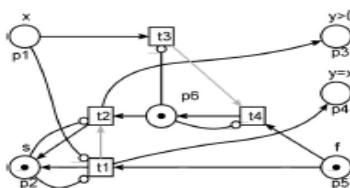
AND(x,y):
 $z := x \wedge y, x := 0, y := 0$



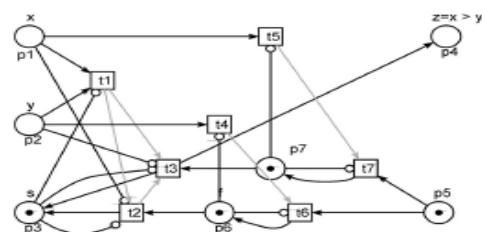
ADD(x,y):
 $z := x + y, x := 0, y := 0$



SUB(x,y):
 $z := x - y, x := 0, y := 0$

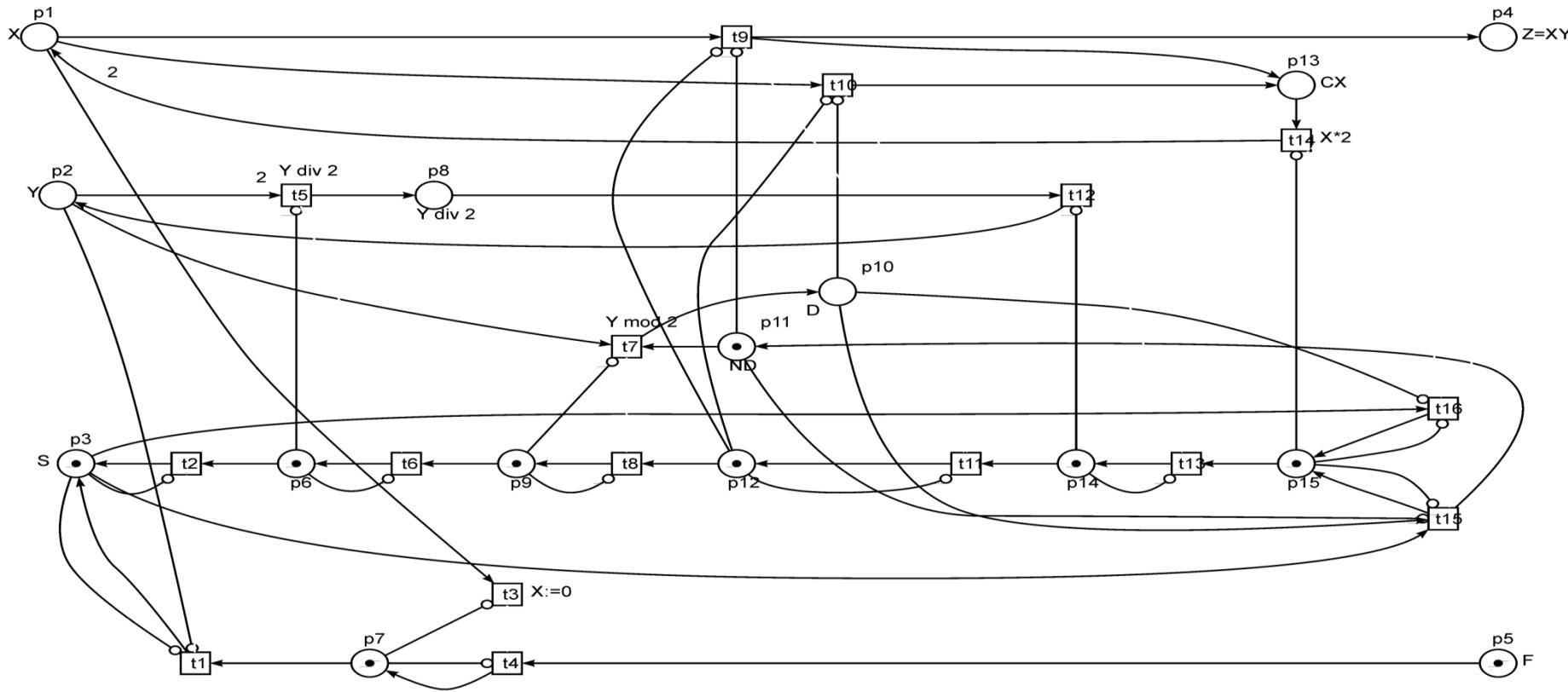


GT0(x):
 $y := (x > 0), z := (x = 0)$

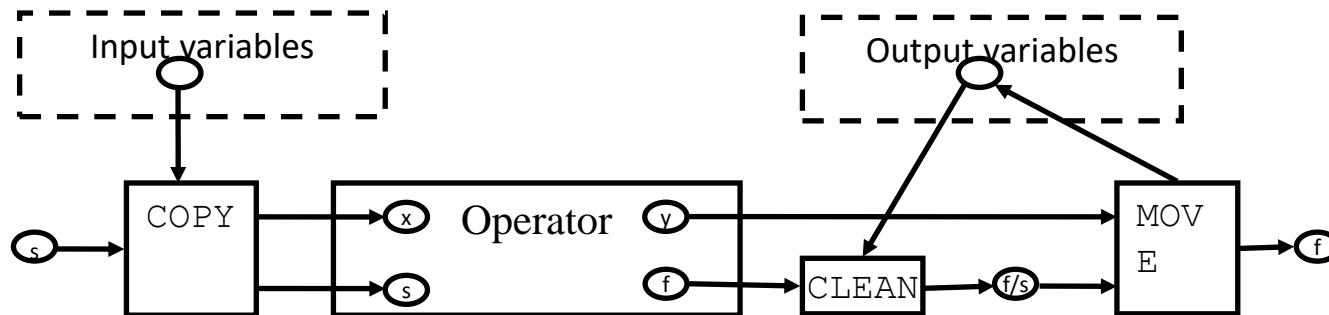
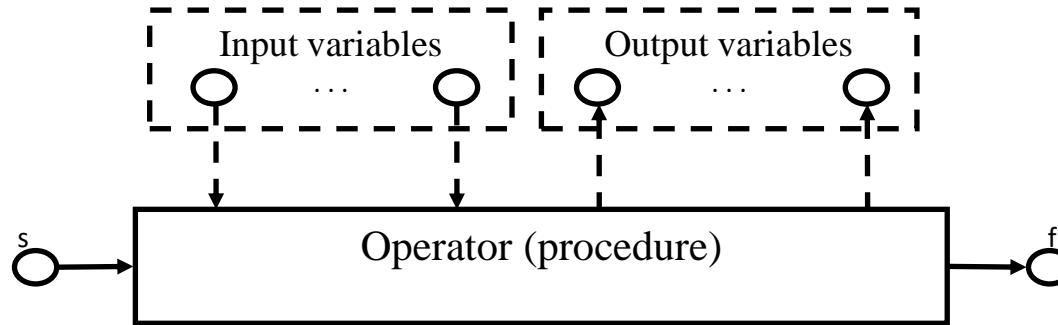


GT(x,y):
 $z := (x > y), x := 0, y := 0$

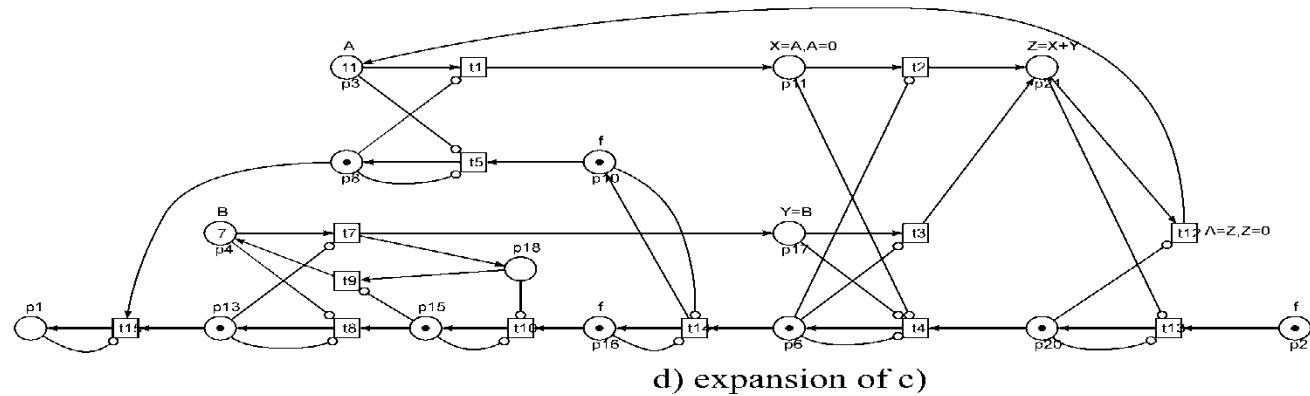
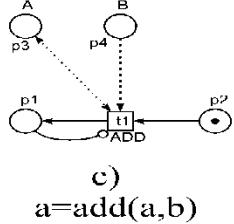
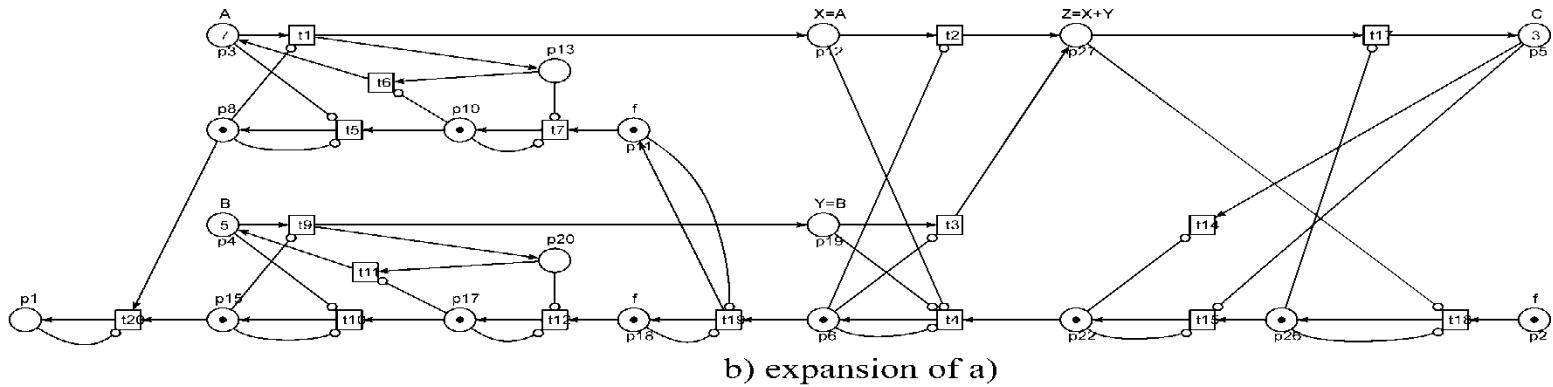
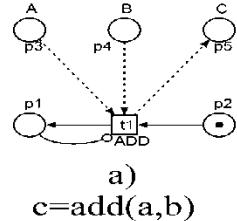
Fast multiplication in Sleptsov nets



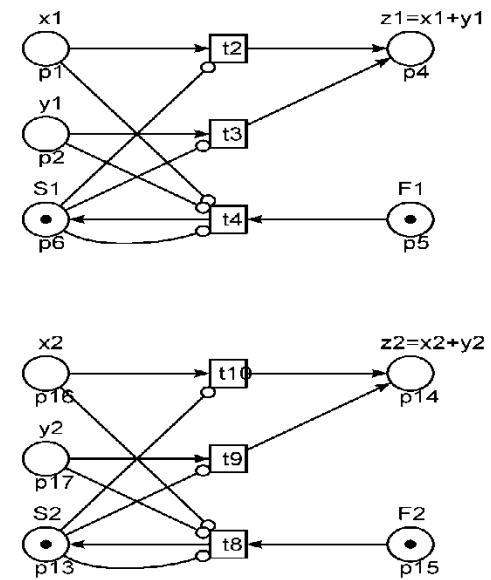
Work with variables



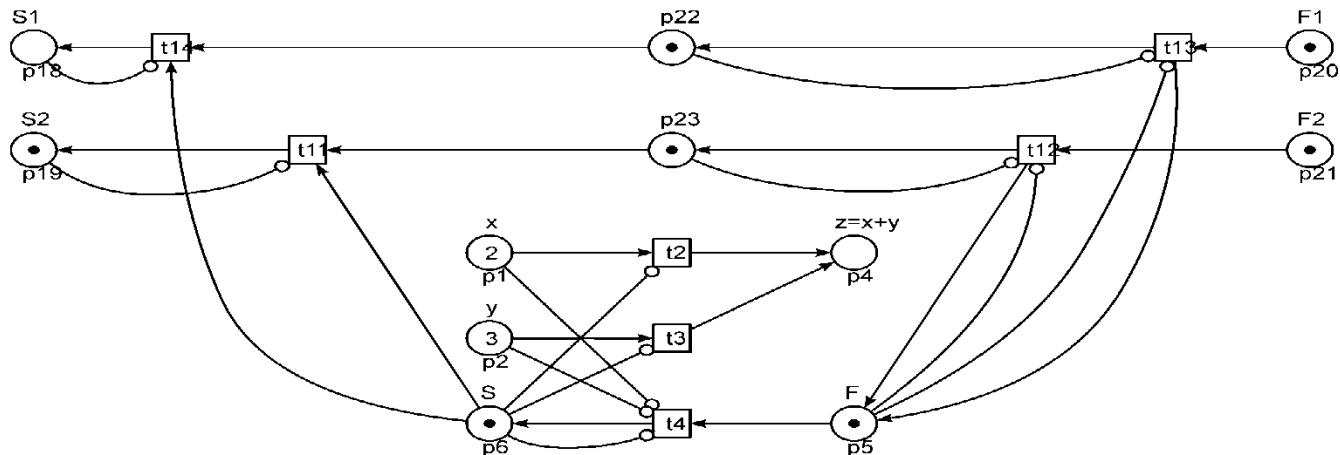
Expansion of dashed/dotted arcs



Subnets (routines) calls



a) inline

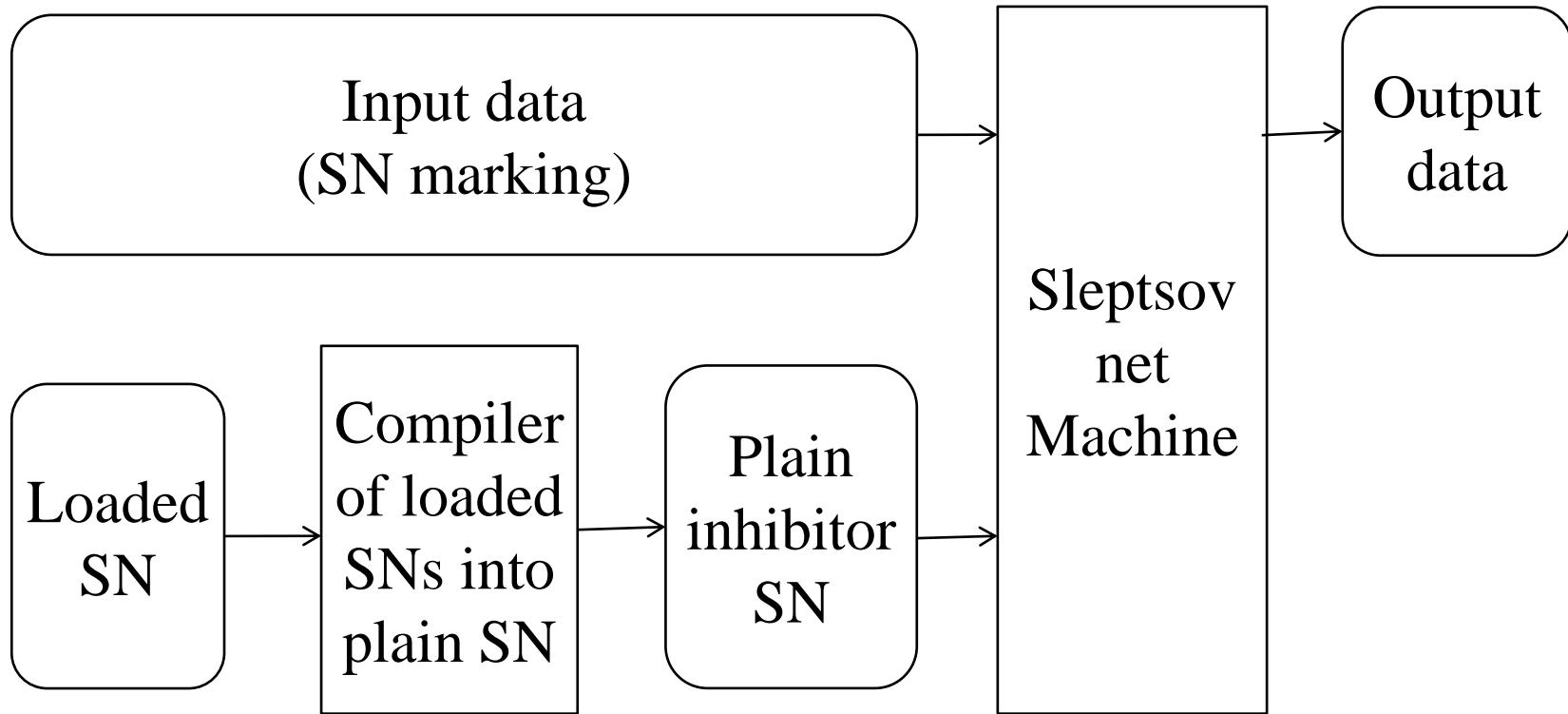


b) call-return

Verification of SN programs

- Boundedness and conservativeness
- Liveness (absence of deadlocks)
- Proof of program function
- Methods:
 - solving Diophantine systems in nonnegative numbers for linear invariants
 - solving systems of logic equations for finding siphons and traps
 - symbolic state space techniques
 - reduction and decomposition (into clans)

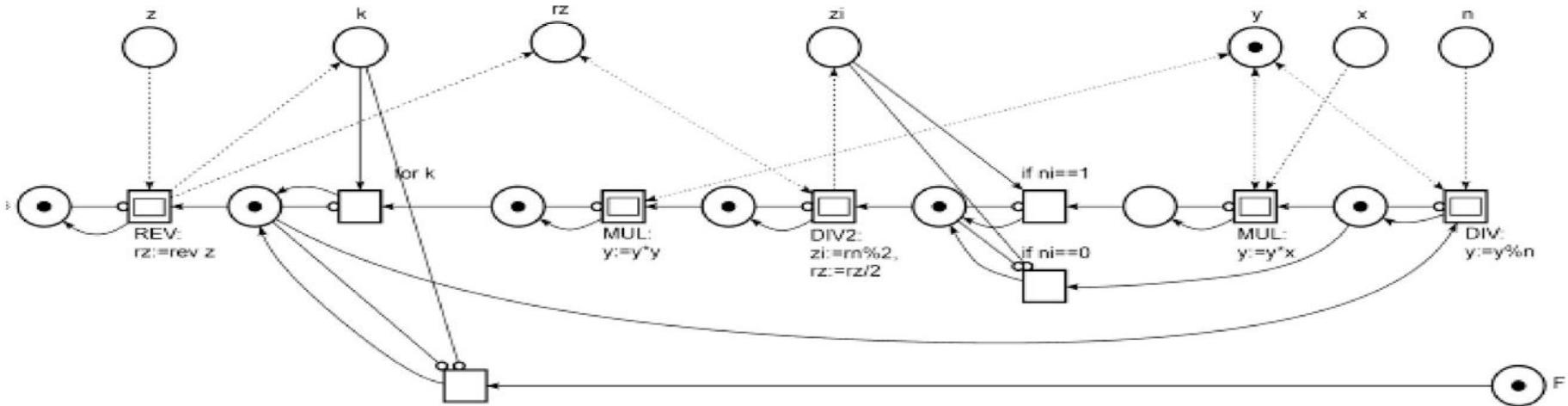
Sleptsov Net Paradigm of Computing



Sleptsov net hardware machine

- Reconfigurable multidimensional sparse matrices of connected transitions and places
- Each transition represents a separate asynchronous process that repeats the following stages:
 - lock the set of incidental places or wait their availability to lock them;
 - compute the firing multiplexity c ;
 - if $c=0$, unlock incidental places and wait update (increment);
 - if $c>0$, fire in c instances and unlock incidental places.
- A transition implements Sleptsov firing rule using division, multiplication, subtraction, addition, and minimum

Examples of SN programs: RSA encoding/decoding



$$y = x^z \bmod n$$

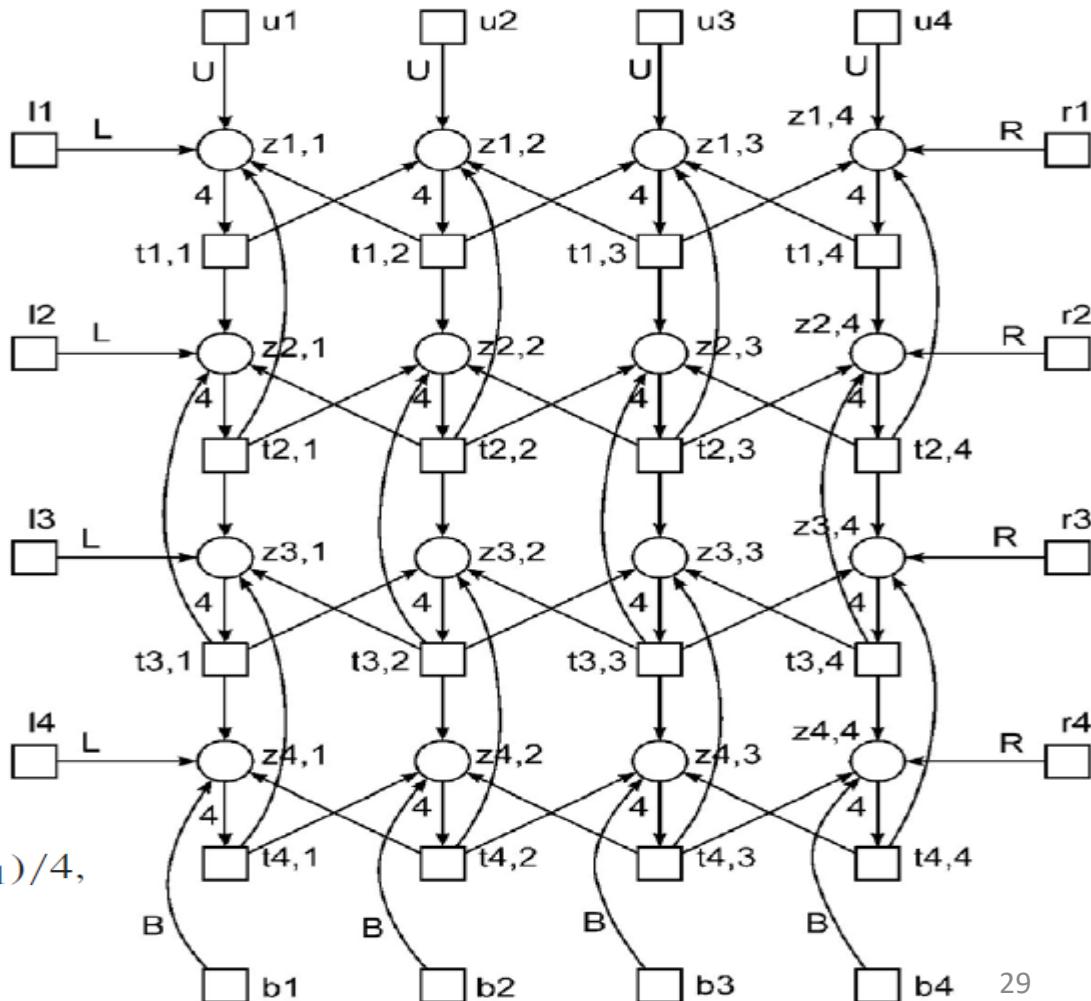
$$\begin{aligned} x^z &= x^{...((z_k \cdot 2 + z_{k-1}) \cdot 2 + z_{k-2}) \dots + z_2) \cdot 2 + z_1} \\ &= \left(\dots \left(\left(\left(x^{z_k} \right)^2 \cdot x^{z_{k-1}} \right)^2 \cdot x^{z_{k-2}} \right)^2 \cdot \dots \cdot x^{z_2} \right)^2 \cdot x^{z_1} \end{aligned}$$

Examples of SN programs: Solving Laplace equation

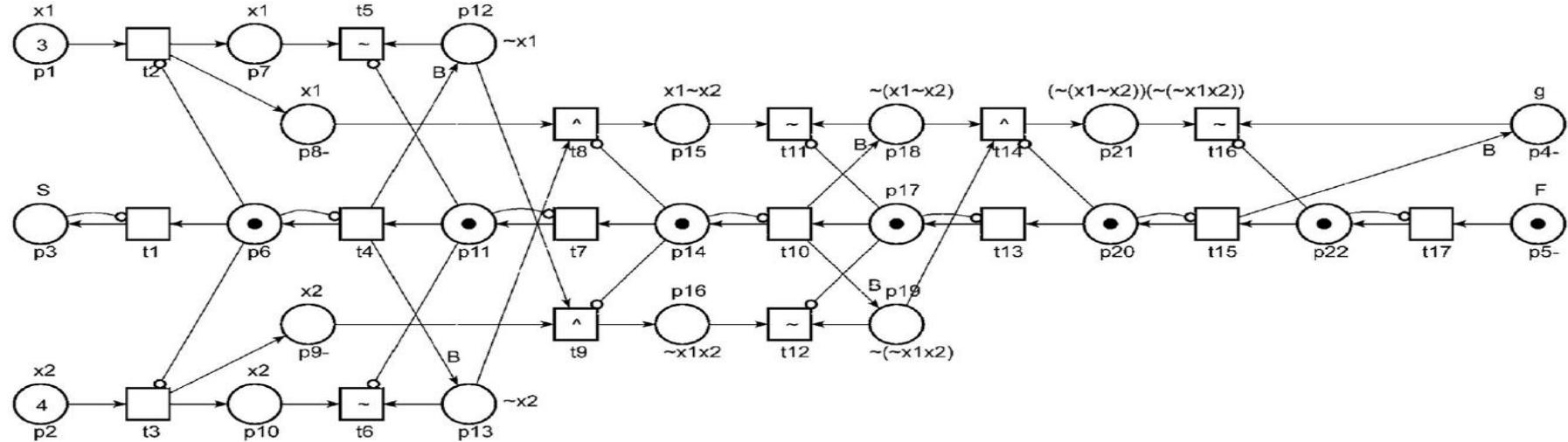
$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$

$$\varphi_{i,j} = (\varphi_{i-1,j} + \varphi_{i+1,j} + \varphi_{i,j-1} + \varphi_{i,j+1})/4,$$

$$\varphi_{i,j} = \varphi(x_i, y_j).$$

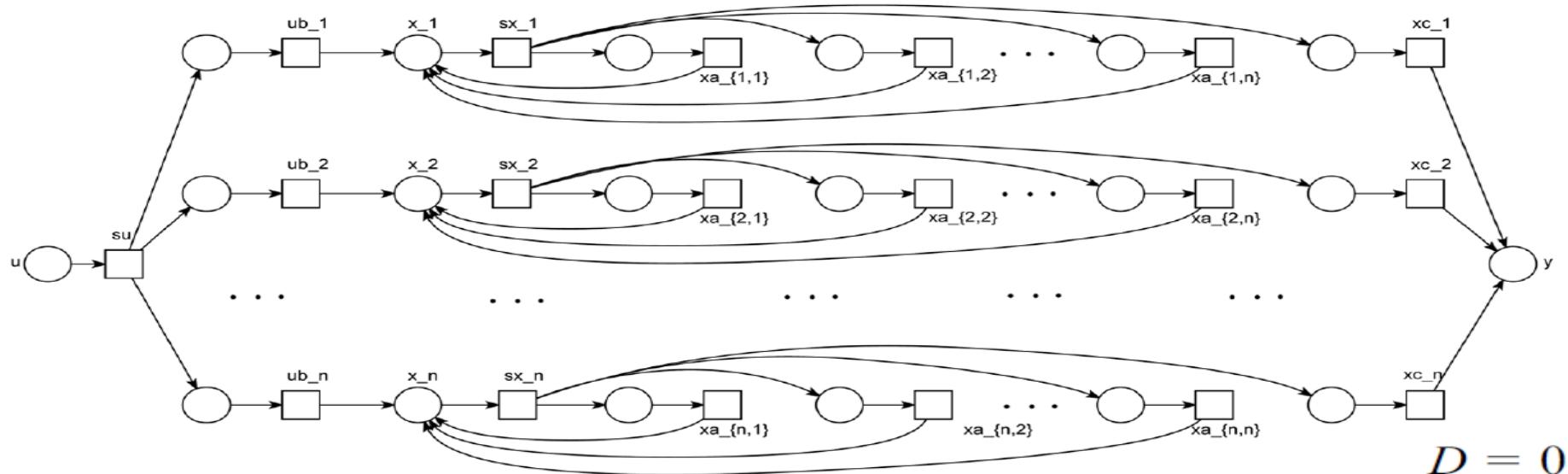


Examples of SN programs: Computing Fuzzy Logic Function



$$\varphi = x_1 \bar{x}_2 \vee \bar{x}_1 x_2$$

Examples of SN programs: Discrete-Time Linear Control in Two Ticks



$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Dy(k) \end{cases}$$

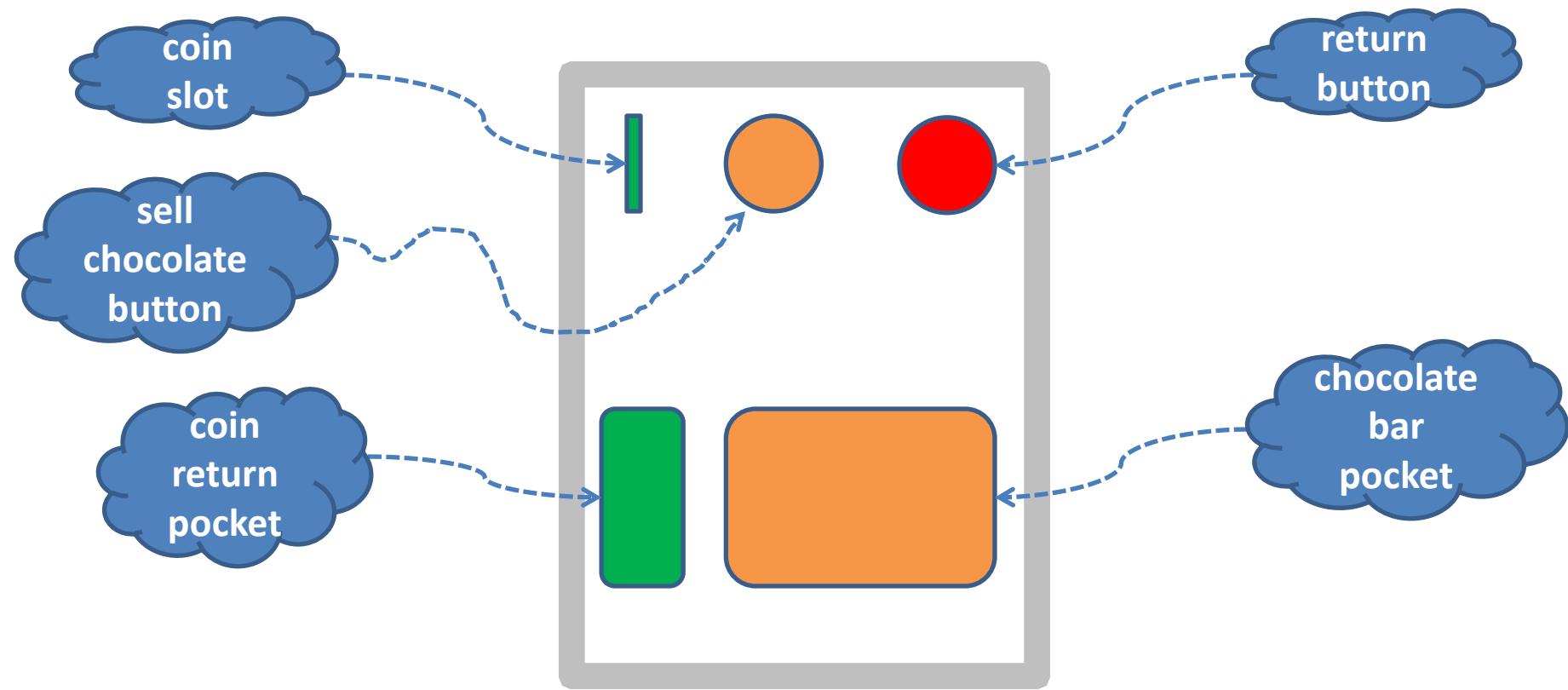
General Purpose Software Design in Sleptsov Net Machine

- SN Virtual Machine on multicore CPU – Qing Zhang, 2023
- SN Virtual Machine on GPU – Tatiana R. Shmeleva, 2023
- Compiler-linker of SN for hierarchical design – Hongfei Zhao, 2023
- Modeling system Tina for graphical design, LAAS, CNRS, France

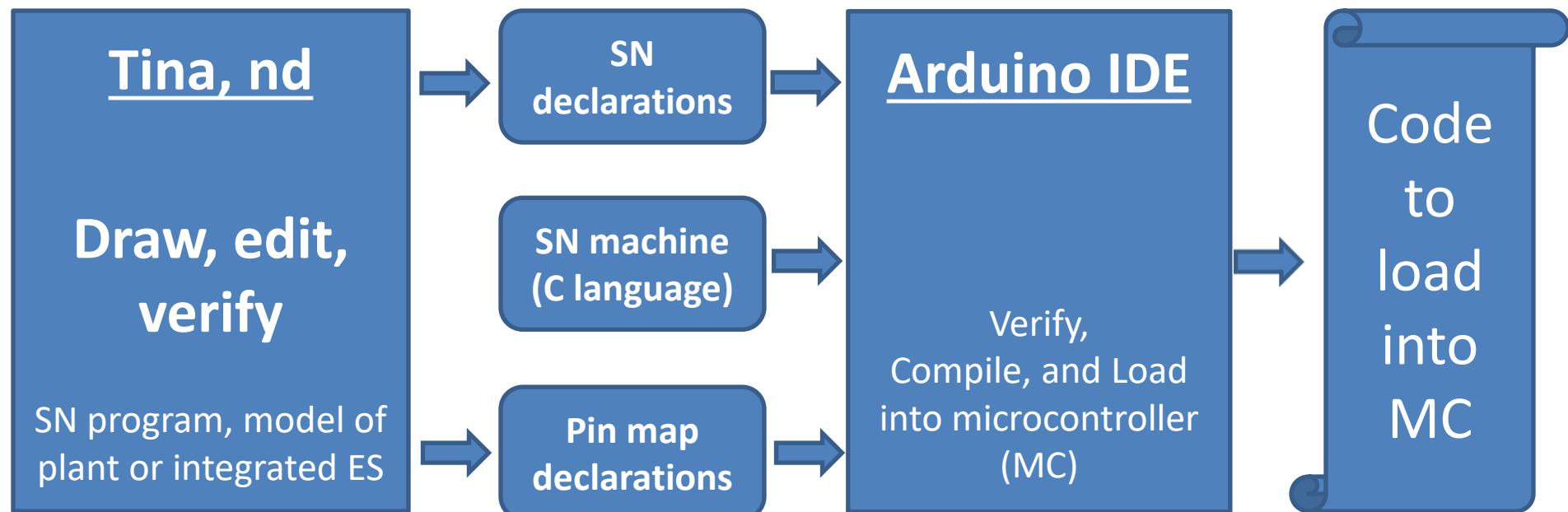
Reliable Embedded System Graphical Design in Sleptsov Net Machine

- SN machine on microcontrollers (in [Arduino IDE](#)) – Ruiyao Xu, 2024
- SN machine on FPGA (in [Gowin FPGA Designer](#)) – Si Zhang, 2024
- [Modeling system Tina](#) for graphical design, LAAS, CNRS, France

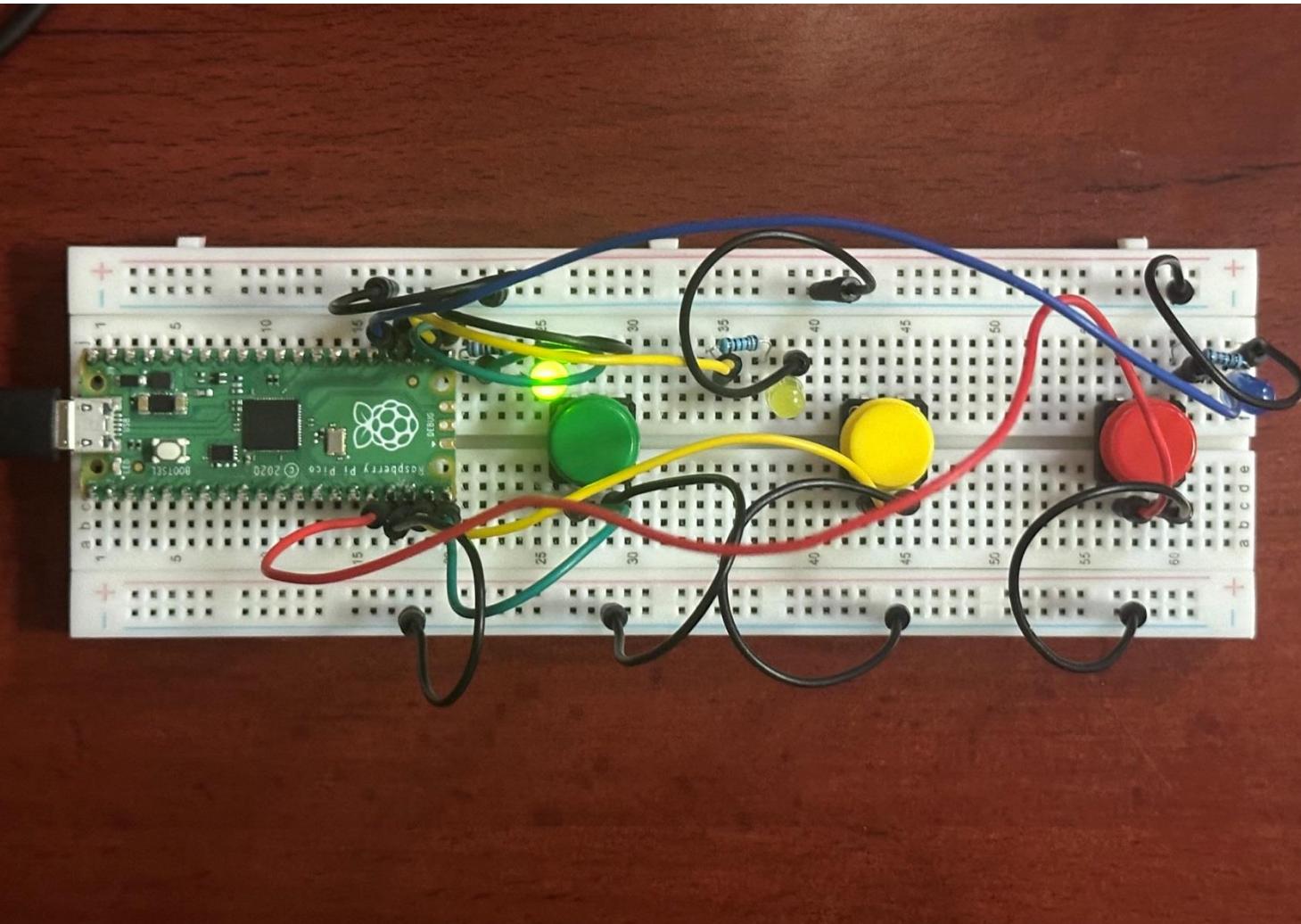
Vending machine example



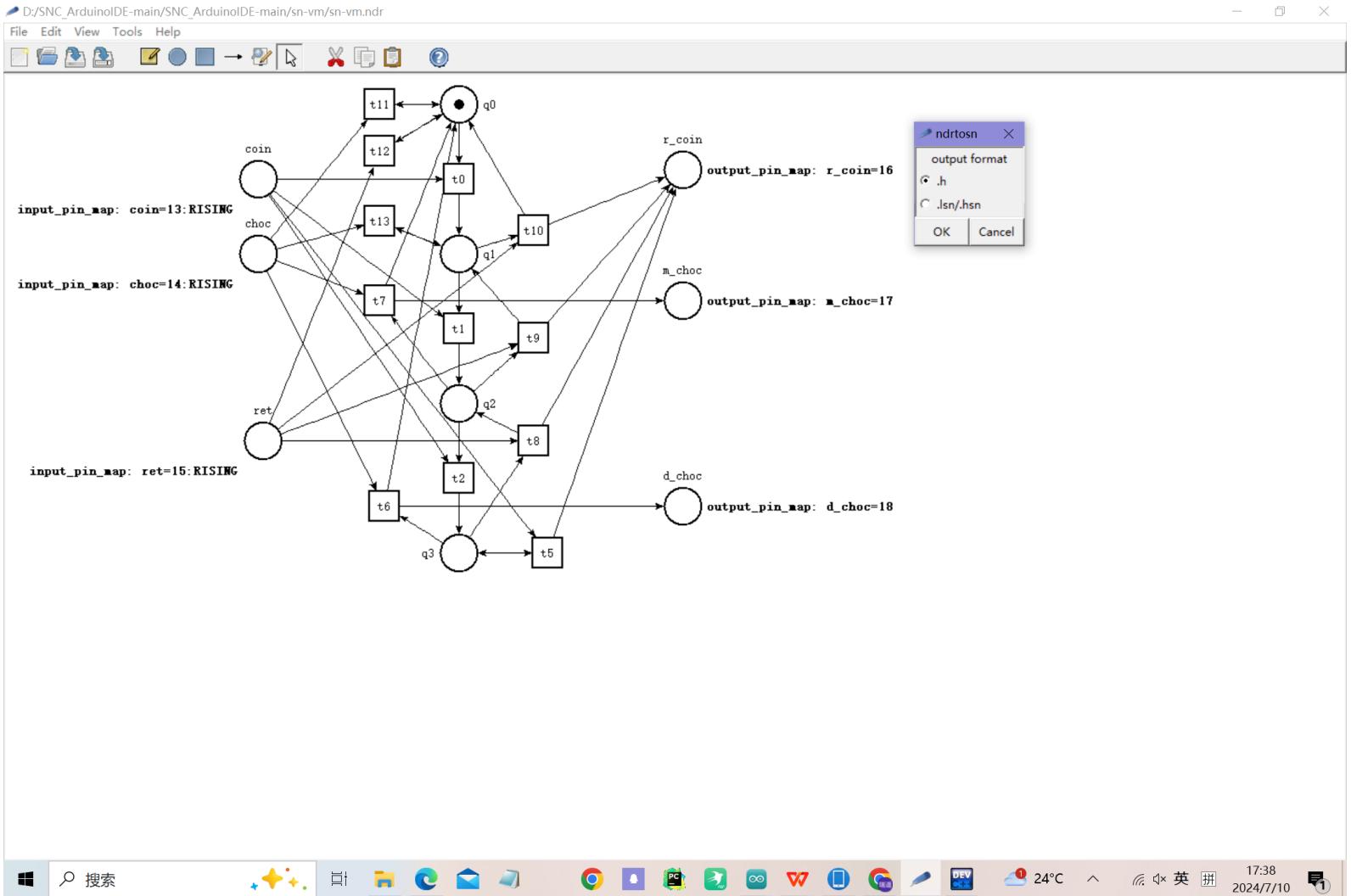
General workflow of SN machine design for microcontrollers



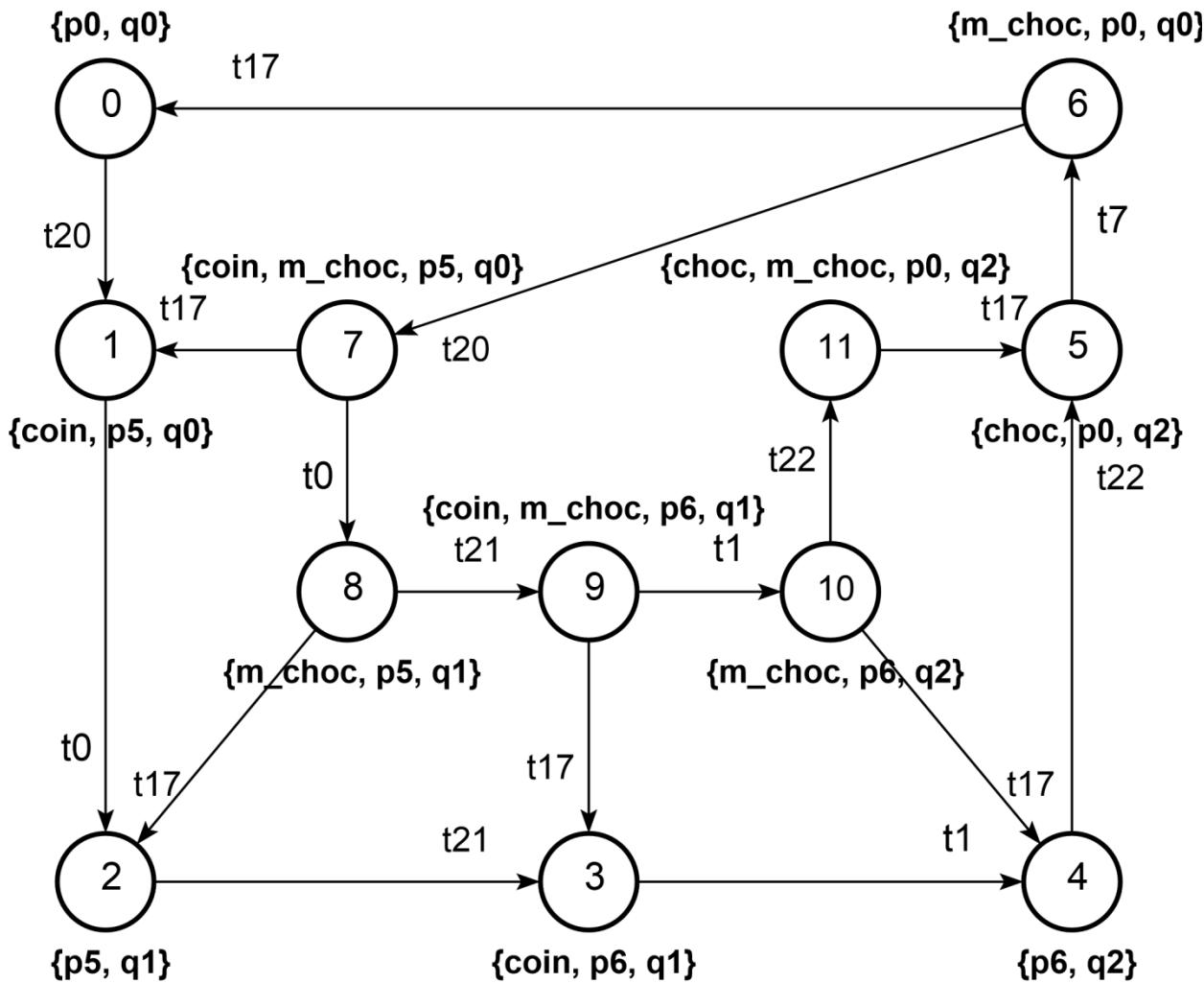
Prototyping with Raspberry Pi Pico



VM control SN program in Tina



Formal verification: State Space analysis



Convert to sparse matrix format

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** D:\NDRtoSN-main\NDRtoSN-main\sns3.c - Dev-C++ 5.11
- Menu Bar:** 文件(F) 编辑(E) 搜索(S) 视图(V) 项目(P) 运行(R) 工具(T) AStyle 窗口(W) 帮助(H)
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Build.
- Compiler:** TDM-GCC 4.9.2 64-bit Release
- Code Editor:** The file sns3.c contains C code for converting matrices to sparse format. The code includes headers for stdio.h and sn-vm-ndrtosn.h, defines m, n, KB, and KD, and prints static int bs[3] = {kb}.

```
1 #include <stdio.h>
2
3 #include "sn-vm-ndrtosn.h"
4
5 int main(){
6     printf("//Sleptsov-net in sparse matrix format\n");
7     int kb = 0;
8     int kd = 0;
9     int i , j;
10    int t=0;
11    for(i = 0;i < m;i++){
12        for(j = 0;j < n;j++){
13            if(b[i][j] != 0){
14                kb++;
15            }
16        }
17    }
18    for(i = 0;i < m;i++){
19        for(j = 0;j < n;j++){
20            if(d[i][j] != 0){
21                kd++;
22            }
23        }
24    }
25    printf("#define m %d\n",m);
26    printf("#define n %d\n",n);
27    printf("#define KB %d\n",kb);
28    printf("#define KD %d\n",kd);
29    printf("static int bs[%d]={\n",kb);
```

- Toolbars:** 编译器, 资源, 编译日志, 调试, 搜索结果, 关闭
- Output Window:** 显示编译结果，输出文件名为 D:\NDRtoSN-main\NDRtoSN-main\sns3.exe
- Status Bar:** 行: 37 列: 6 已选择: 0 总行数: 55 长度: 1311 插入 在 0.094 秒内完成解析

Compose SN machine in Arduino IDE

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sna1 | Arduino IDE 2.3.2
- Menu Bar:** File, Edit, Sketch, Tools, Help
- Toolbar:** Includes icons for Save, Run, Stop, and others.
- Sketch Name:** Raspberry Pi Pico
- Code Editor:** Displays the `sna1.ino` file content. The code includes comments about SNM and pin mapping, and defines functions for getting sparse binary and decimal values from arrays.
- Output Window:** Shows the compilation and upload process:
 - Sketch uses 54596 bytes (2%) of program storage space. Maximum is 2093056 bytes.
 - Global variables use 10884 bytes (4%) of dynamic memory, leaving 251260 bytes for local variables. Maximum is 262144 bytes.
 - Converting to uf2, output size: 145920, start address: 0x2000
 - Scanning for RP2040 devices
 - Flashing E: (RPI-RP2)
 - Wrote 145920 bytes to E:/NEW.UF2
- Status Bar:** Indicators for Offline status, connection to Raspberry Pi Pico on UF2 Board [not connected], battery level (5), and other system information.

Video-presentation of SN machine work on Raspberry Pi Pico microcontroller

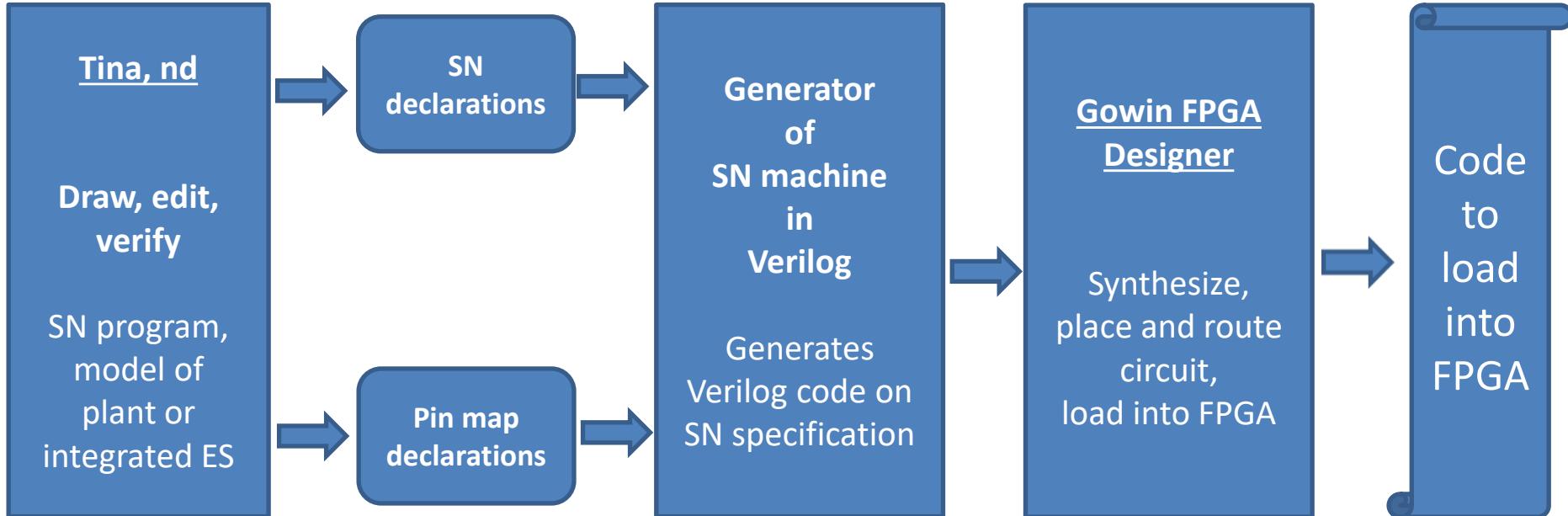


- Vending Machine on
Raspberry Pi Pico

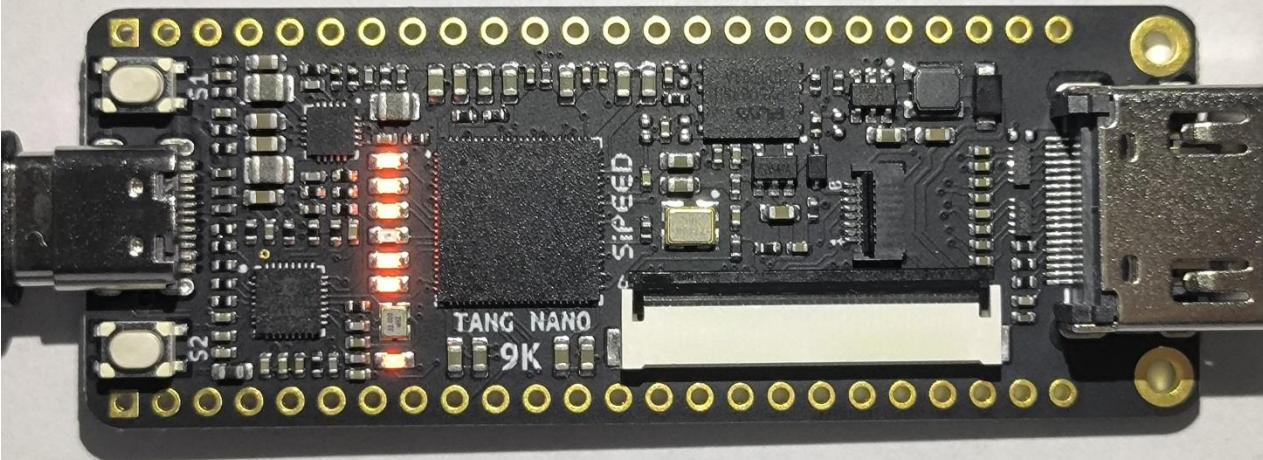
(哔哩哔哩)

<https://b23.tv/Ed21ze3>

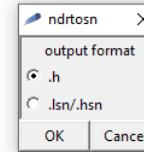
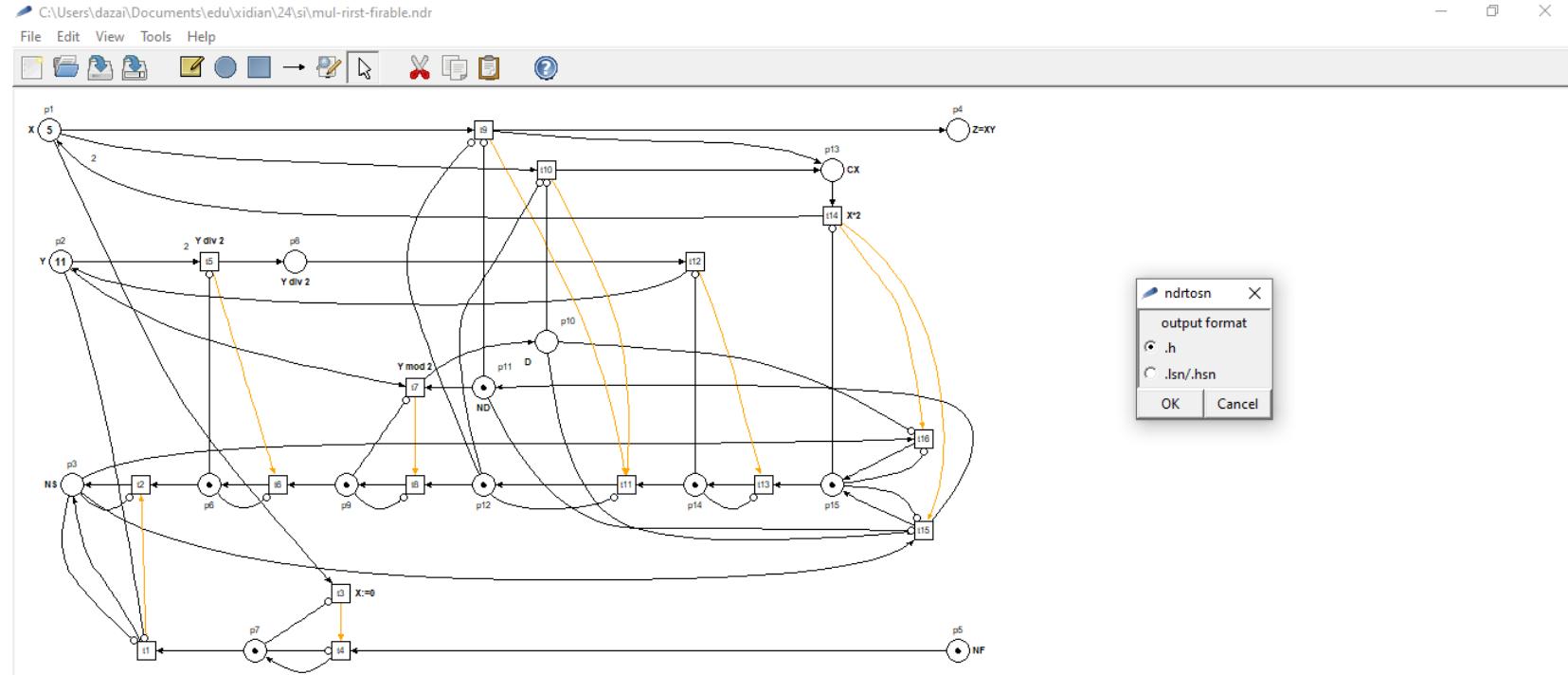
General workflow of SN machine design for FPGAs



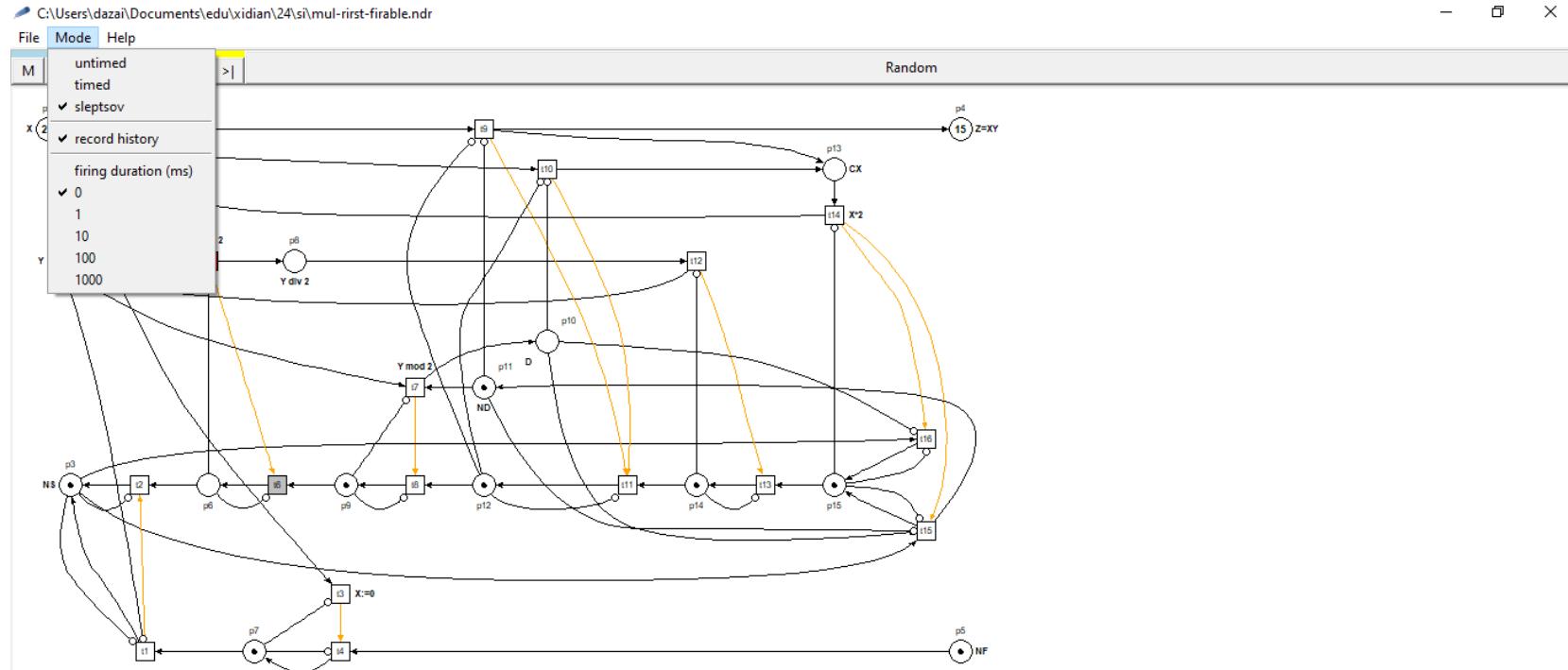
Run SN program on FPGA Tang Nano 9K



SN program for multiplication in Tina



Run SN program in Tina



SN machine generator (Dev-C)

D:\lyjsqj\zs\Task\Task4\generator\mul_snvg.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Project Classes mul_snvg.cpp mul_ndrtosn.h

```
1 #include <stdio.h>
2 #include "mul_ndrtosn.h" //mul
3
4 #define pres "[5:0]"
5 #define RES_PL 3
6
7 void int_to_binary(unsigned int num, char* binary) {
8     int i;
9     for (i = 5; i >= 0; i--)
10    {
11        binary[5 - i] = (num & (1 << i)) ? '1' : '0';
12    }
13 }
14
15 int main()
16 {
17     int i,j,t;
18     char binary[6];
19
20     printf("regis ",pres);
21     for(i=0;i<m;i++)
22    {
23         printf("p%d",i);
24         printf("%c", (i<m-1)?',';:');
25     }
26     printf("\nreg% s ",pres);
27     for(j=0;j<n;j++)
28    {
29         printf("f%d",j);
30         printf("%c", (j<n-1)?',';:');
31     }
32     printf("\nreg% s tf;",pres);
33     printf("\nreg% s tc;",pres);
34     printf("\nalways @posedge sys_clk or negedge sys_rst_n) begin\n"
35     if(!sys_rst_n) begin\n"
36     \ttf <= 6'b000000;\n"
37     \ttc <= 6'b000000;\n"
```

Compiler Resources Compile Log Debug Find Results Close

Abort Compilation

- Errors: 0
- Warnings: 0
- Output Filename: D:\lyjsqj\zs\Task\Task4\generator\mul_snvg.exe
- Output Size: 133.880859375 Kib
- Compilation Time: 0.64s

Line: 51 Col: 34 Sel: 0 Lines: 105 Insert Done parsing in 0.016 seconds

Verilog code in Gowin FPGA Designer

GOWIN FPGA Designer - [D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64]\DE\bin\Documents\add_sn\src\mul_snvg.v*

File Edit Project Tools Window Help

Process

- Design Summary
- User Constraints
 - FloorPlanner
 - Timing Constraints Editor
- Synthesize
 - Synthesis Report
 - Netlist File
- Place & Route
 - Place & Route Report
 - Timing Analysis Report
 - Ports & Pins Report
 - Power Analysis Report
- Programmer

```
1 module mul_sn(
2     input sys_clk,
3     input sys_rst_n,
4     output reg [5:0] led
5 );
6 function [5:0] MIN;
7     input [5:0] a, b;
8     MIN = (a < b) ? a : b;
9 endfunction
10
11 function automatic [5:0] INH;
12     input [5:0] place; // 假设place是一个6位的寄存器
13     // 如果place为0，则INH返回6'b111111 (十进制的63)
14     // 否则返回0，表示有抑制
15     INH = (place == 0) ? 6'b111111 : 6'b000000;
16 endfunction
17
18 reg[5:0] p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14;
19 reg[5:0] f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15;
20 reg[5:0] tf;
21 reg[5:0] tc;
22 always @(posedge sys_clk or negedge sys_rst_n) begin
23     if(!sys_rst_n) begin
24         tf <= 6'b000000;
25         tc <= 6'b000000;
26         led <= 6'b000000;
27         p0 <= 6'b000101;
28         p1 <= 6'b001011;
29         p2 <= 6'b000000;
30         p3 <= 6'b000000;
```

Design Process Design Summary mul_snvg.v* Start Page

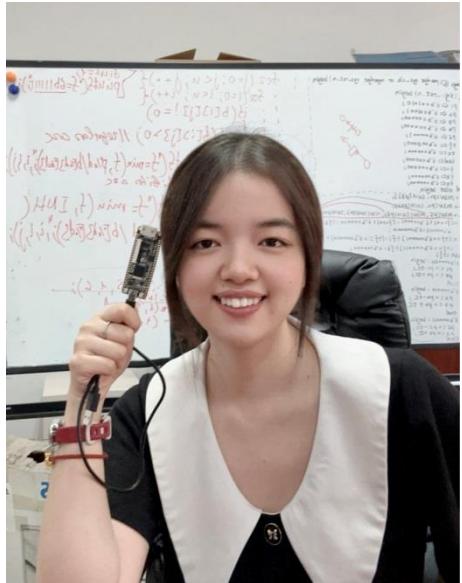
Console

```
PSL file generation completed
Running power analysis.....
[100%] Power analysis completed
Generate file "D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64\IDE\bin\Documents\add_sn\impl\pnr\add_sn.power.html" completed
Generate file "D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64\IDE\bin\Documents\add_sn\impl\pnr\add_sn.pin.html" completed
Generate file "D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64\IDE\bin\Documents\add_sn\impl\pnr\add_sn.rpt.html" completed
Generate file "D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64\IDE\bin\Documents\add_sn\impl\pnr\add_sn.rpt.txt" completed
Generate file "D:\1yjsqj\zs\Task\Gowin\Gowin_V1.9.9.03_x64\IDE\bin\Documents\add_sn\impl\pnr\add_sn.tr.html" completed
Thu Jul 11 15:03:05 2024
```

%

Console Message In: 50 Col: 33

Video-presentation of SN machine work on FPGA Tang Nano 9k (Gowin IC)



- SN machine design for FPGAs

(哔哩哔哩)

<https://b23.tv/c8JBHiY>

Conclusions

- **Sleptsov net computing benefits:**
 - graphical concurrent programming language,
 - formal verification of concurrent programs,
 - fine granulation of parallel processes,
 - massively parallel computations,
 - fast computing memory hardware concept
- **Sleptsov net computing application area:**
 - general purpose computing
 - reliable embedded system design

Basic References

- Zaitsev D.A. Sleptsov Nets Run Fast, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, Vol. 46(5), 682–693.
- Dmitry A. Zaitsev, Strong Sleptsov nets are Turing complete, Information Sciences, Vol. 621, 2023, 172-182.
- Bernard Berthomieu, Dmitry A. Zaitsev, Sleptsov Nets are Turing-complete, Theoretical Computer Science, Volume 986, 2024.
- Dmitry A. Zaitsev, Tatiana R. Shmeleva, Qing Zhang, and Hongfei Zhao, Virtual Machine and Integrated Developer Environment for Sleptsov Net Computing, Parallel Processing Letters, Vol. 33, No. 03, 2350006 (2023).

References for Sleptsov Net Computing (SNC) to read, watch, run, cite, and join

<https://dimazaitsev.github.io/snc.html>