Asynchronous system design flow based on Petri nets

Microelectronics System Design Research Group

School of Electrical, Electronic and Computer Engineering
University of Newcastle upon Tyne

DATE-2005

Outline

- Motivation
- BESST (BEhavioural Synthesis of Self-Timed Systems) design flow
- Splitting of control and data paths
- Synthesis of data path
- Adding security features to data path
- Direct mapping of control path from Labelled PNs
- Direct mapping of low-latency control path from STGs
- Logic synthesis of control path
- Summary

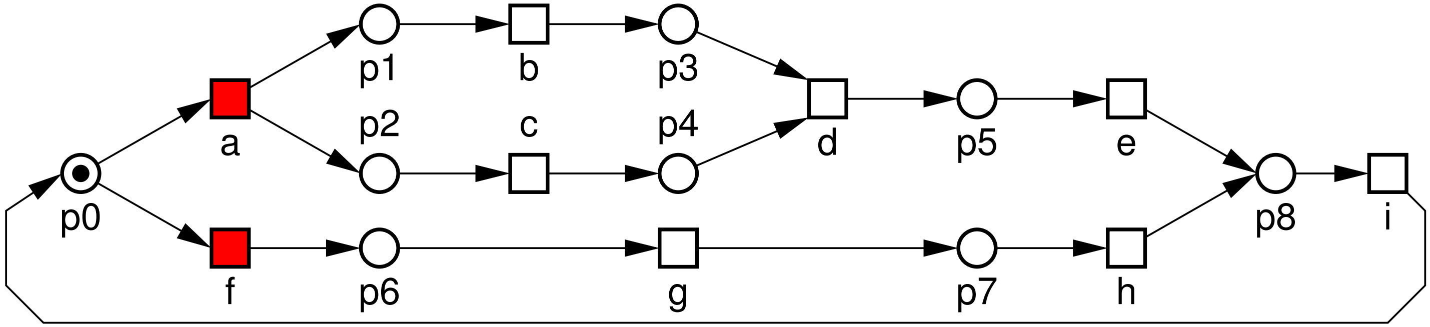
Motivation: Asynchronous circuits

- Asynchronous circuits address International Technology Roadmap for Semiconductors (ITRS) challenges

  ITRS-2003: “As it becomes impossible to move signals across large die within one clock cycle or in a power-efficient manner, or to run control and dataflow processes at the same clock rate, the likely result is a shift to asynchronous design style.”
- Modularity and scalability (productivity and reuse)
- Low noise and electromagnetic emission (security)
- Robustness to parametric variations
- No clock skew

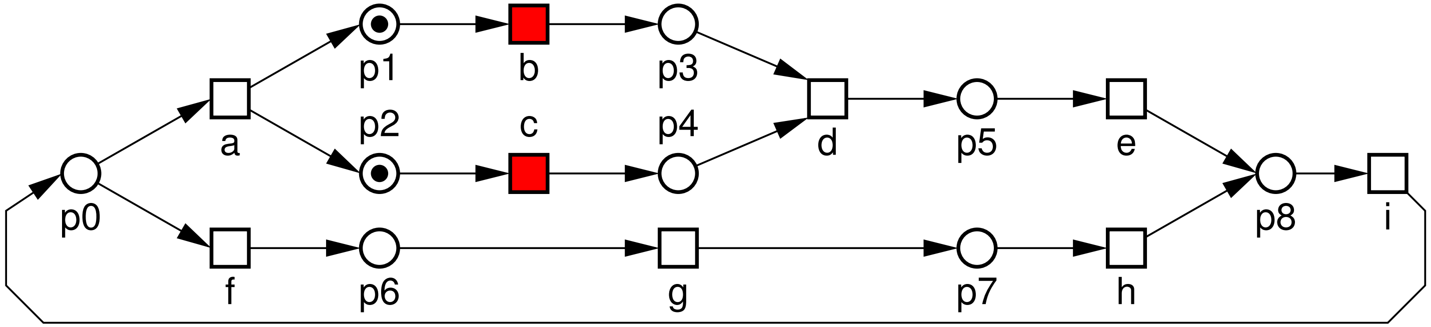
Motivation: Petri nets

- Well developed theory
- Powerful modeling language
- Simple to understand
- Can be hidden from the designer (intermediate system representation in our design flow)



Motivation: Petri nets

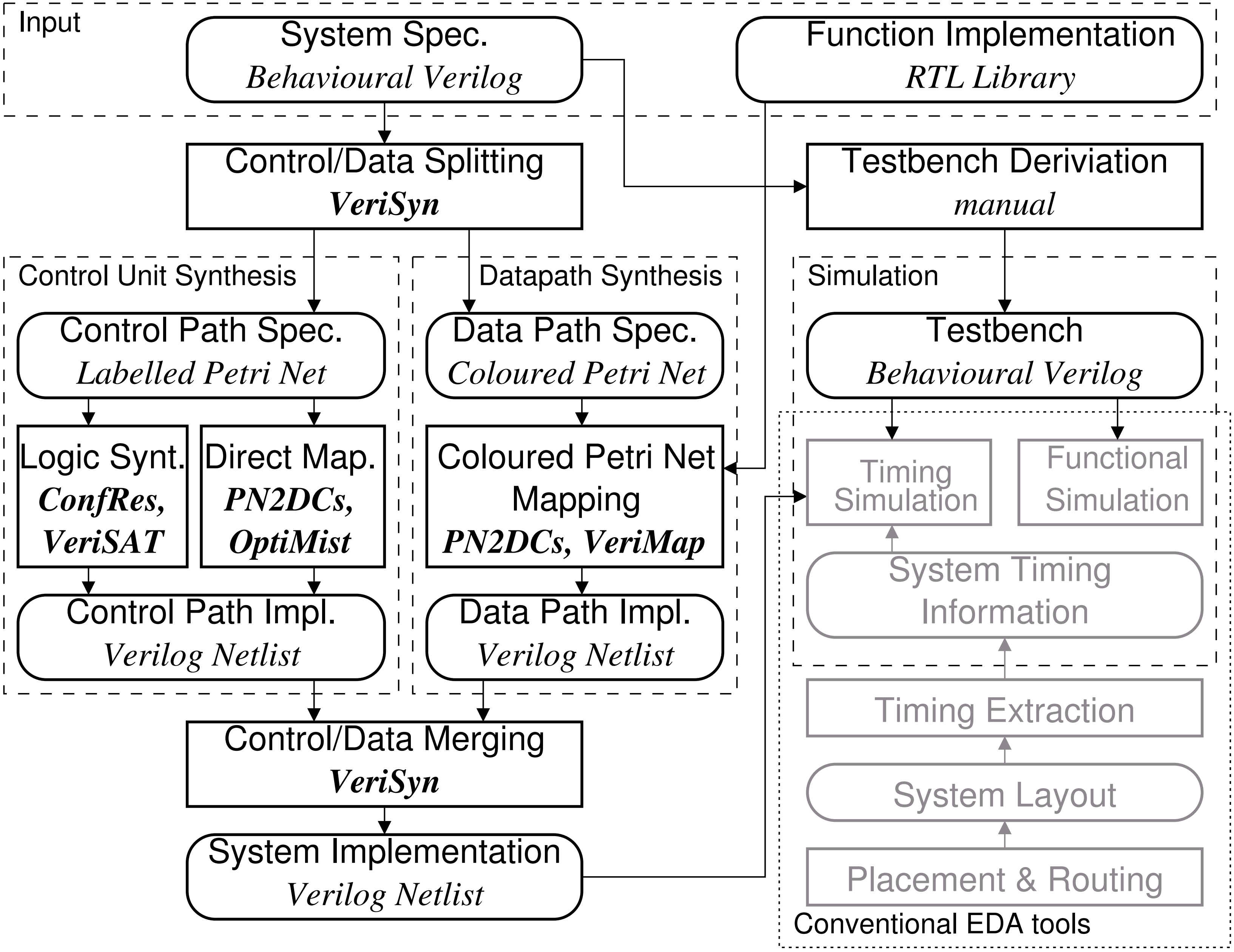
- Well developed theory
- Powerful modeling language
- Simple to understand
- Can be hidden from the designer (intermediate system representation in our design flow)



Asynchronous design flows

- Syntax-driven translation (Tangram, Balsa)
  - Computationally simple
  - Local peephole optimisation
  - Adopted by industry (Philips' incubator company Handshake Solutions)
  - Slow control circuit
- Logic synthesis (PipeFitter, TAST, MOODs, CASH)
  - Separate synthesis and optimisation of control and data paths
  - Adopted by industry (Theseus Logic (NCL))
  - Computationally hard for explicit logic synthesis

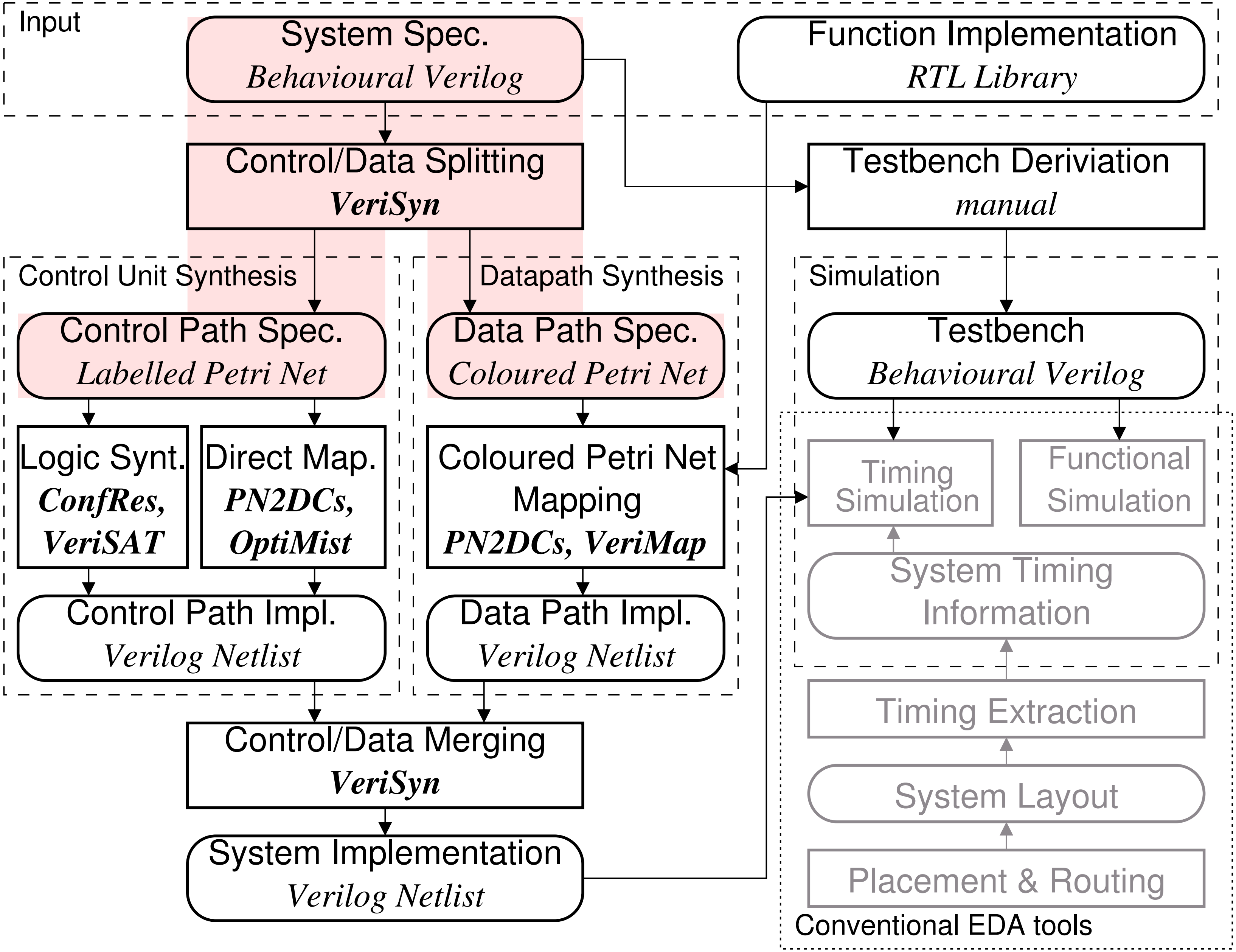
BESST Design flow



Greatest Common Divisor (GCD) spec.

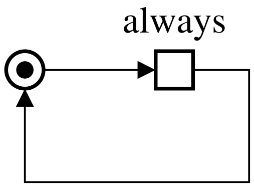
```
module gcd(x, y, z);
  input [7:0] x, y;
  output reg [7:0] z;
  reg [7:0] x_reg, y_reg;
  always
  begin
    x_reg = x; y_reg = y;
    while (x_reg != y_reg)
    begin
      if (x_reg < y_reg)
        y_reg = y_reg - x_reg;
      else
        x_reg = x_reg - y_reg;
    end
    z <= x_reg;
  end
endmodule
```

Splitting of control and data paths



Global PN for GCD

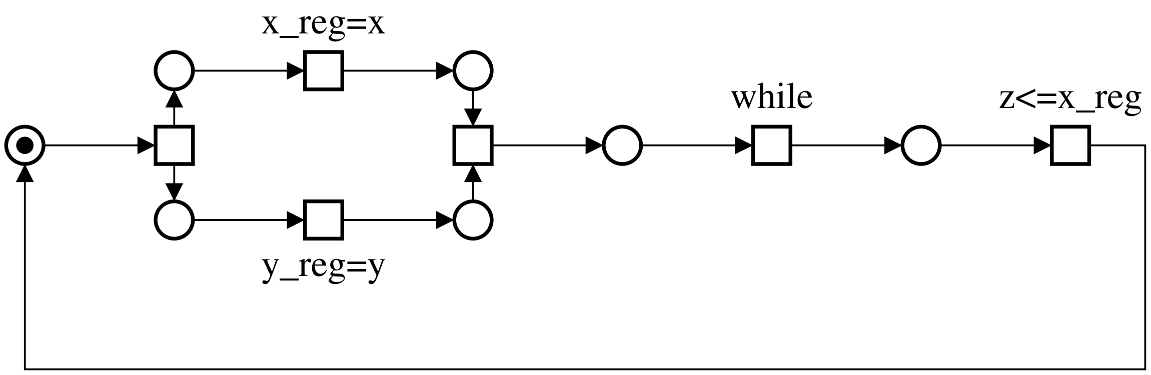
- Initial PN for GCD module



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

Global PN for GCD

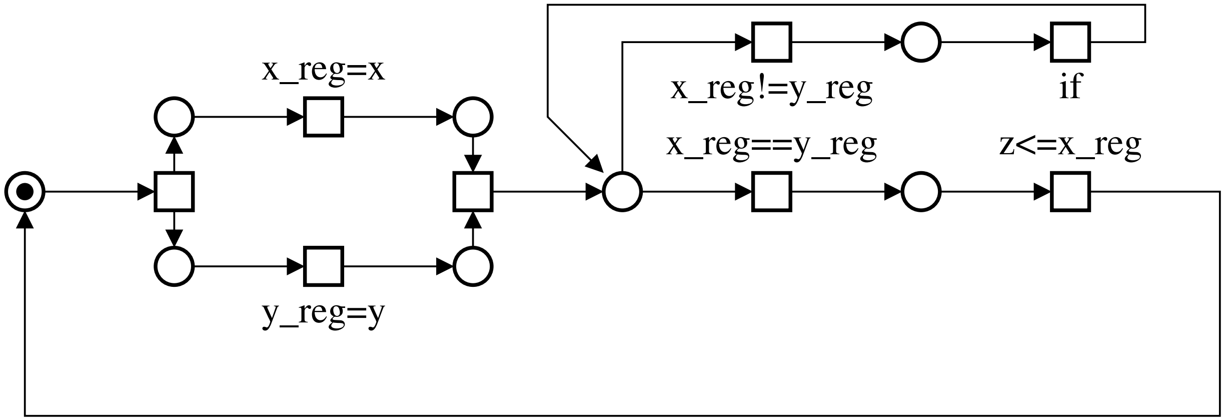
- Always-statement refinement



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

Global PN for GCD

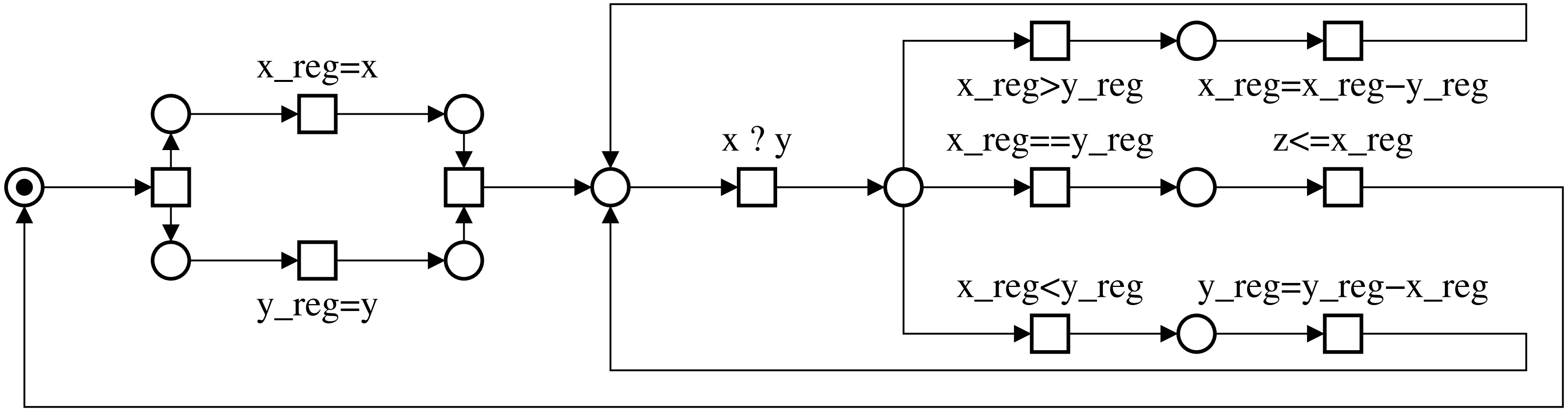
- While-statement refinement



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

Global PN for GCD

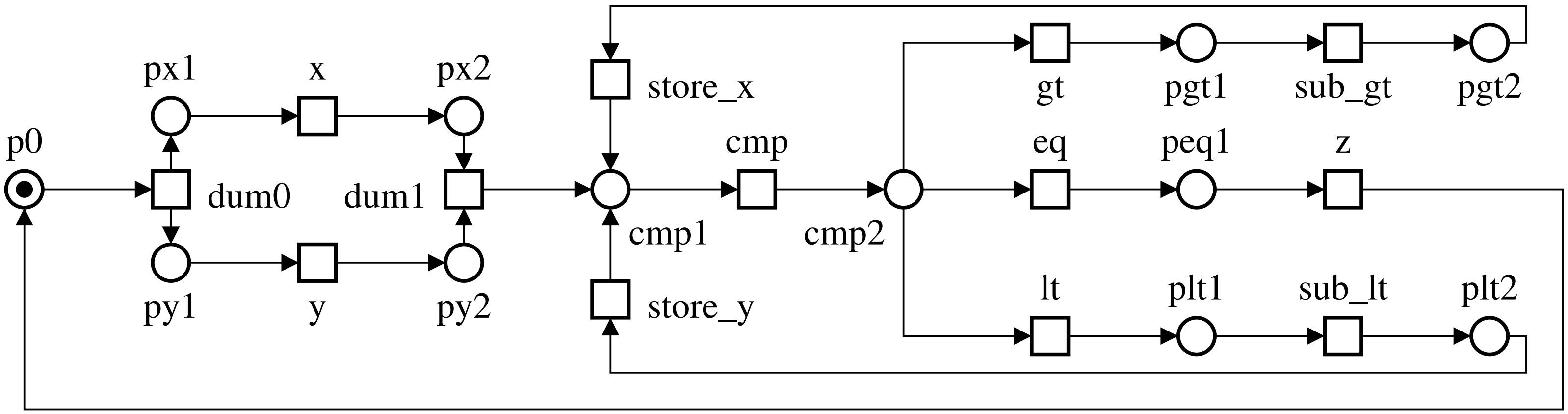
- If-statement refinement



```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

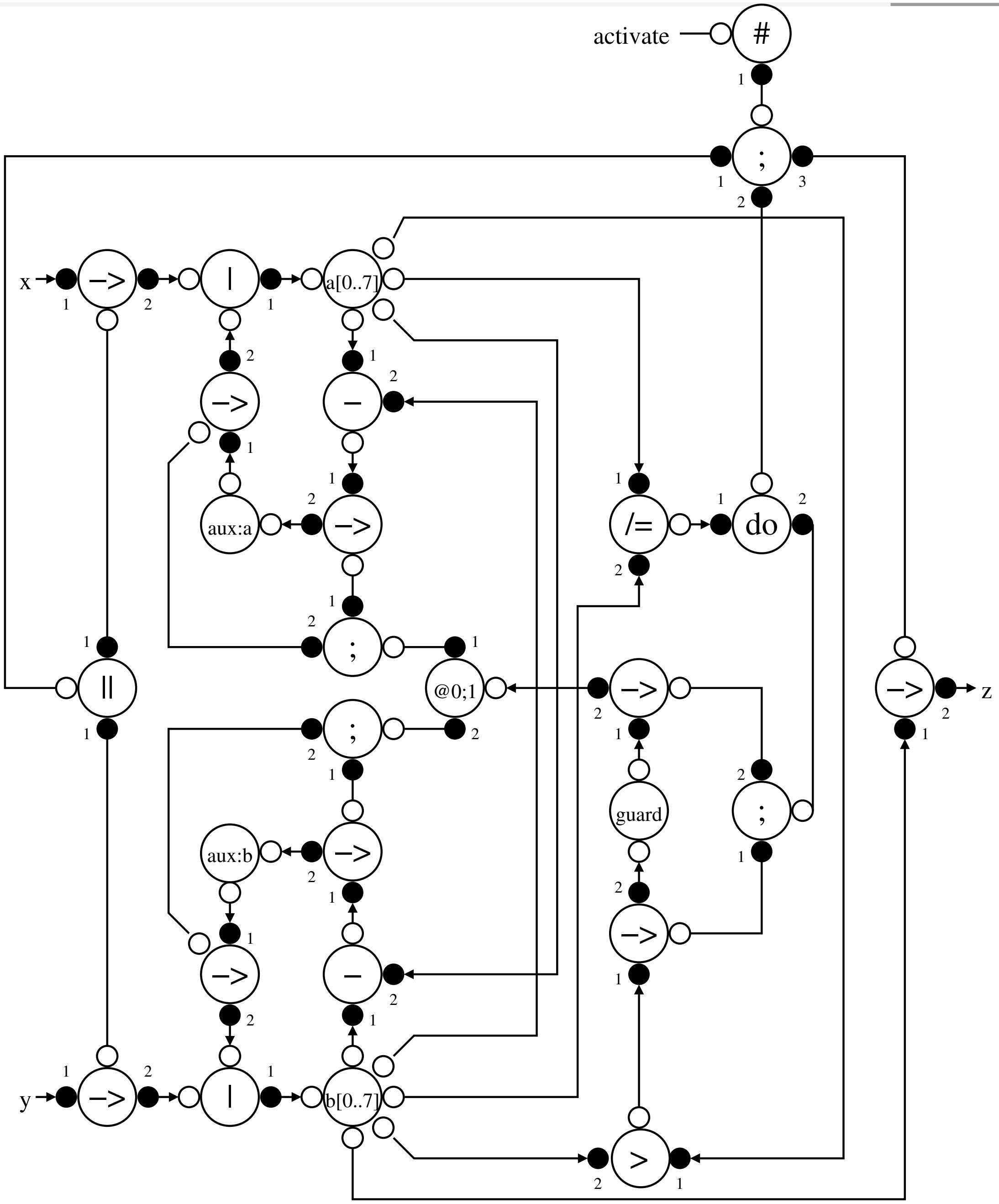
Global PN for GCD

- Assignment-operation refinement

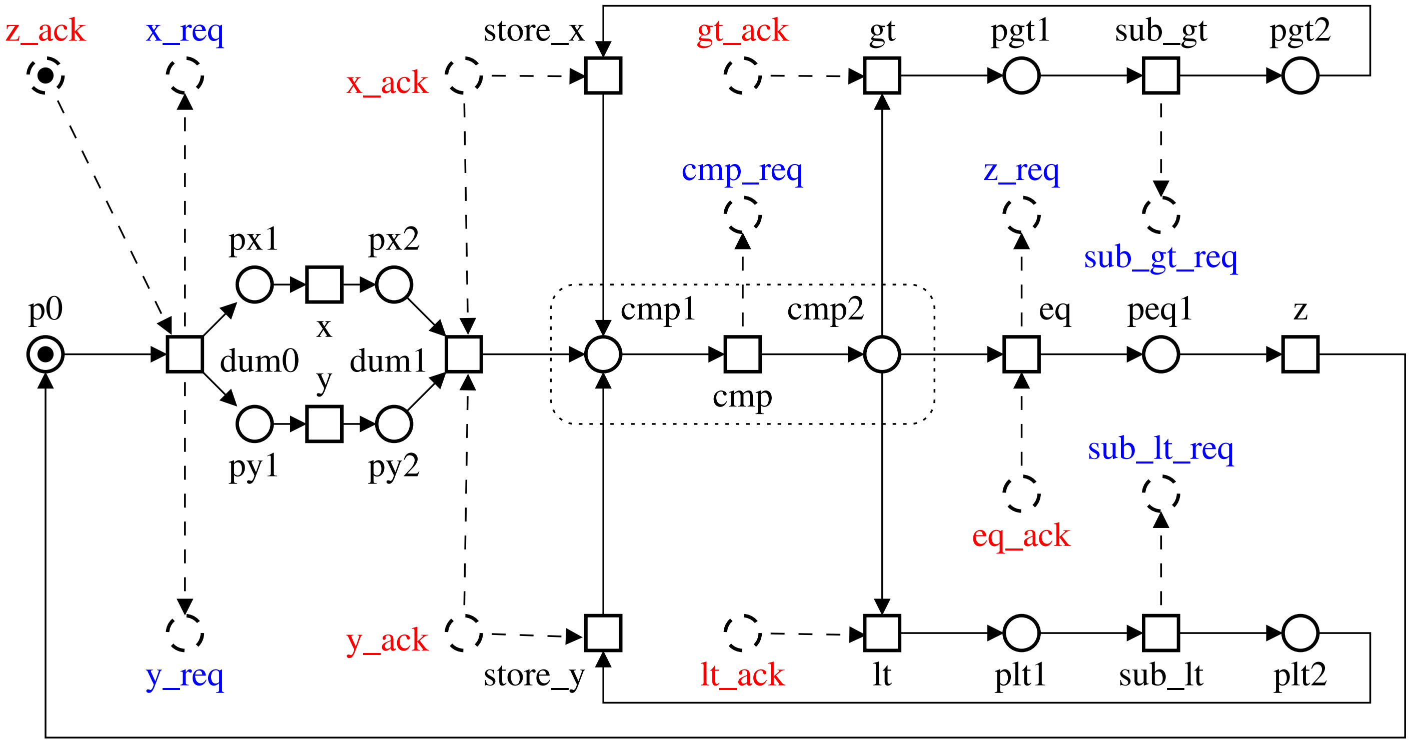


```
x_reg = x; y_reg = y;
while (x_reg != y_reg)
begin
  if (x_reg < y_reg)
    y_reg = y_reg - x_reg;
  else
    x_reg = x_reg - y_reg;
end
z <= x_reg;
```

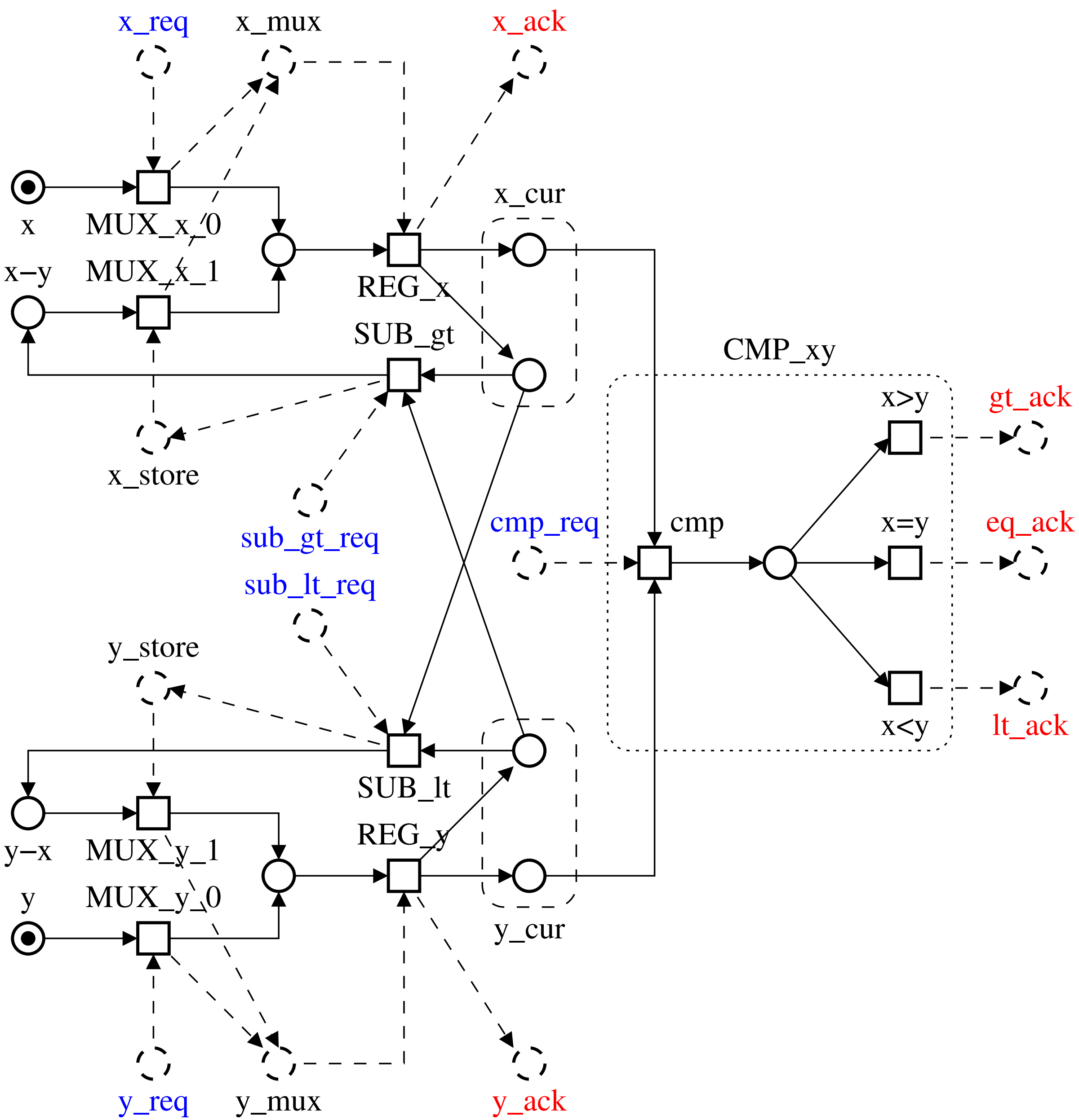
Syntax-driven translation



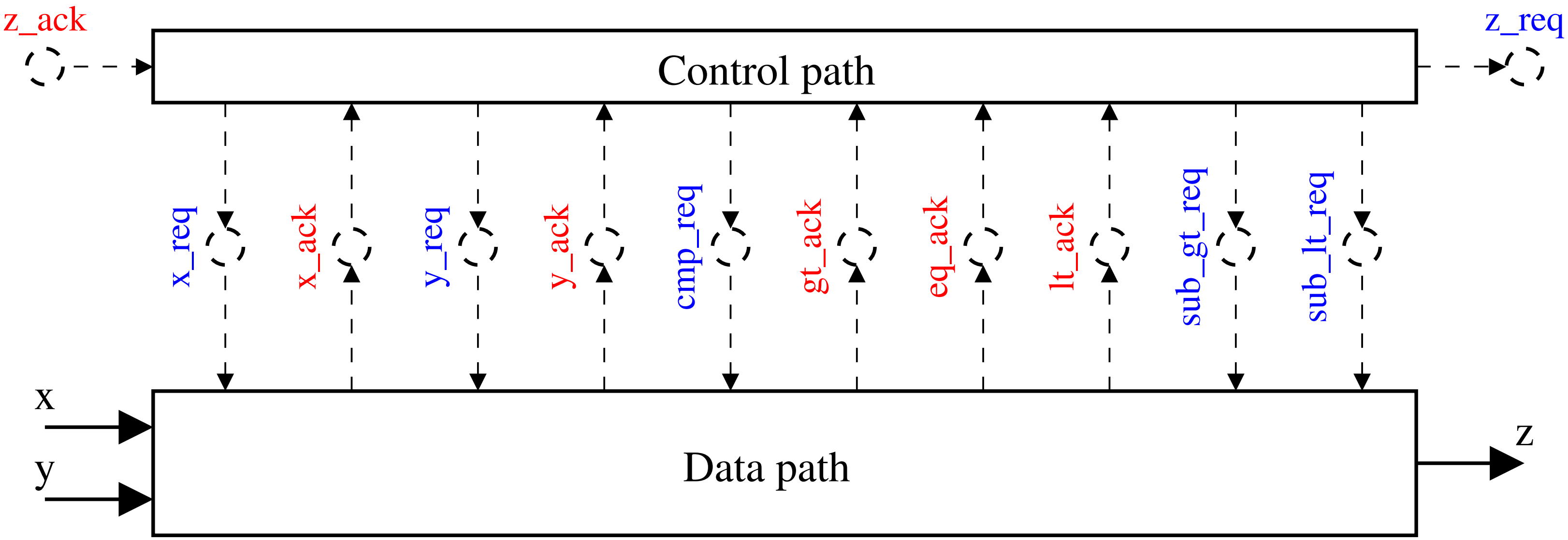
Labelled PN for GCD control path



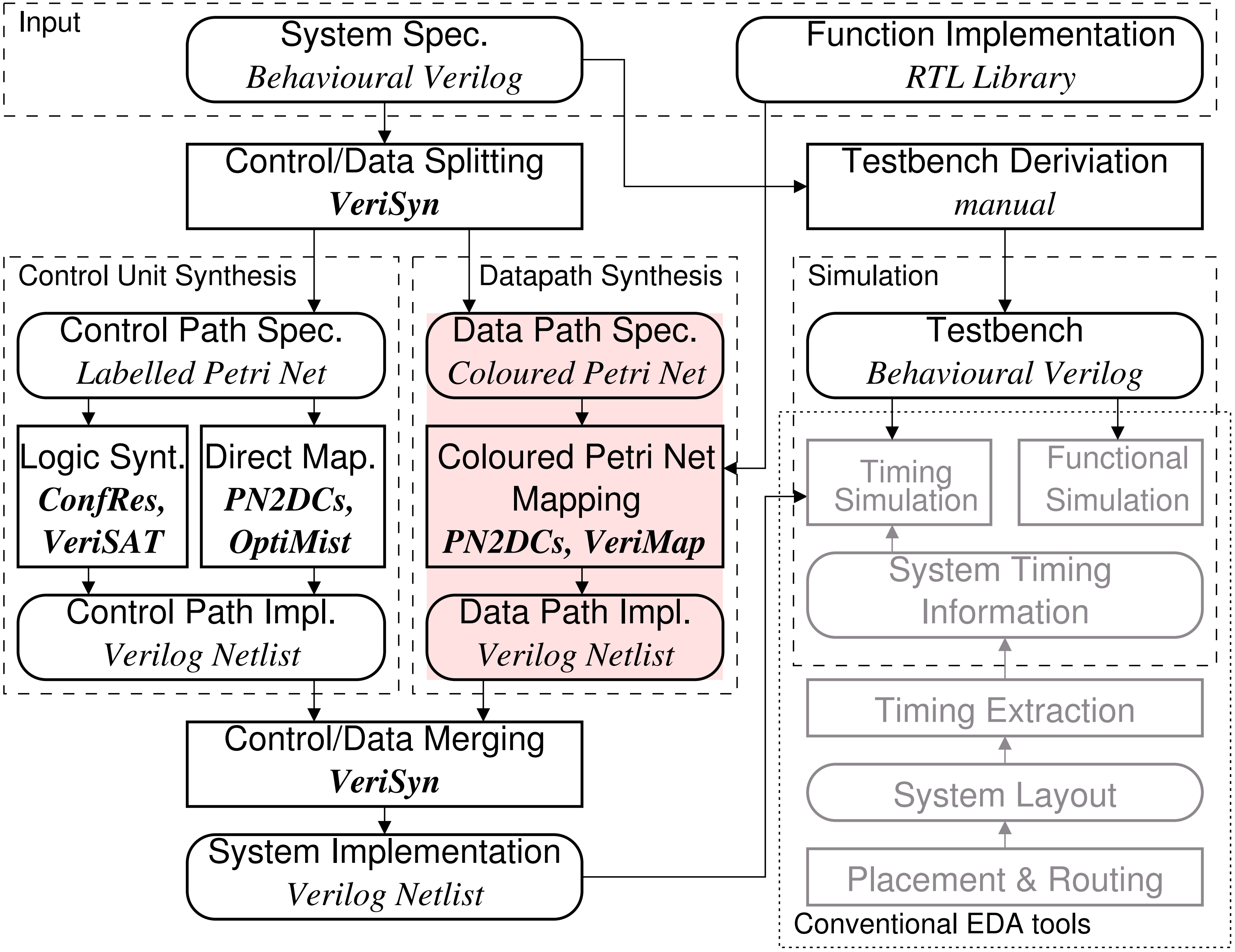
Coloured PN for GCD data path



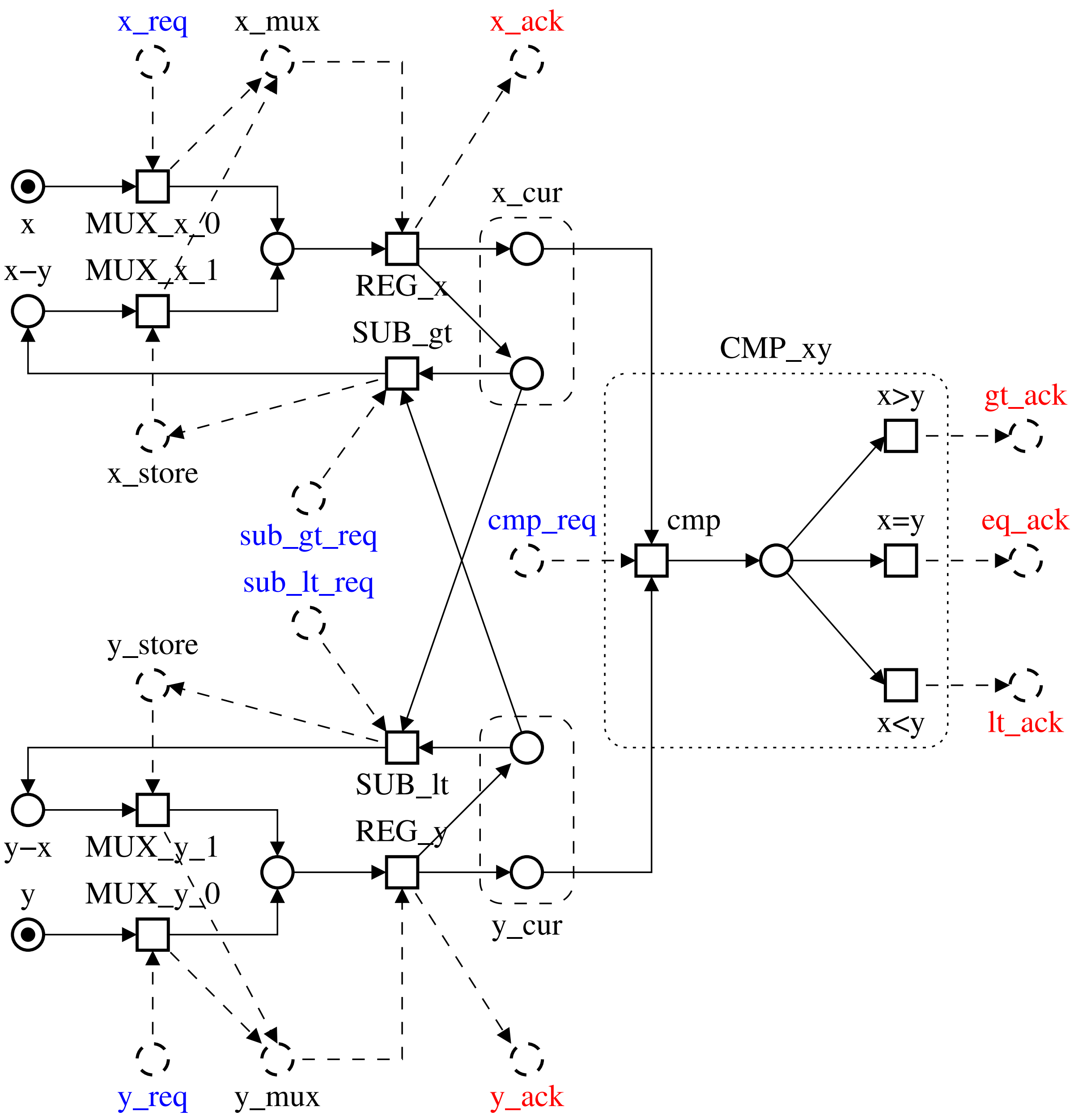
Control-data path interface



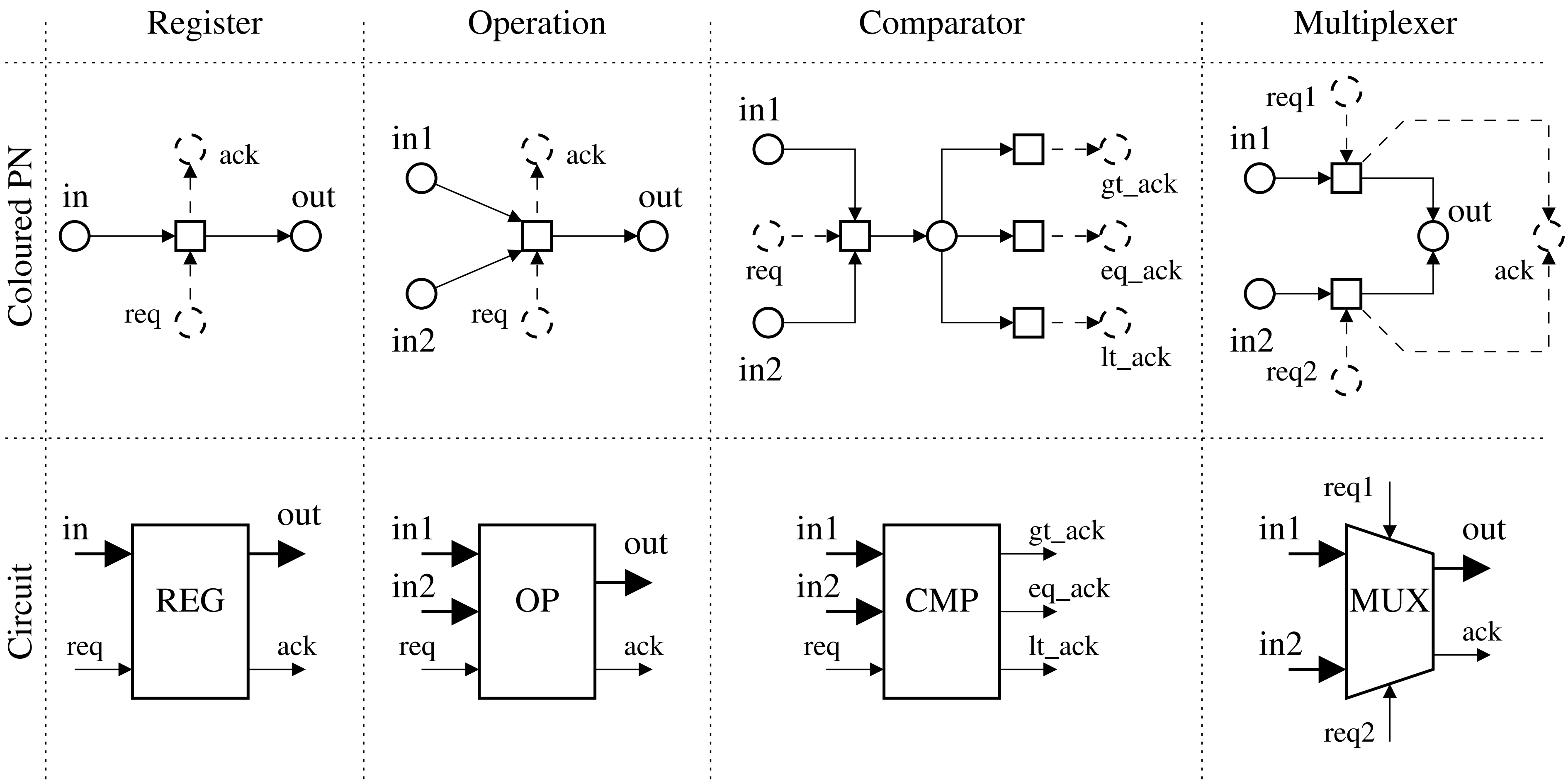
Synthesis of data path



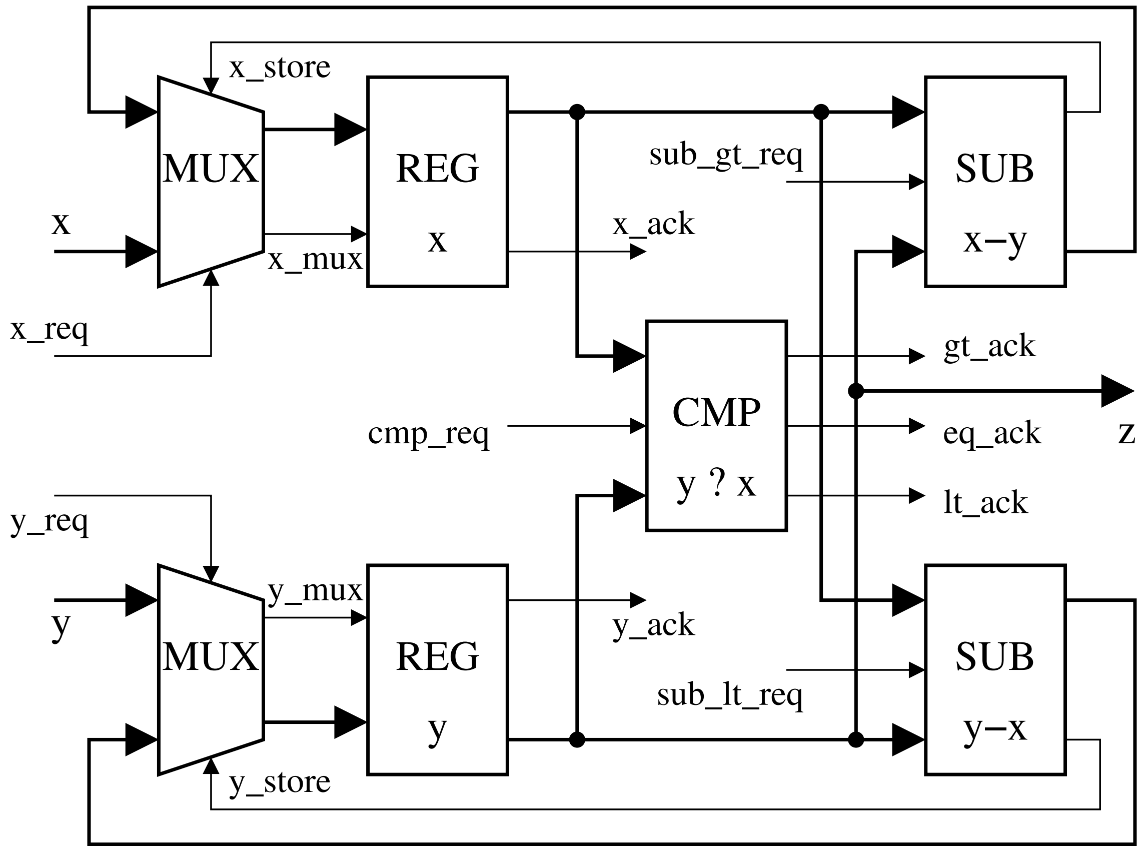
Coloured PN for GCD



Mapping Coloured PN into circuit

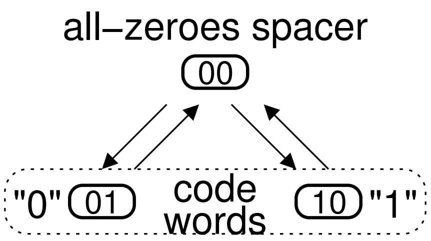


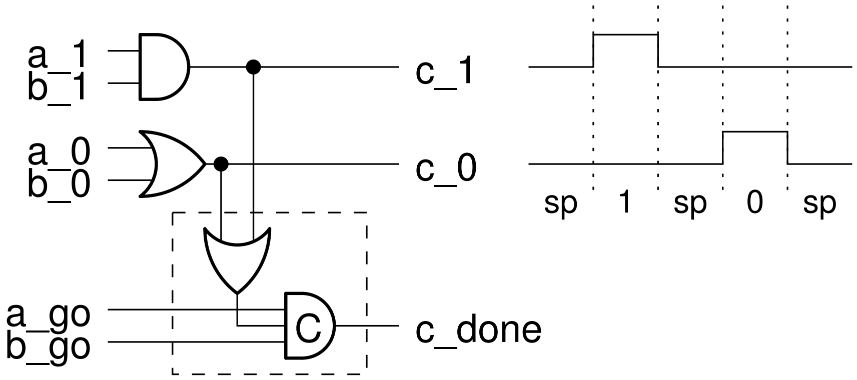
GCD data path schematic



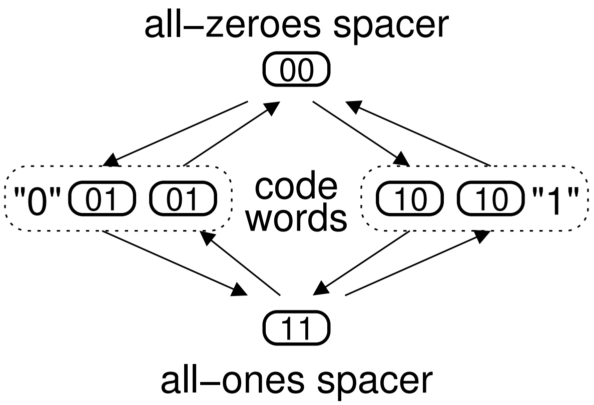
Dual-Rail Protocols

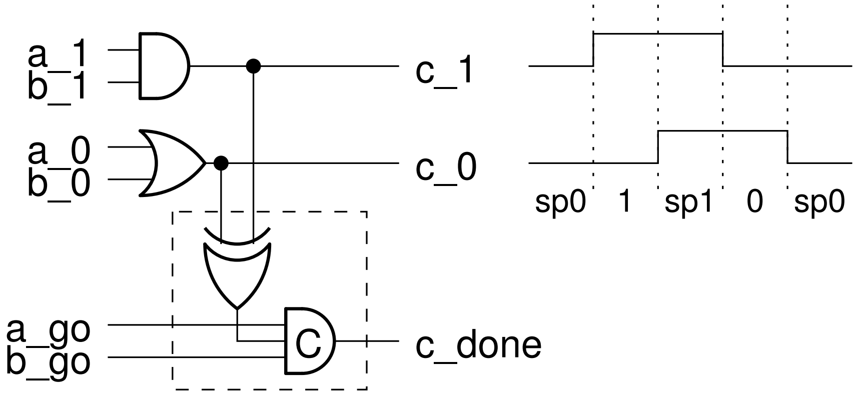
- Single spacer dual-rail protocol





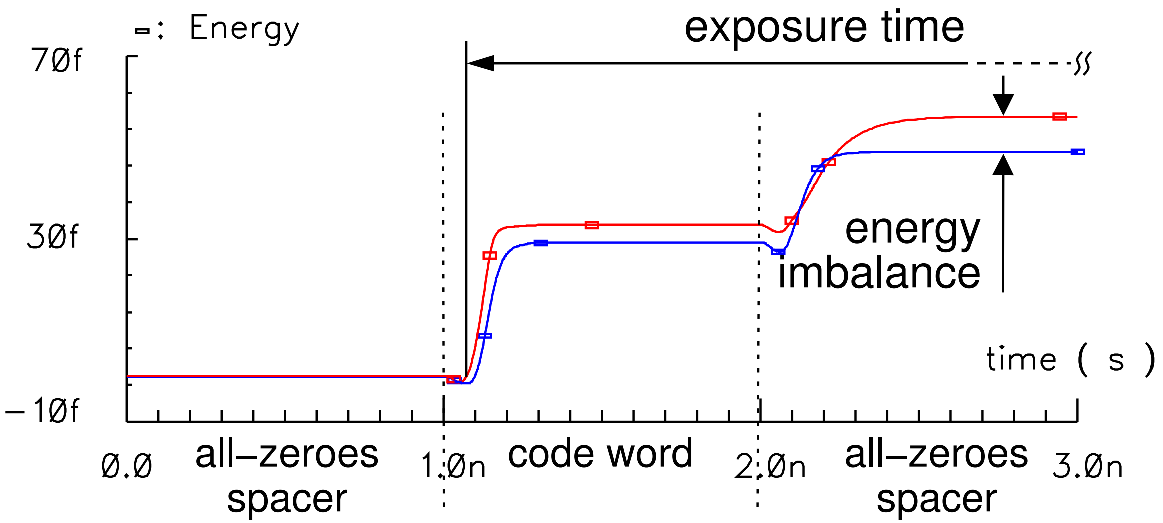
- Alternating spacer dual-rail protocol



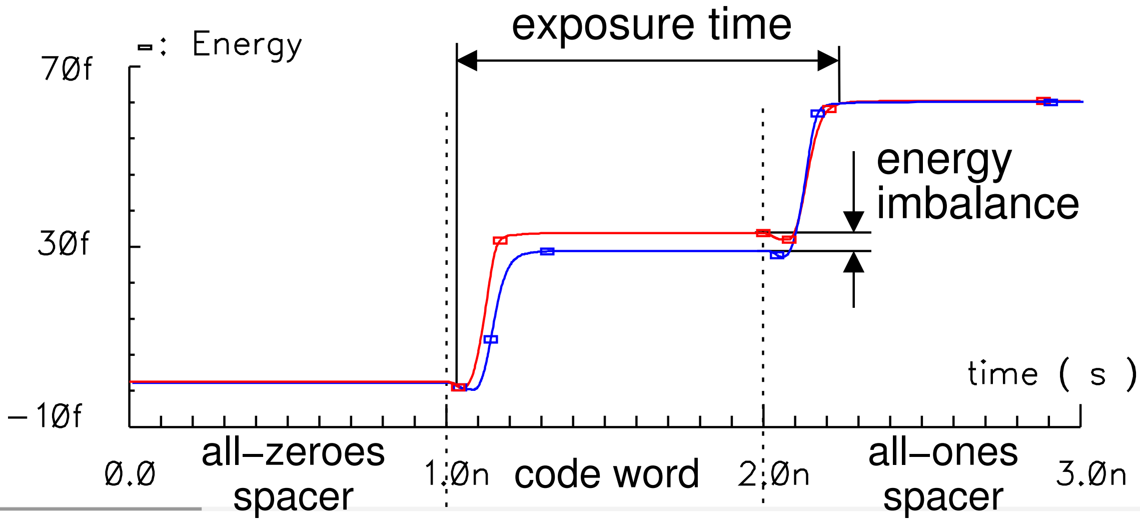


Energy imbalance and exposure time

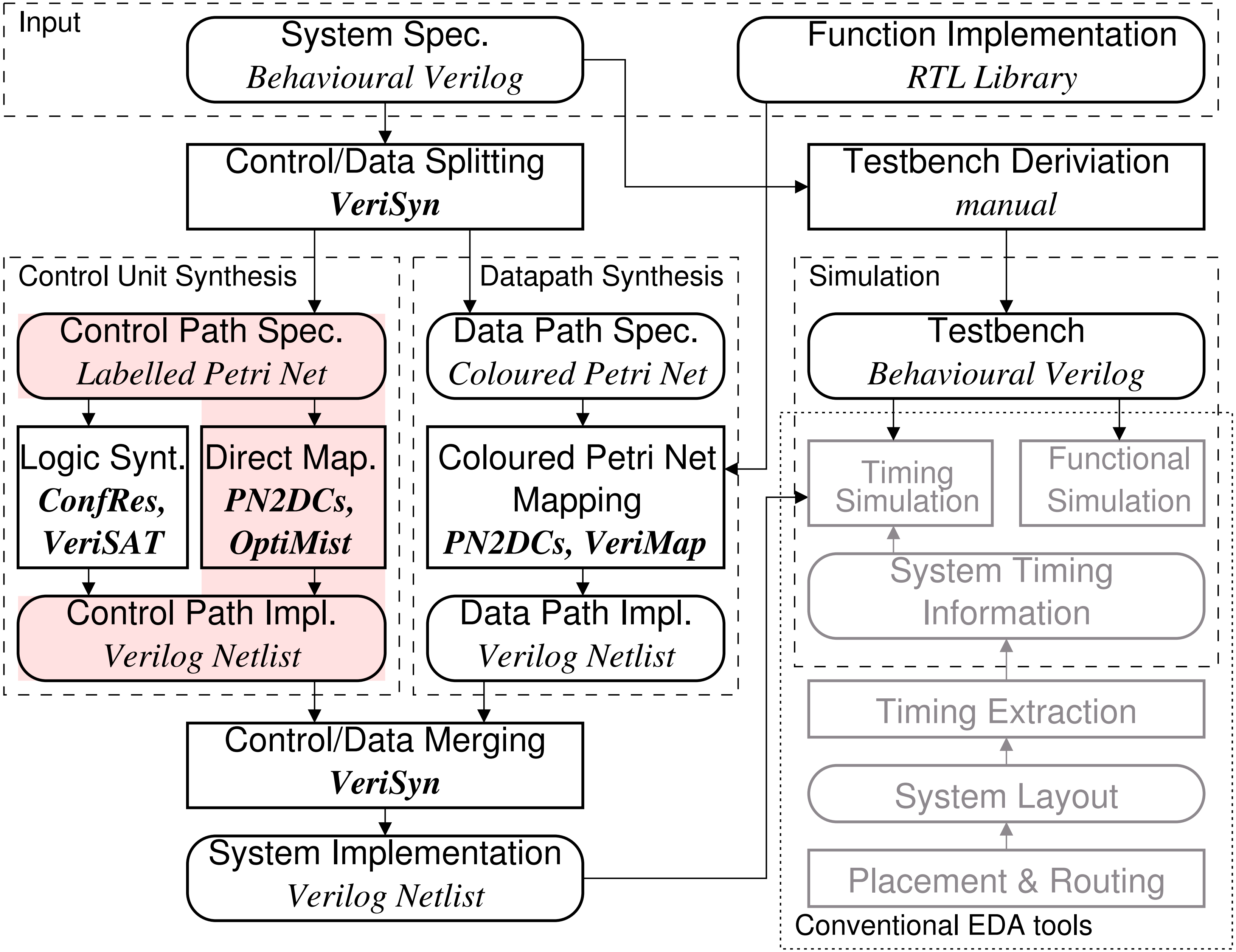
- Single spacer protocol in 2-input dual-rail AND gate



- Alternating spacer protocol in 2-input dual-rail AND gate



Direct mapping of control path



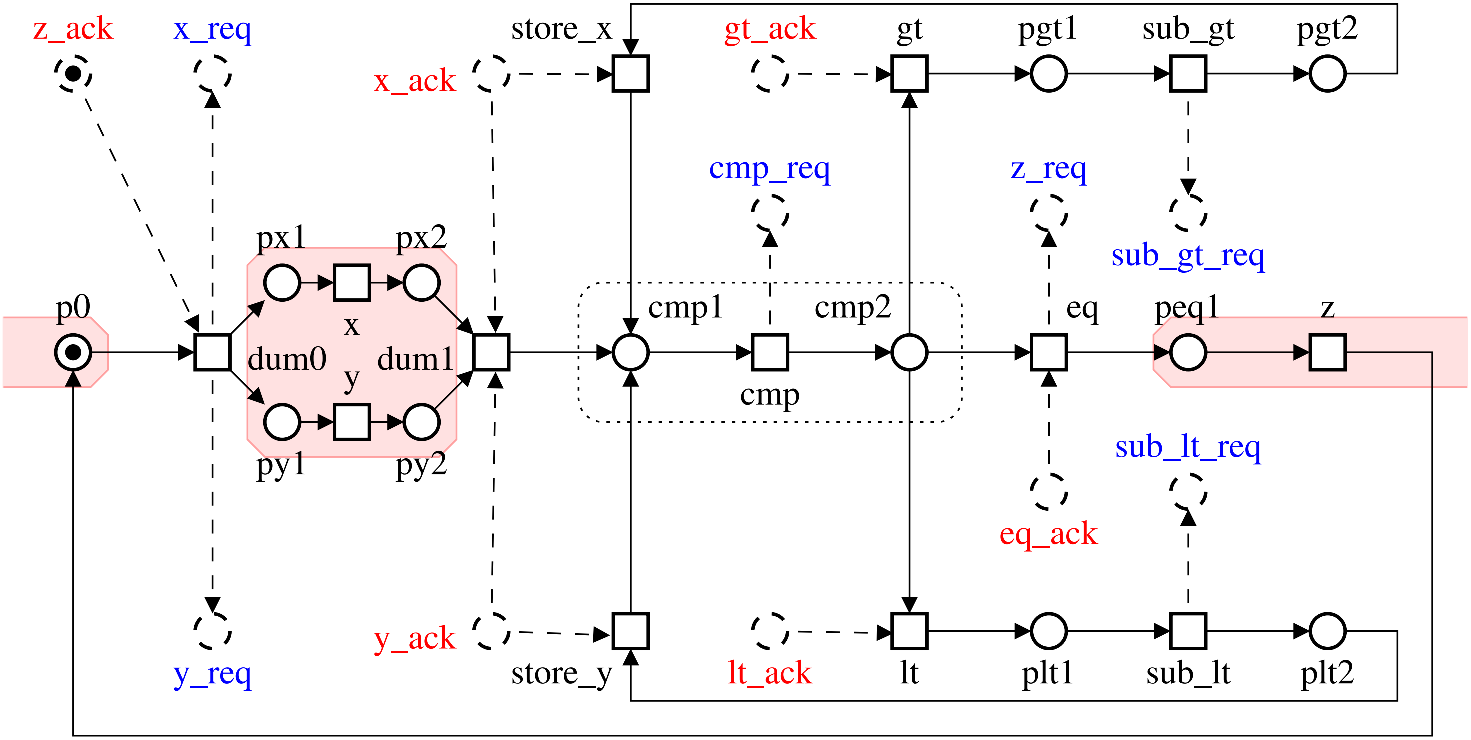
Direct mapping of control path

- Two approaches to direct mapping of control path (implemented in **PN2DCs** and **OptiMist** tools)

Tool	**PN2DCs**	**OptiMist**
Specification	Labeled PN	Signal Transition Graph (STG)
Abstraction level	Abstract	Detailed
Advantages	Smaller size	Lower latency

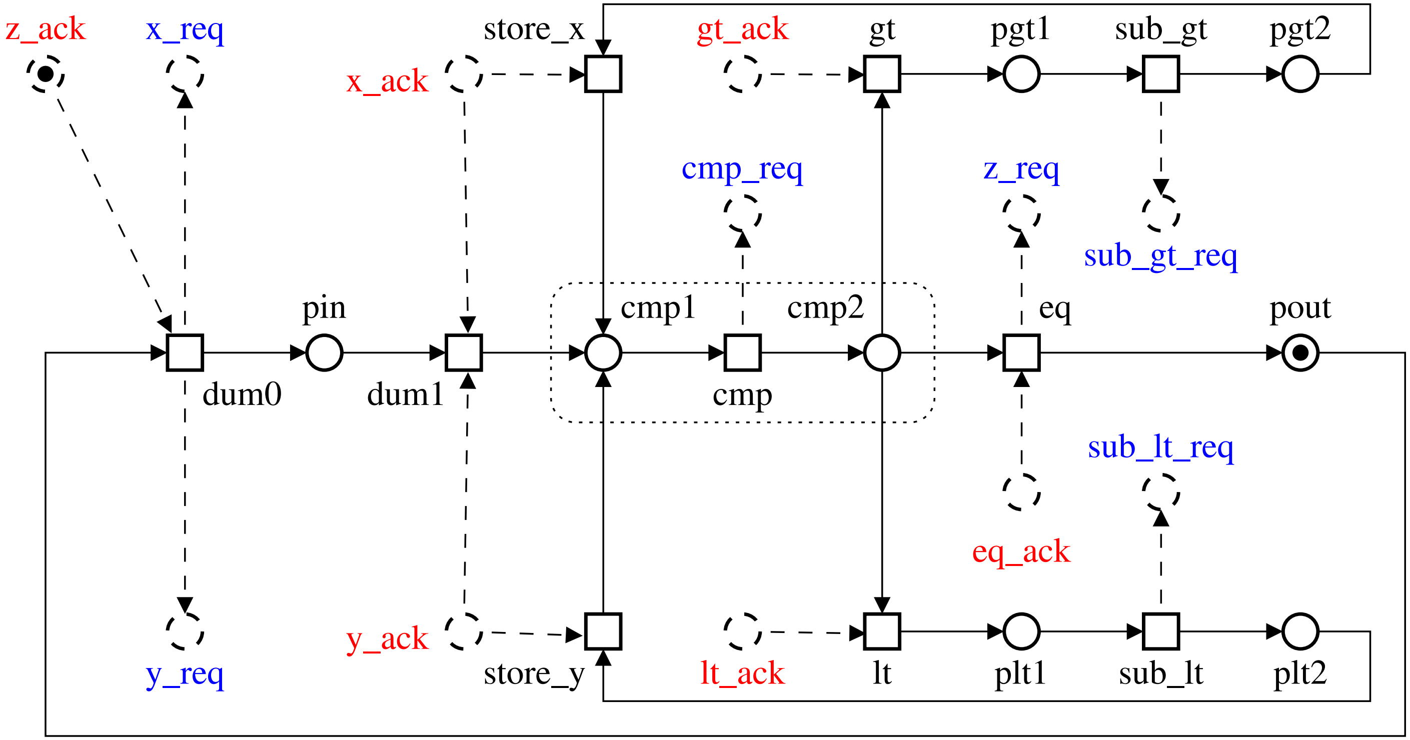
Labelled PN for GCD control path

- Labelled PN for GCD control unit



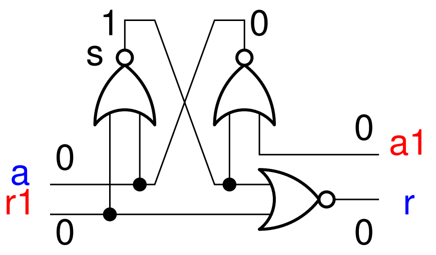
Labelled PN for GCD control path

- Optimised Labelled PN for GCD control unit

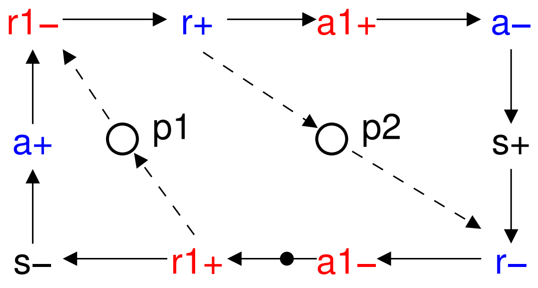


Mapping of Labelled PN places into DCs

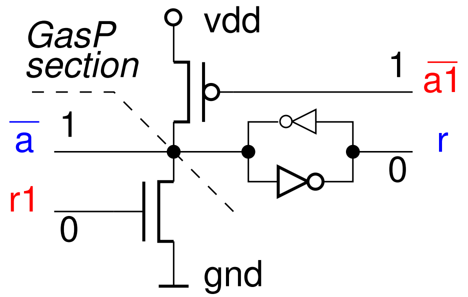
- David Cell (DC) - state holding element for one token



Gate-level DC

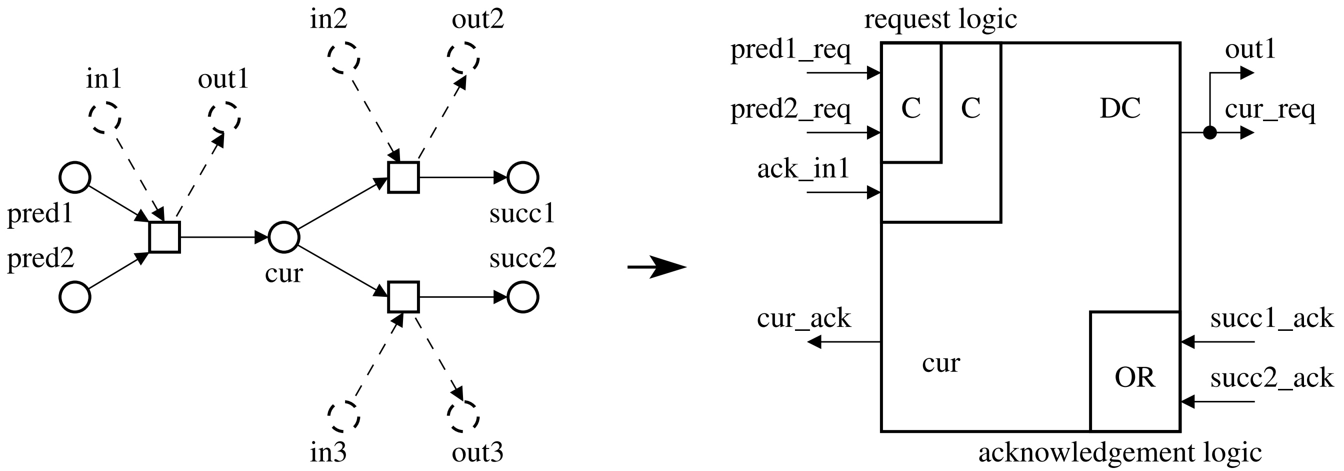


STG

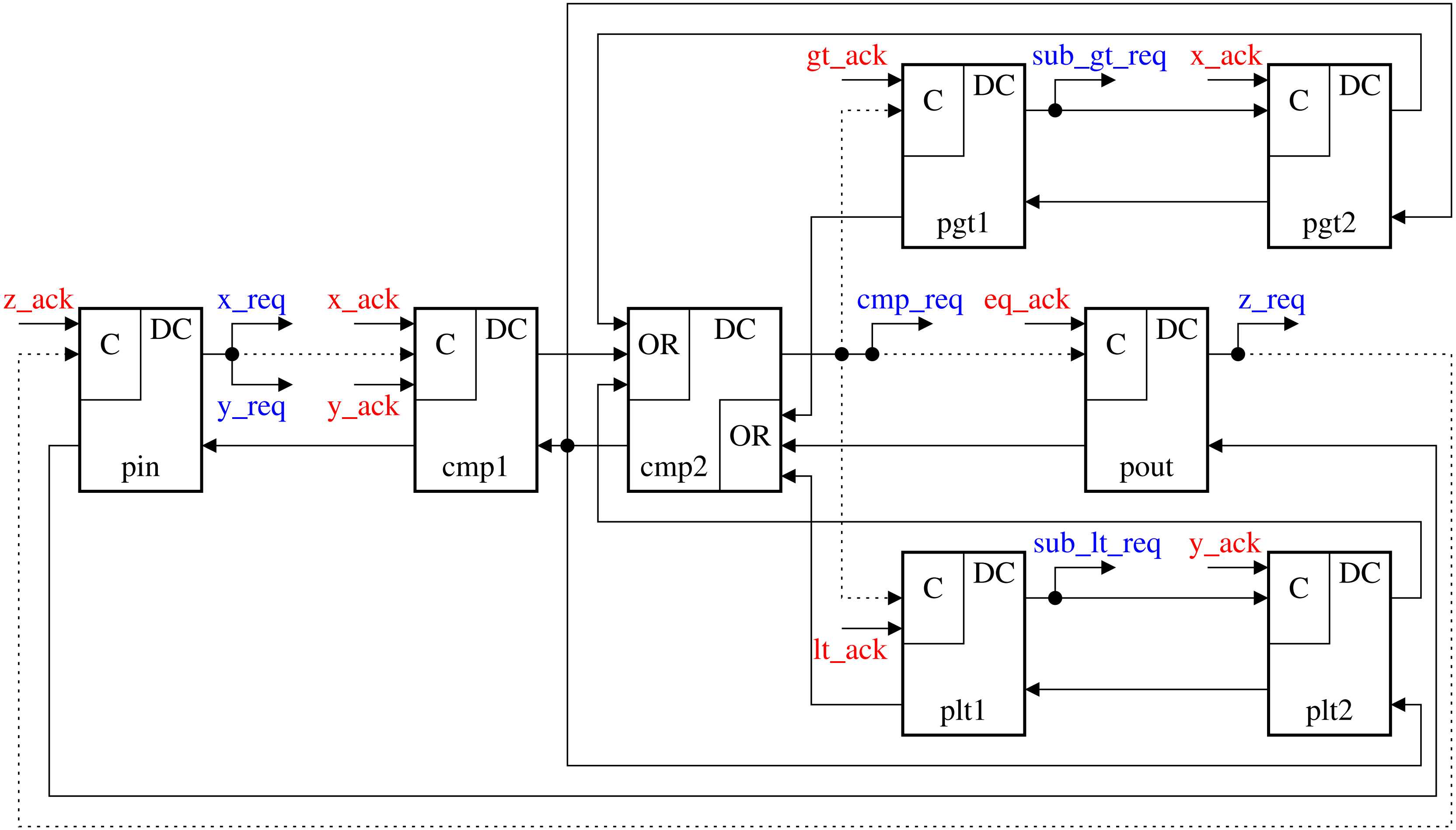


Transistor-level DC

- Mapping of a Labelled PN place into a DC

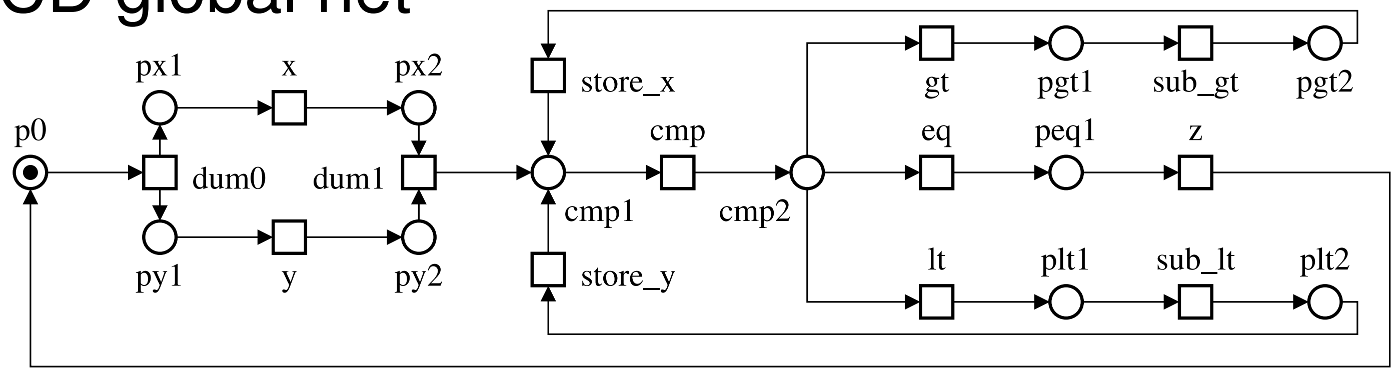


GCD control path schematic

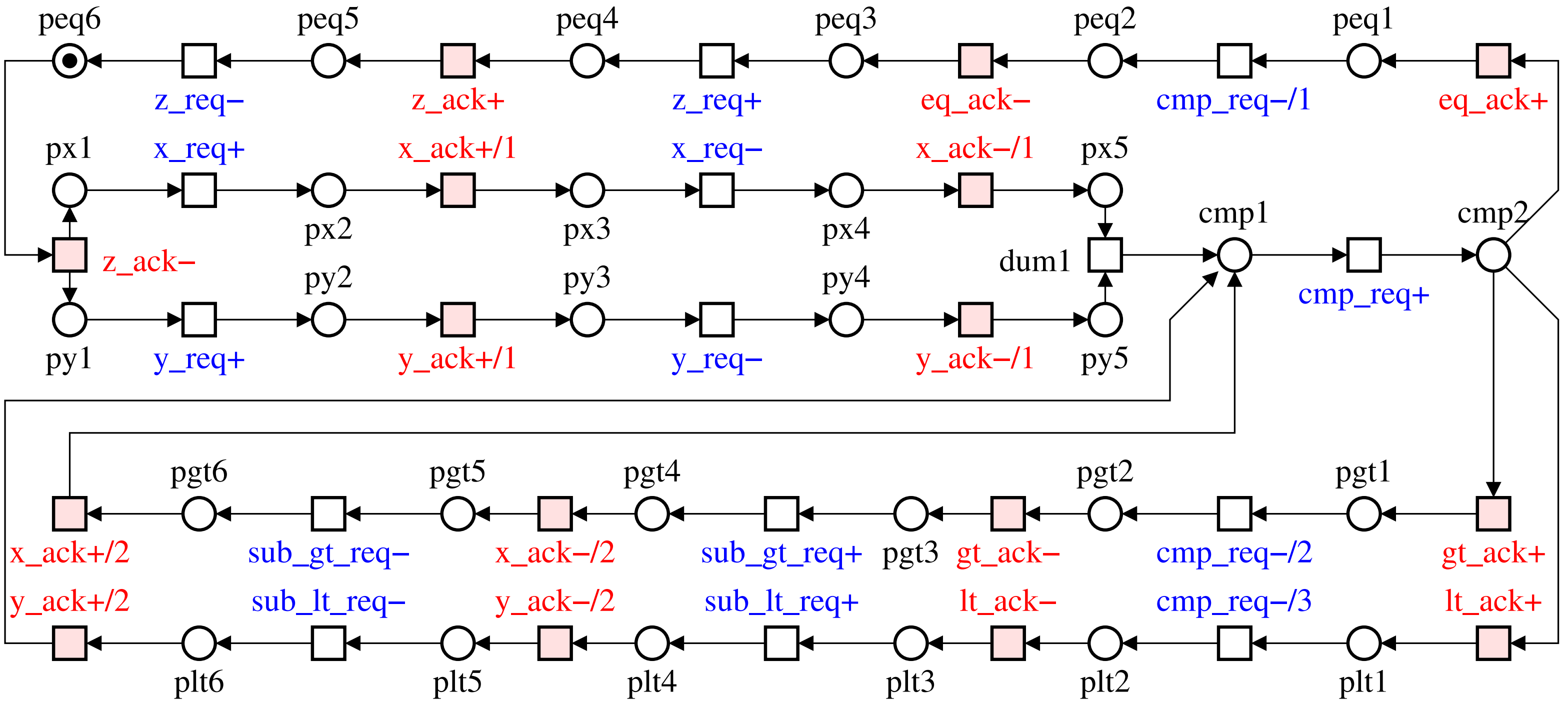


Deriving STG for GCD control path

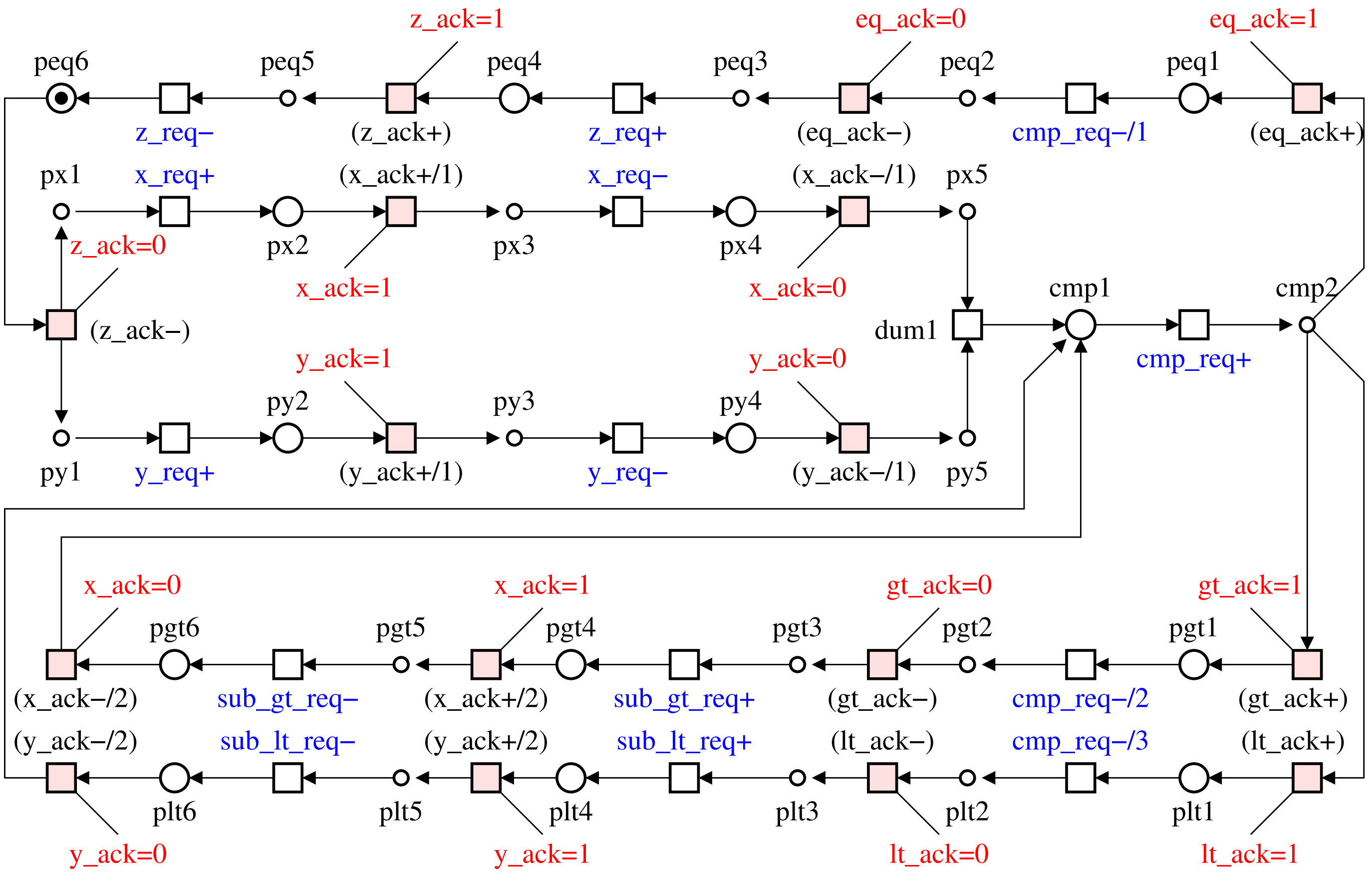
- GCD global net



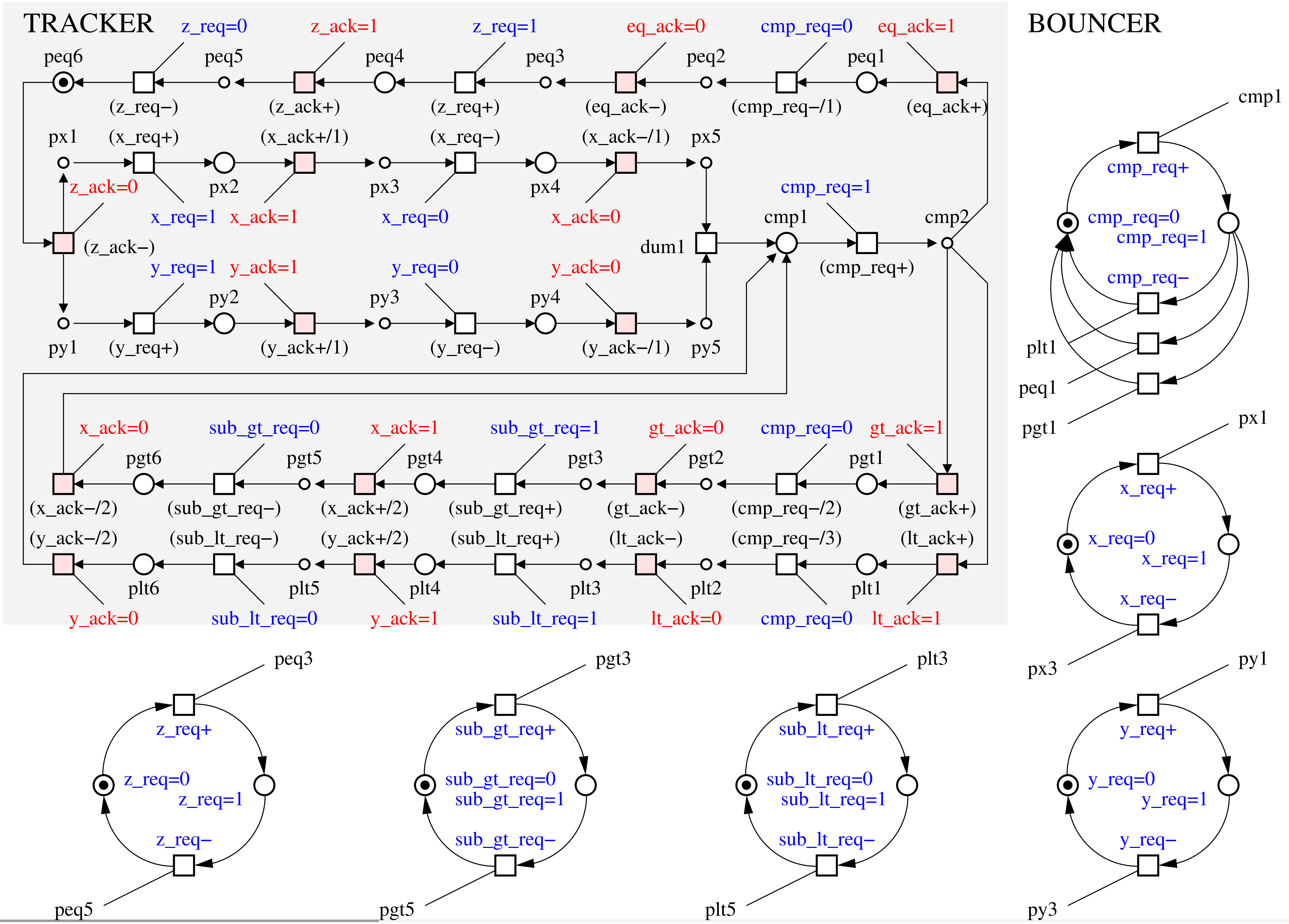
- Control unit STG (with explicit places)



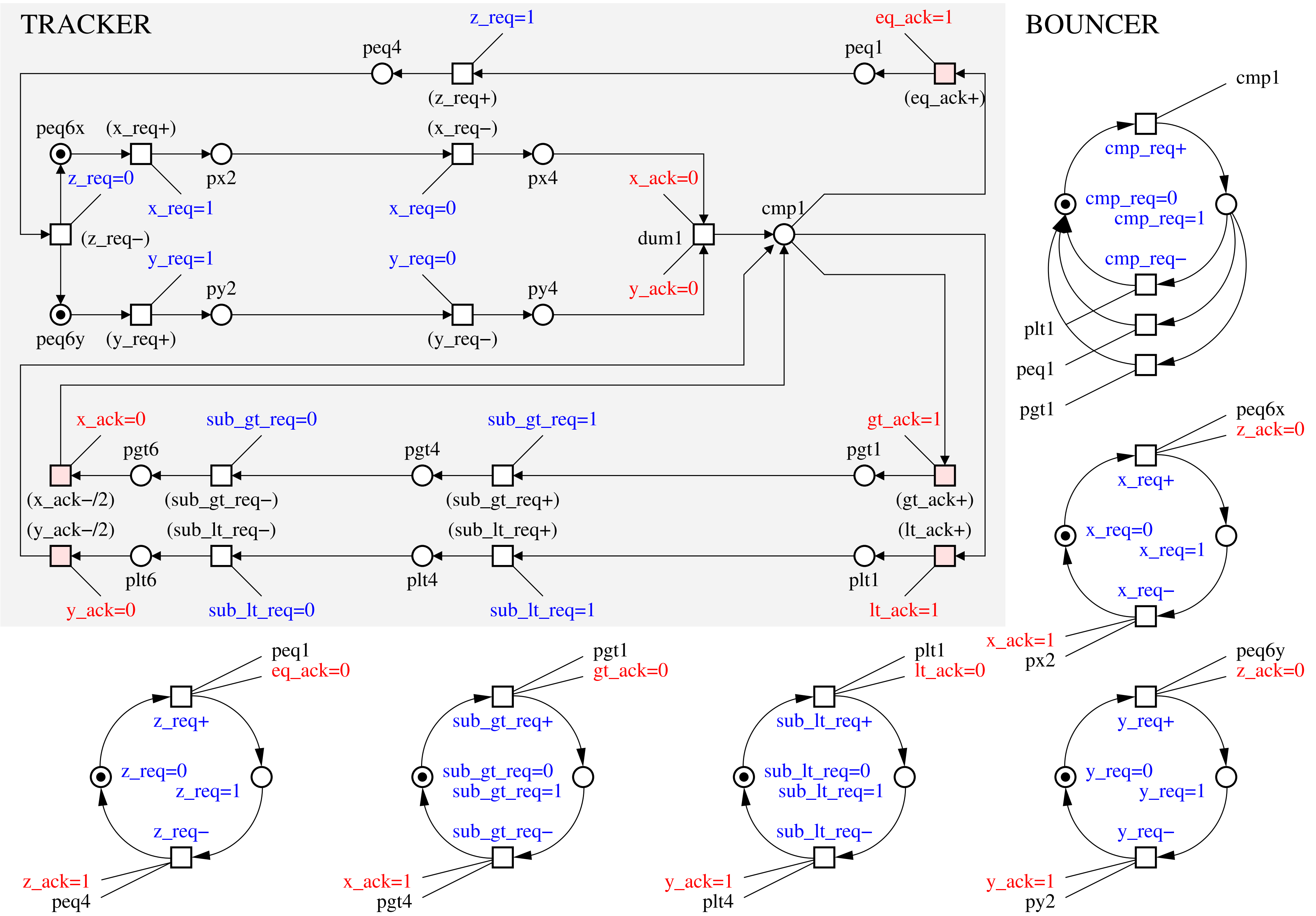
Device-environment splitting



Output exposure

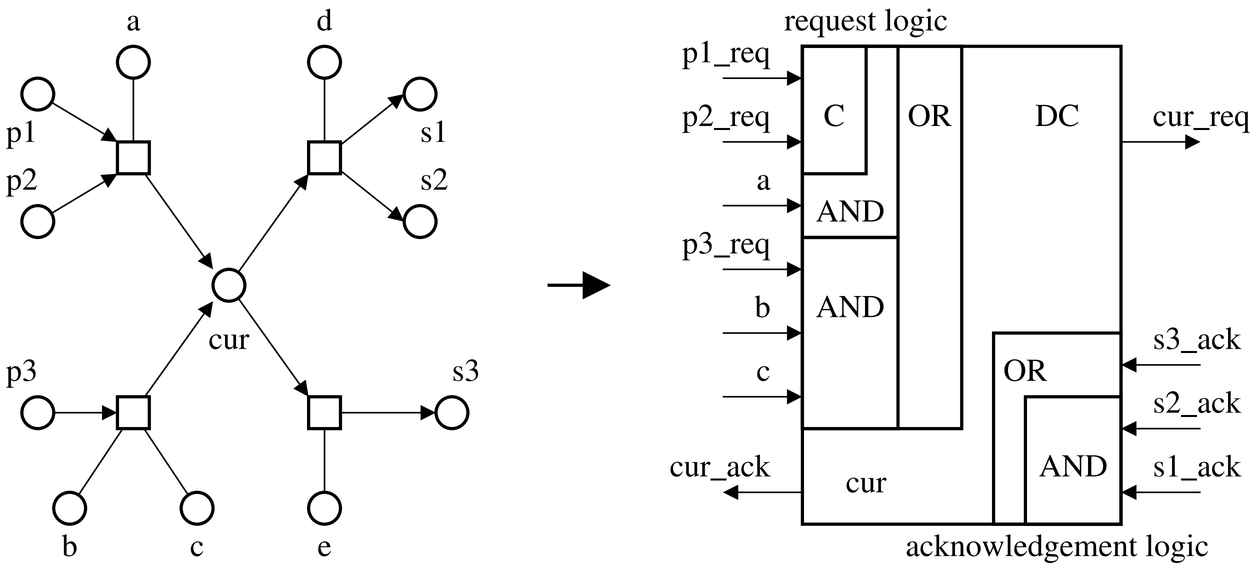


Optimisation

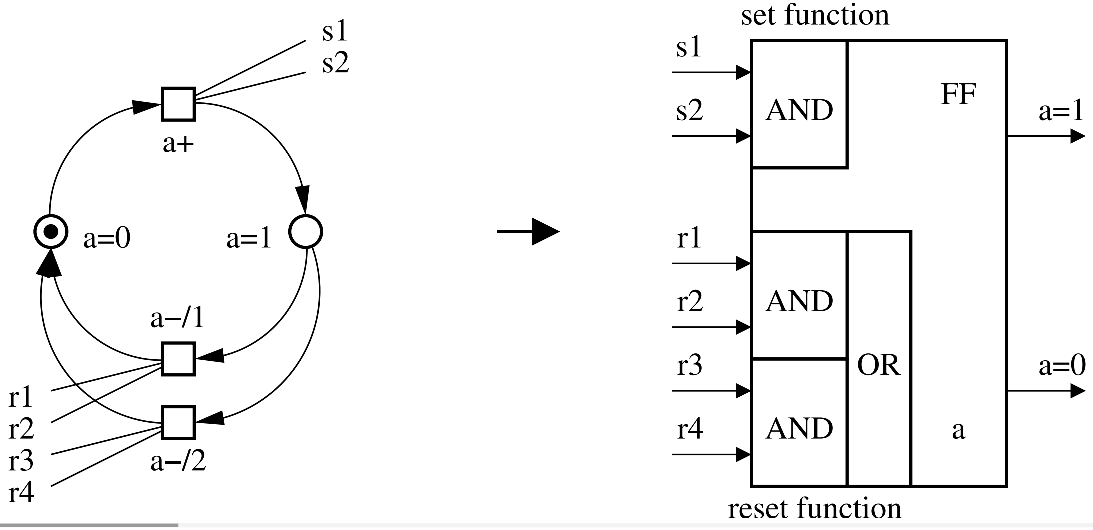


Mapping into David cells and flip-flops

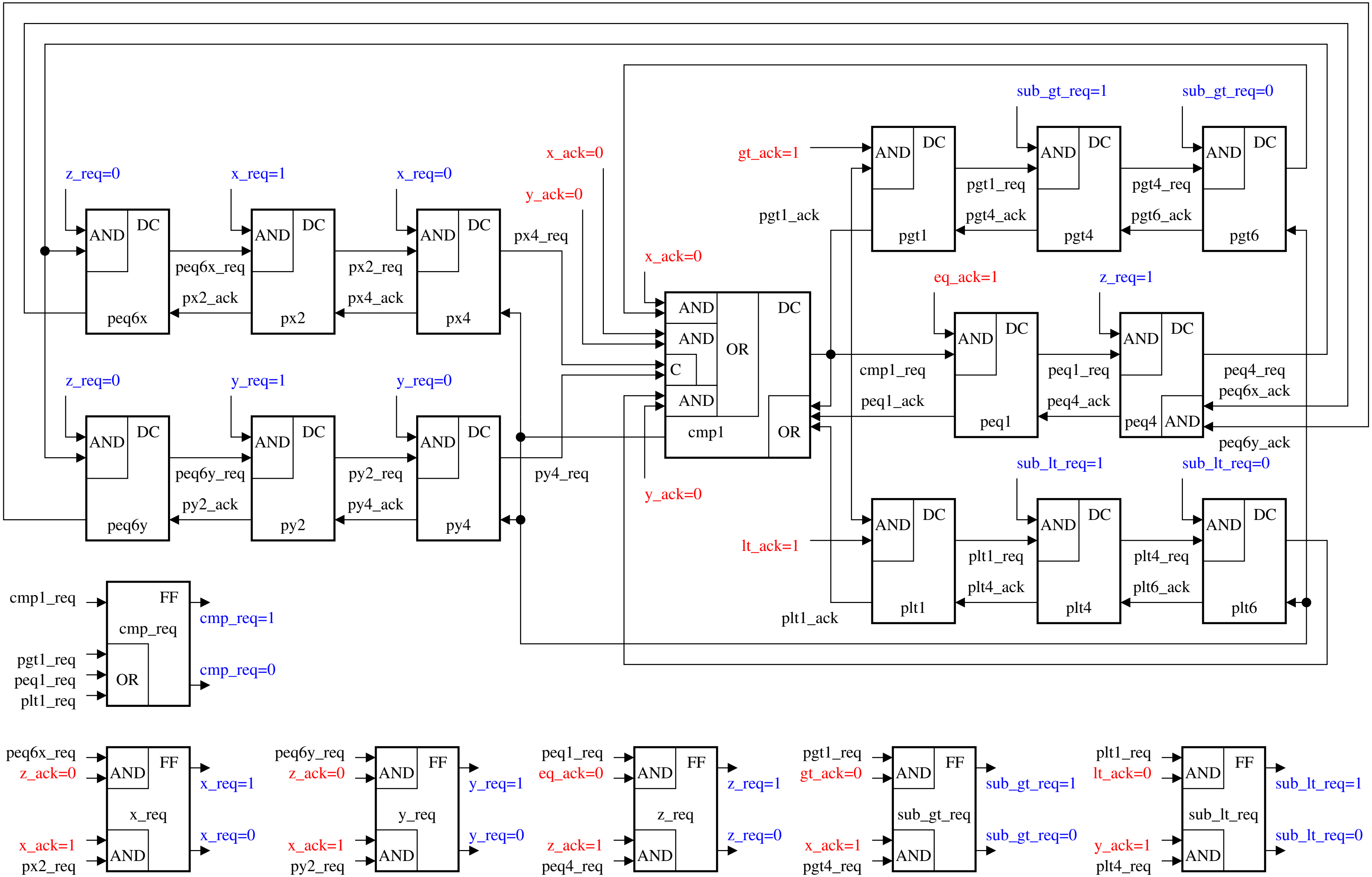
- Mapping tracker places into David cells



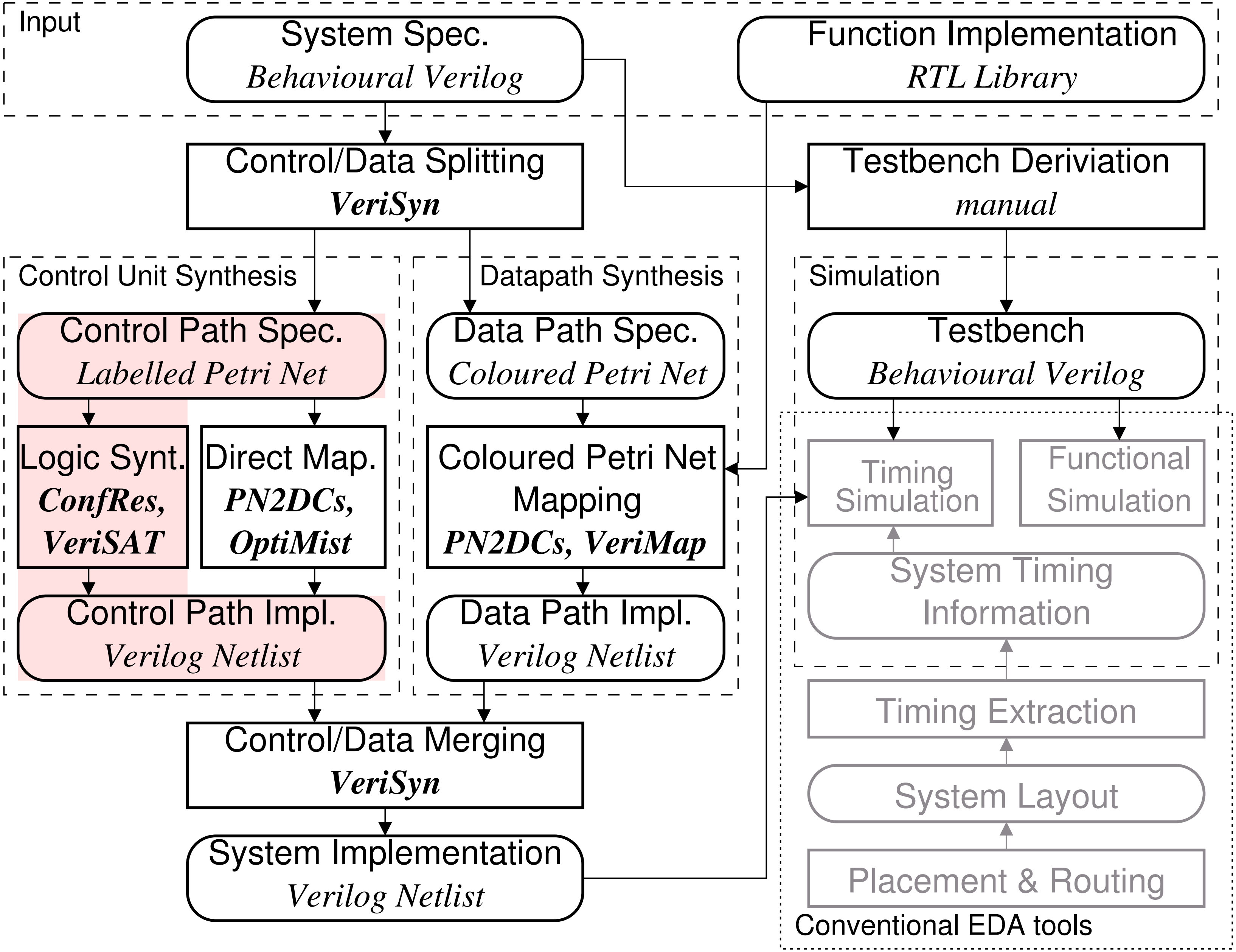
- Mapping elementary cycles into flip-flops



Low-latency GCD control circuit

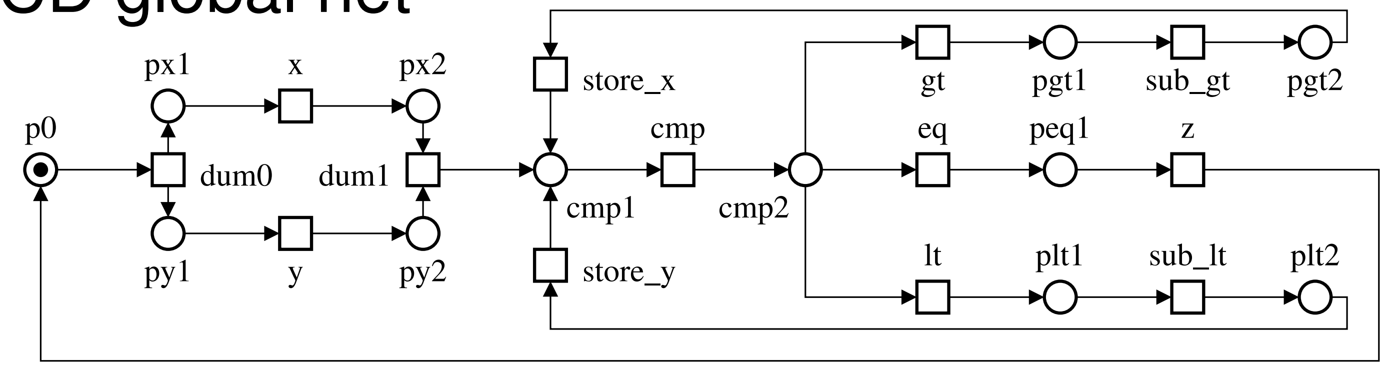


Logic synthesis of control unit

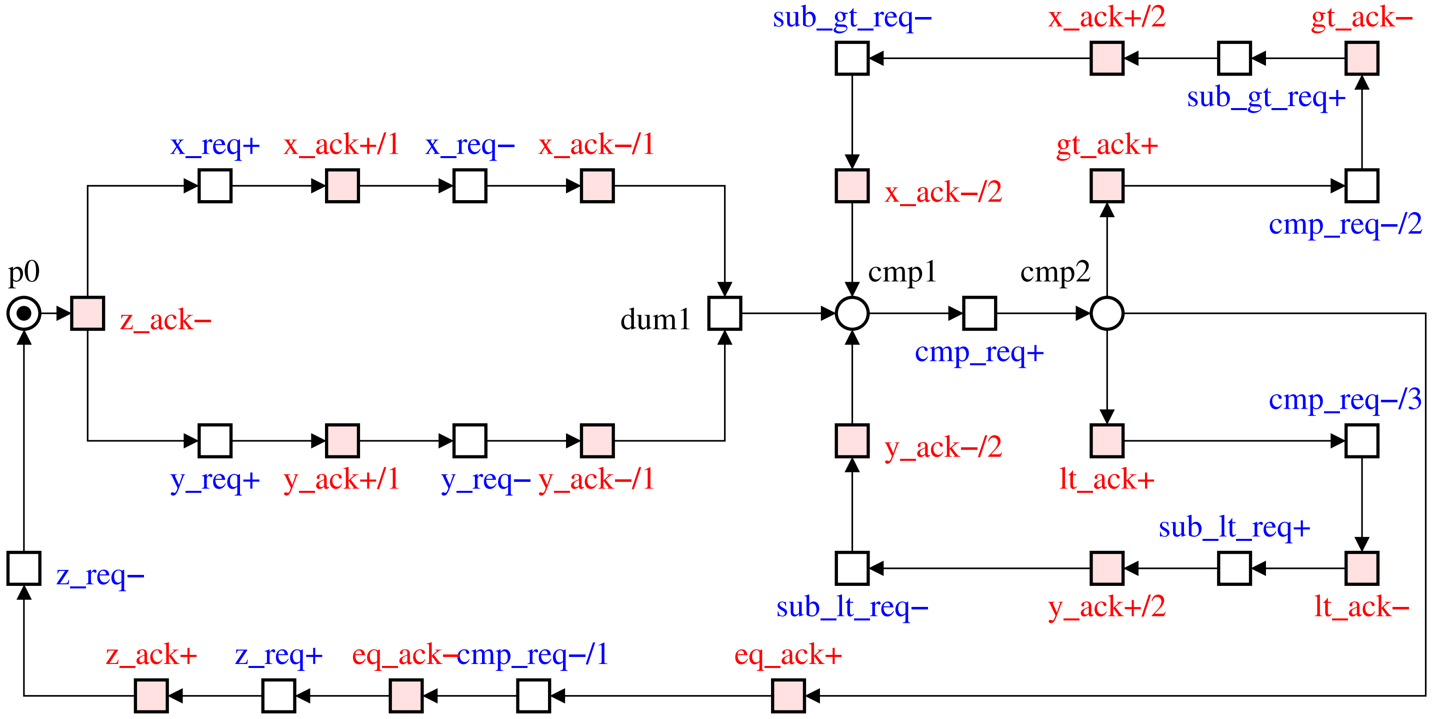


GCD control unit STG

- GCD global net

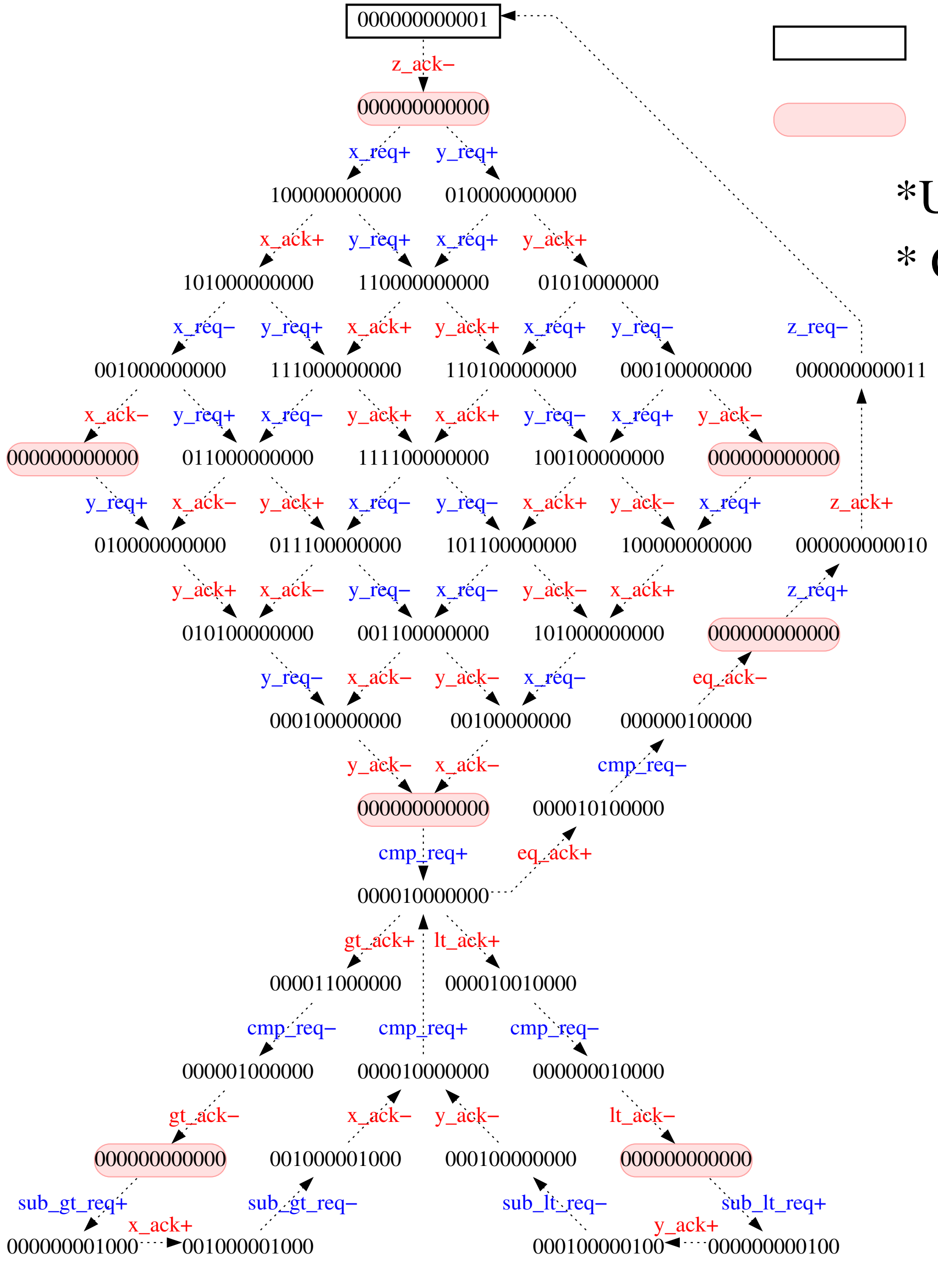


- Control unit STG (with implicit places)



Reachability graph and CSC

< x_req, y_req, x_ack, y_ack, cmp_req, gt_ack, eq_ack, lt_ack, sub_gt_req, sub_lt_req, z_req, z_ack >



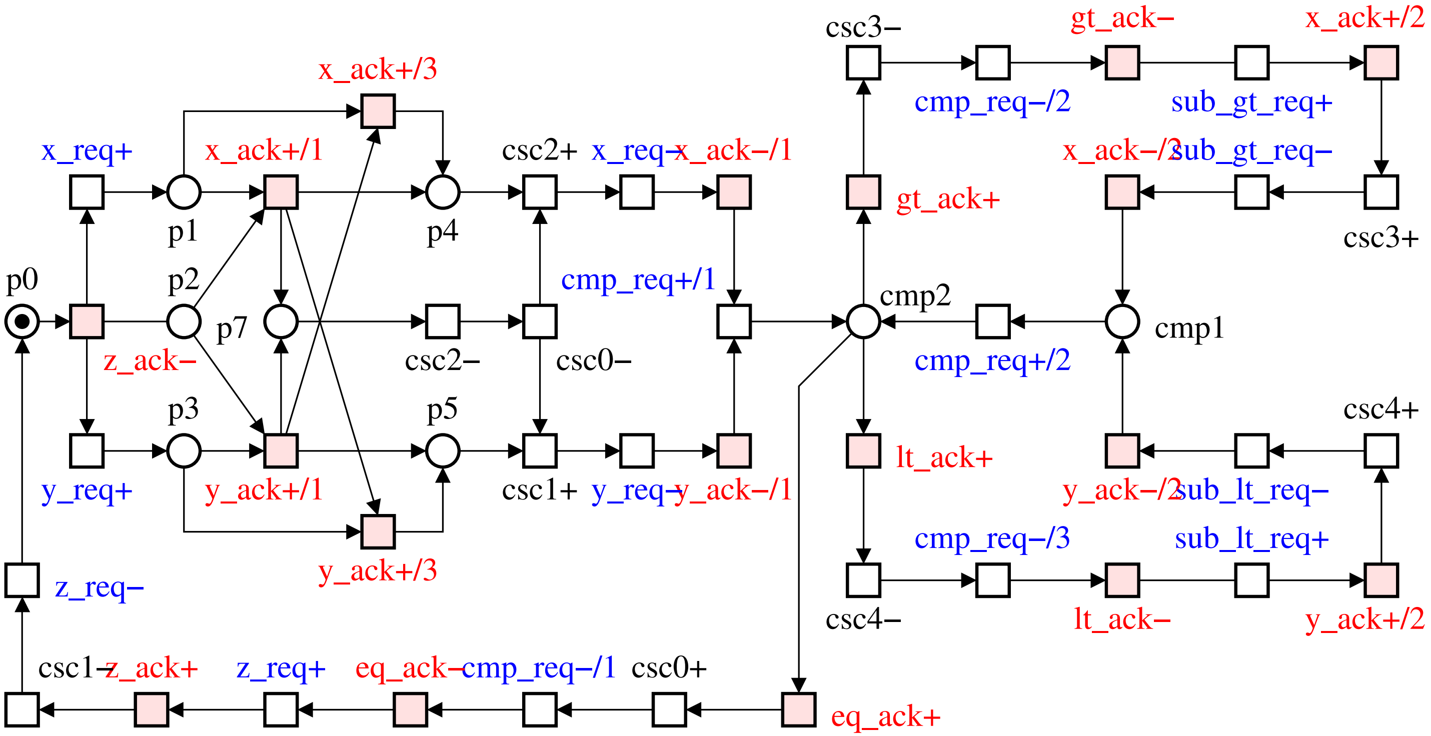
□ – initial state

▢ – conflicting states

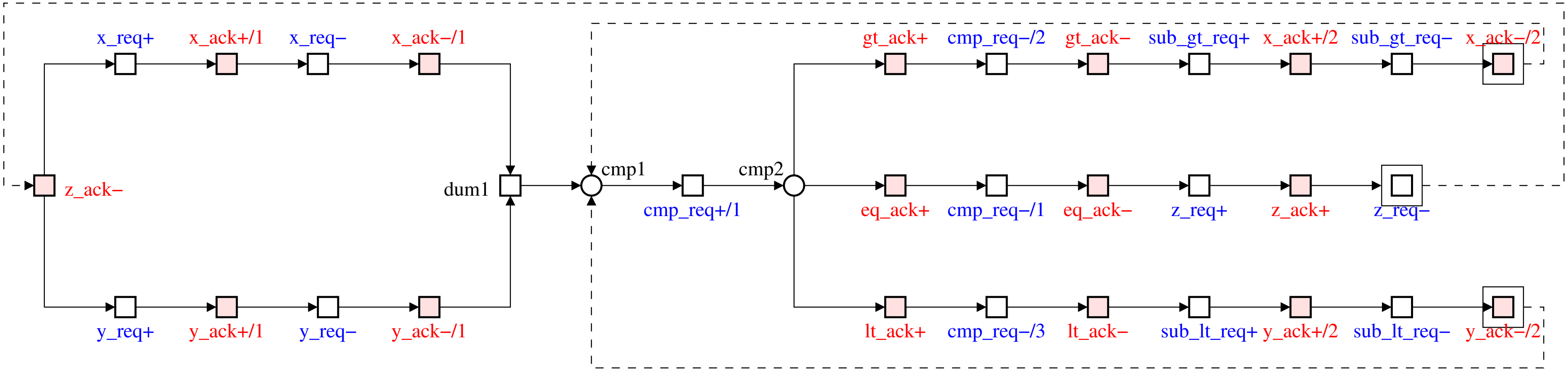
*Unique State Coding (USC)

* Complete State Coding (CSC)

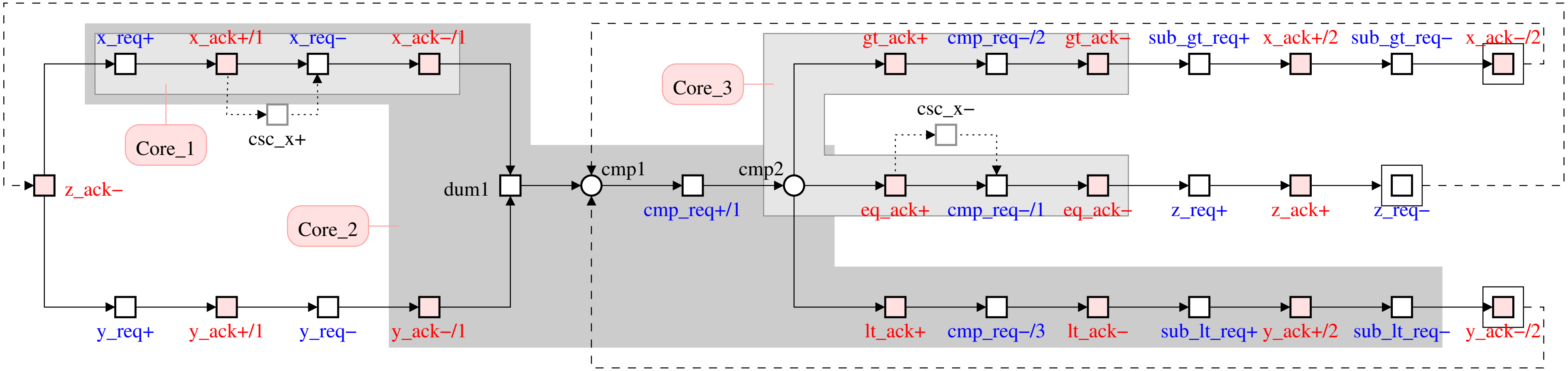
Automatic CSC solution



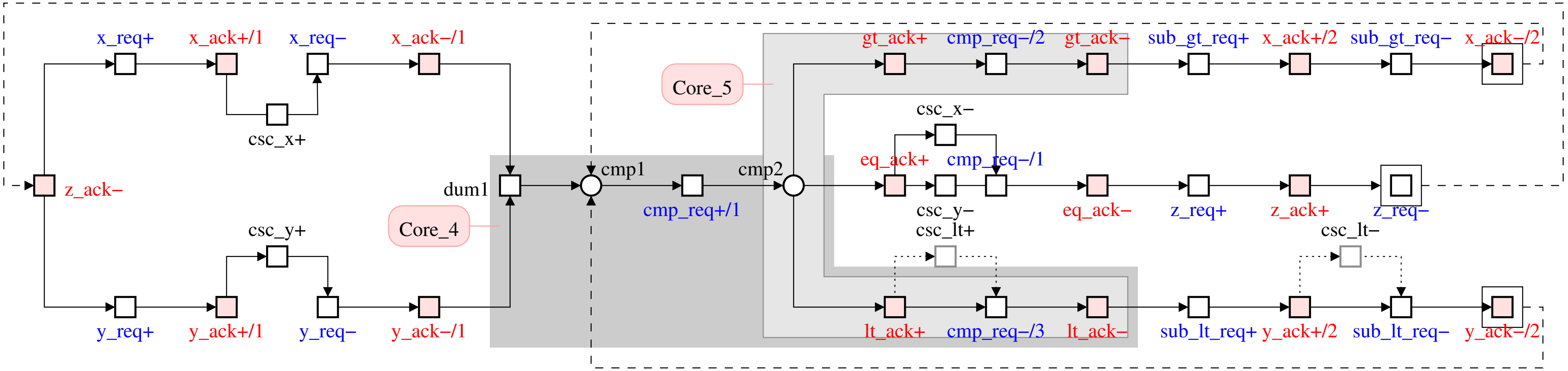
Semi-automatic CSC solution



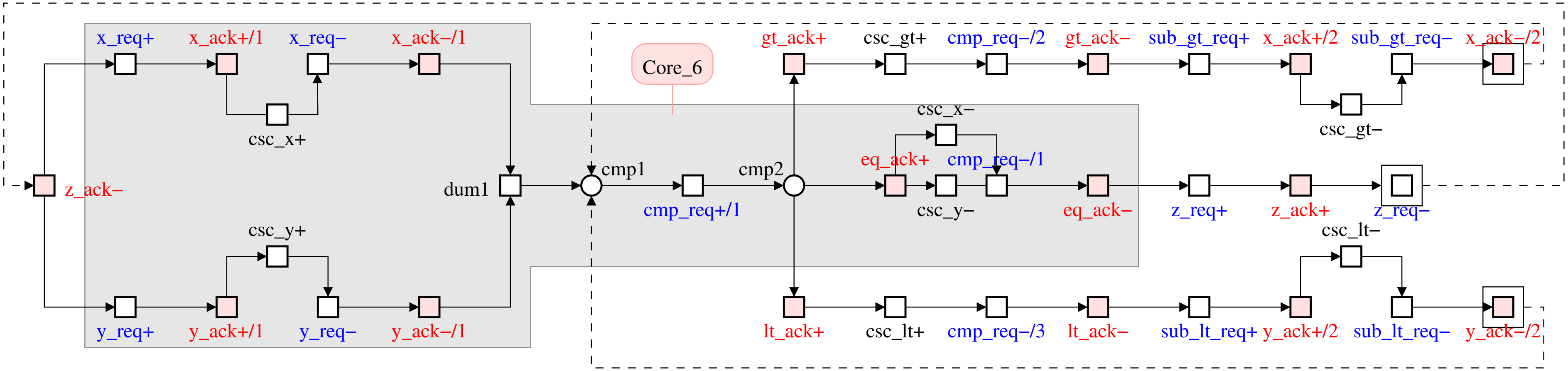
Semi-automatic CSC solution



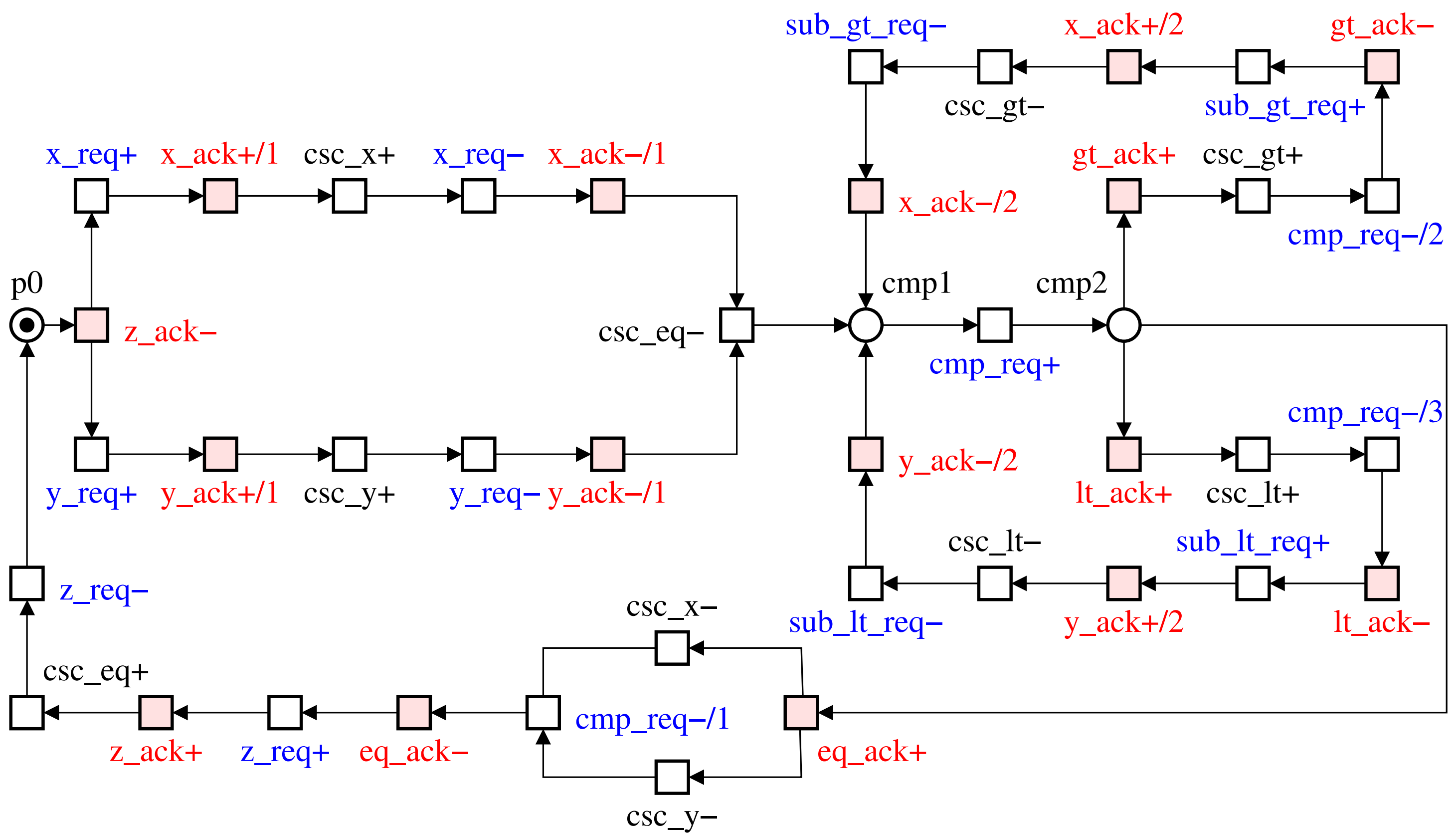
Semi-automatic CSC solution



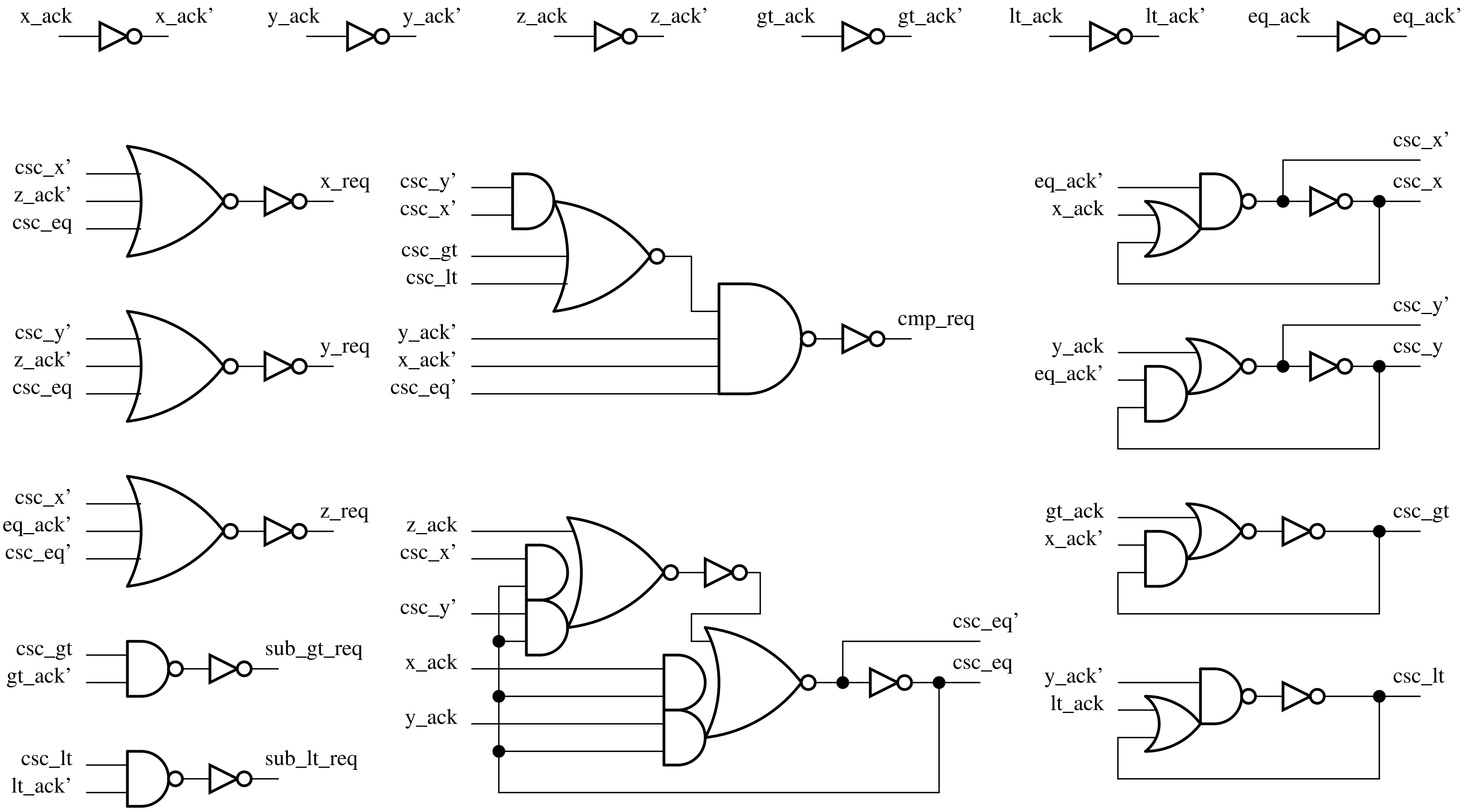
Semi-automatic CSC solution



Result of semi-automatic CSC solution



Complex gate impl. of GCD controller



Comparison: Control path

method name		circuit size (trans)	estimated latency (neg.gate)	comp. time (sec)
direct mapping	from Labeled PNs	55	4	<1
	from STGs	174	3	<1
logic synth.	automatic CSC solution	116	9	18
	semi-automatic CSC solution	120	5	2

Comparison: GCD circuit

Tool	Area	Speed (ns)		computation
	(μm^2)	x=y	x=12, y=16	time (s)
Balsa	119,647	21	188	< 1
PN2DCs	100,489	14	109	< 1
Improvement	16%	33%	42%	0

Conclusions

- Coherent asynchronous circuit design flow
  - Initial spec is in behavioural Verilog form
  - Petri nets intermediate circuit representation
  - Interface to conventional EDA tools for place-and-route and simulation
- Control path synthesis allows to trade off
  - Circuit size
  - Output latency
  - Computation time
- Data path synthesis
  - Direct mapping from Coloured PNs
  - Optional security features