Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

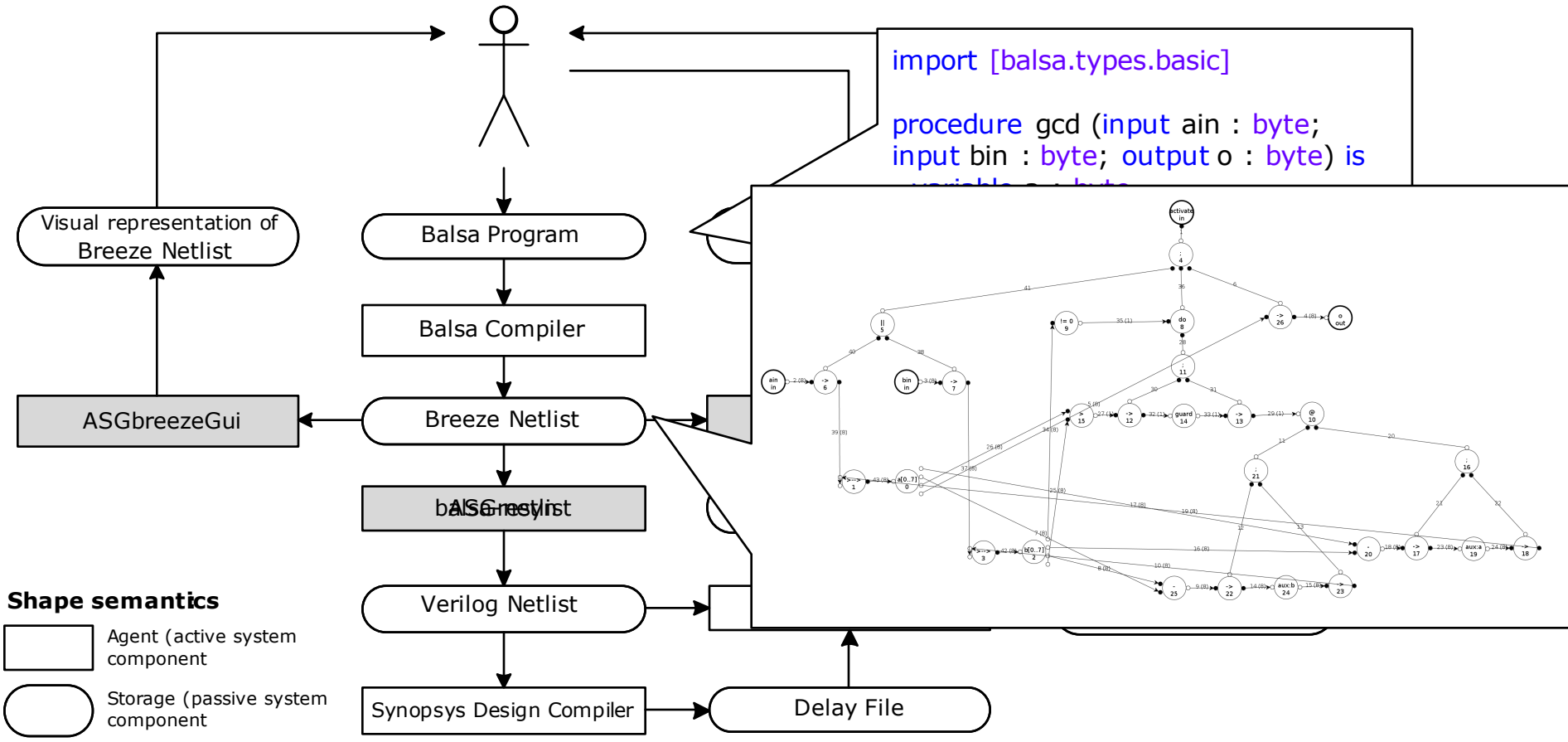
Optimising Bundled-Data Balsa Circuits

Norman Kluge, Ralf Wollowski

Agenda

1. **Overview**
2. Resynthesis idea
3. Resynthesis design flow
4. Results

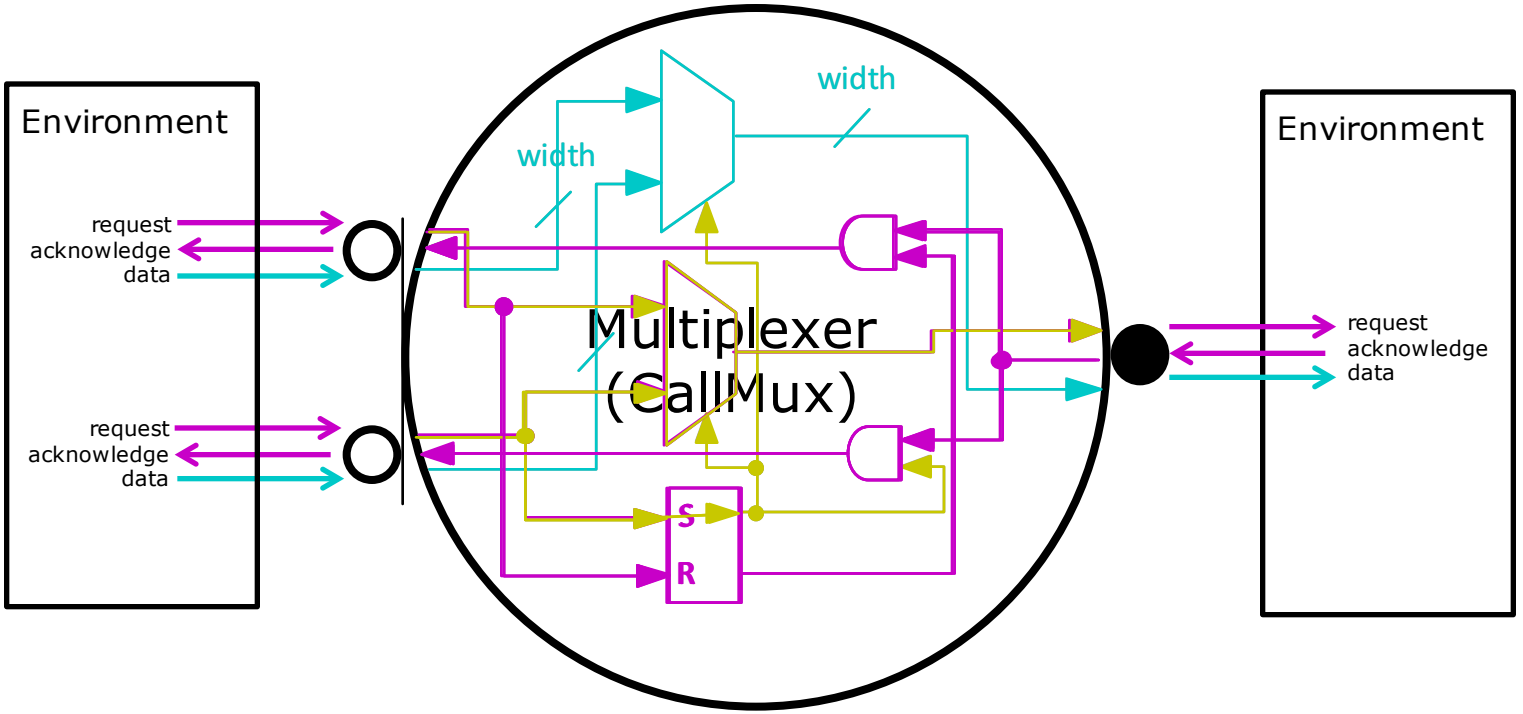
Overall design system



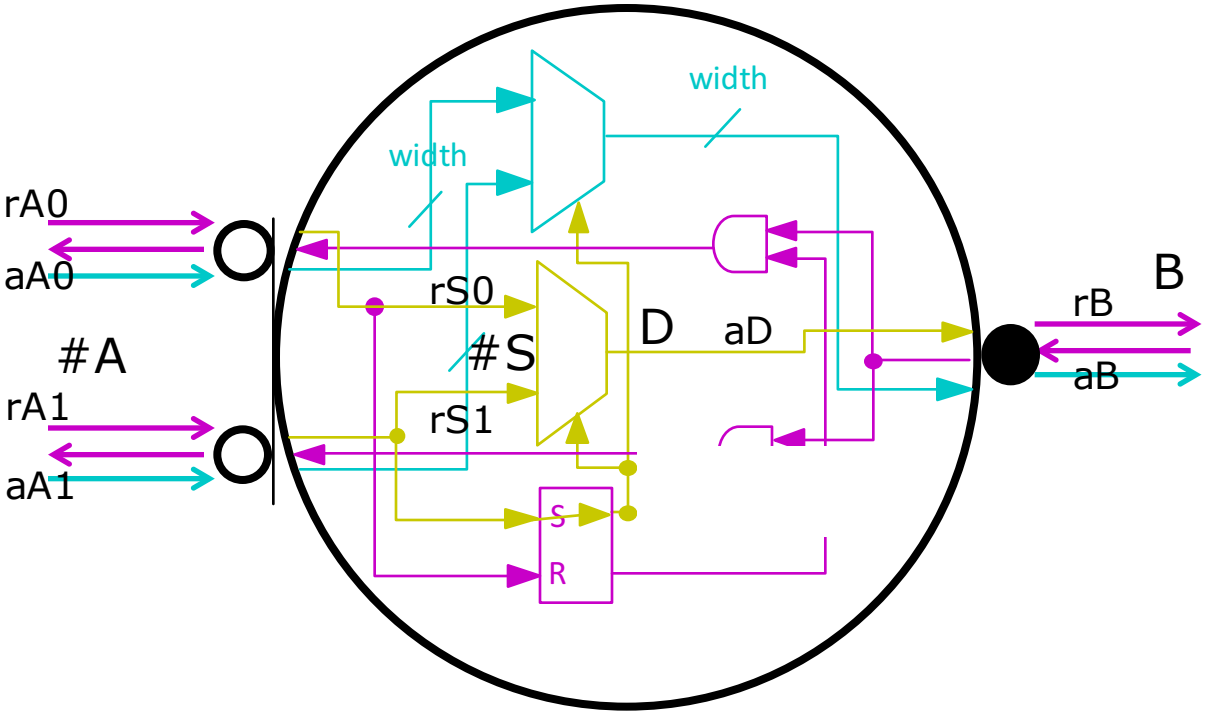
Agenda

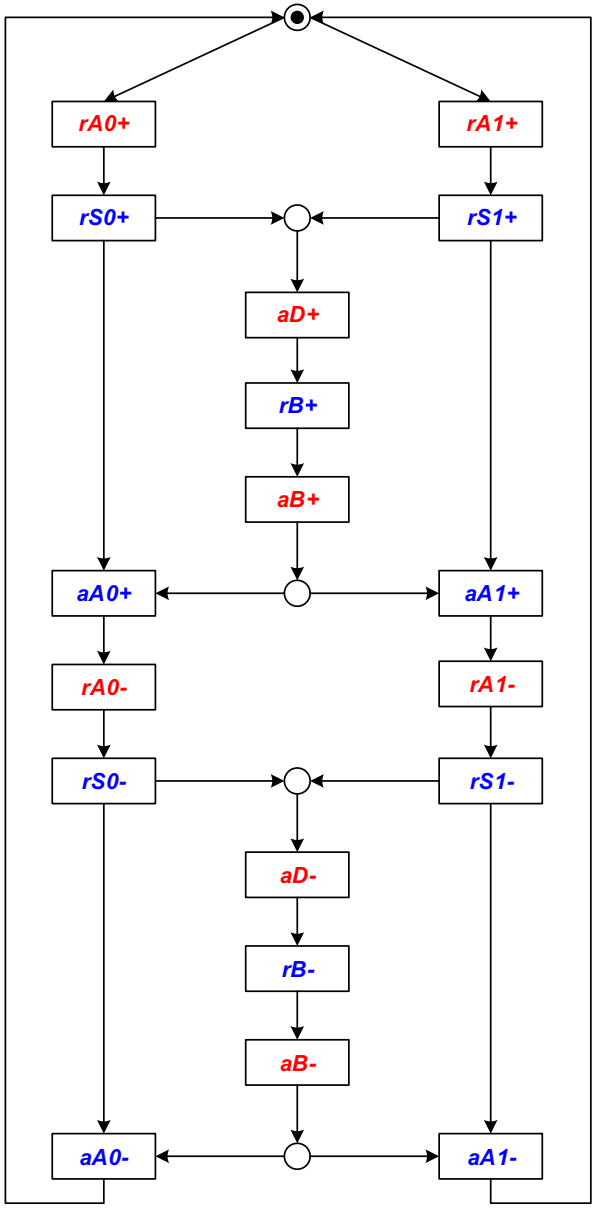
1. Overview
2. **Resynthesis idea**
3. Resynthesis design flow
4. Results

Resynthesis idea



Resynthesis idea – extract control

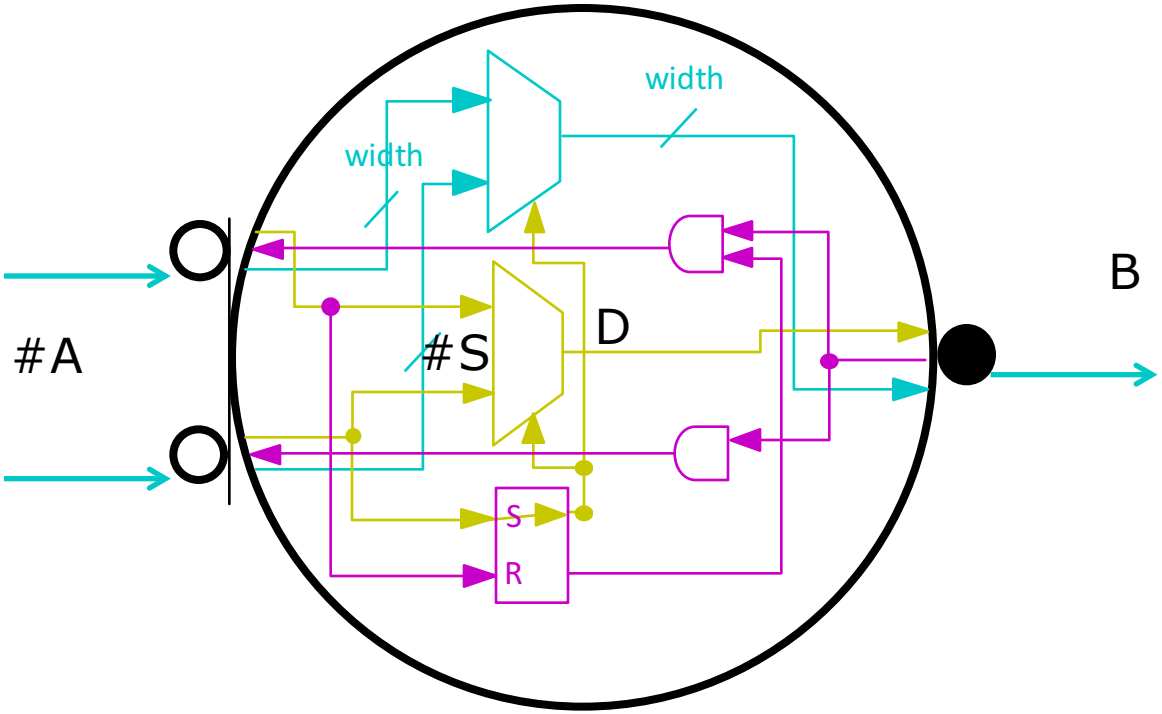




S. Golubcovs, W. Vogler, and N. Kluge, "STG-based resynthesis for balsa circuits," in *Proceedings of the 2013 13th International Conference on Application of Concurrency to System Design*, ser. ACSD '13.

active B, S, D
scaled A, S
f = #(#|(A : (rS;aD;B)))

Resynthesis idea – extract data

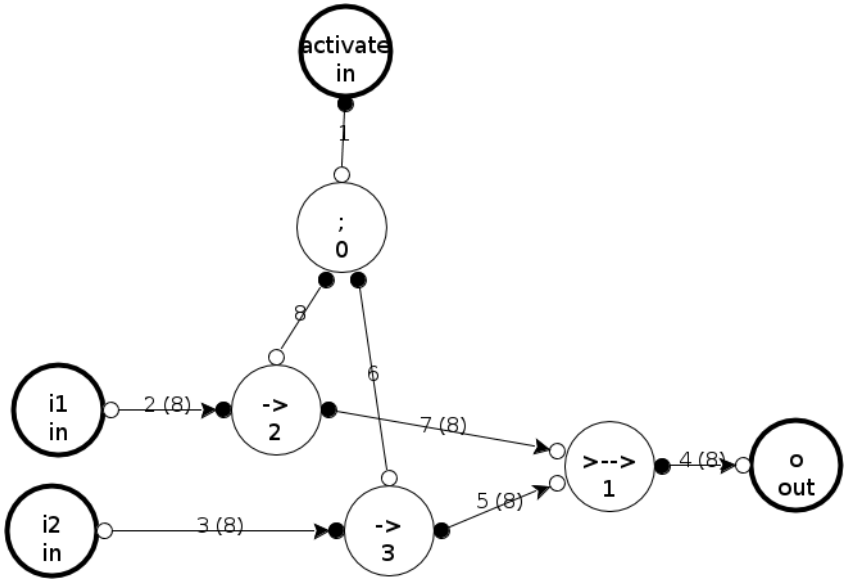


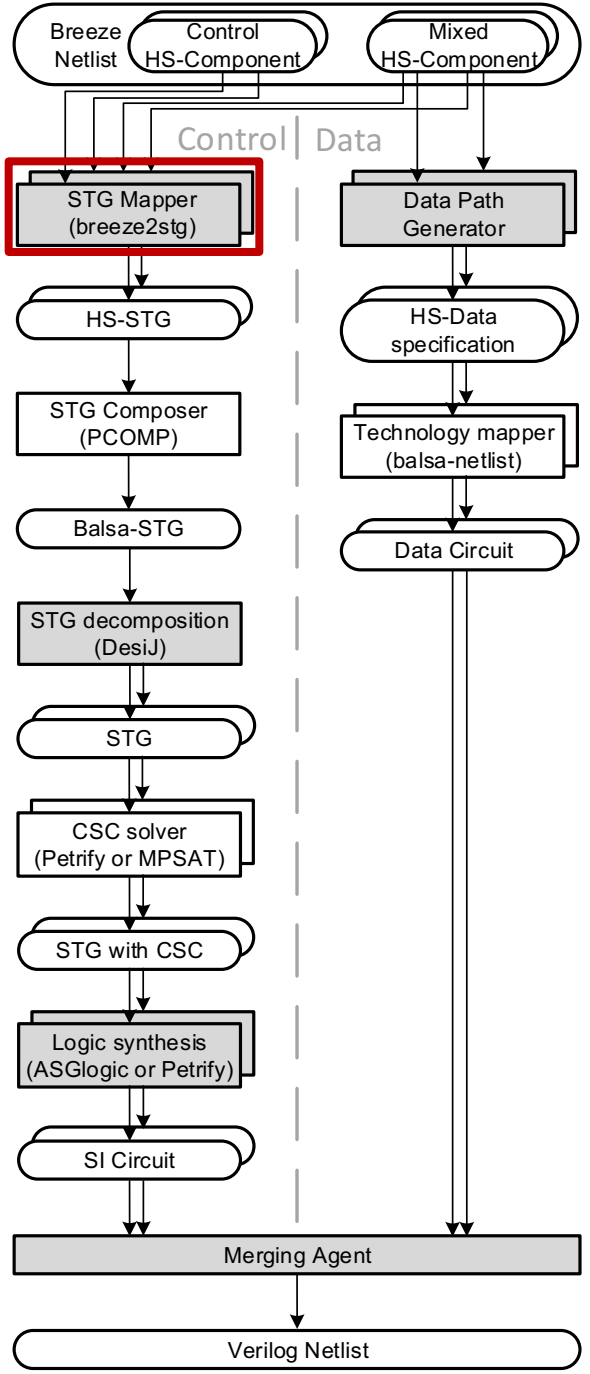

```
module BrzCallMux_3_2 (
  inp_0r, inp_0a, inp_0d,
  inp_1r, inp_1a, inp_1d,
  out_0r, out_0a, out_0d,
  initialise
);
  input inp_0r, inp_1r, out_0a, initia
  output inp_0a, inp_1a, out_0r;
  input [2:0] inp_0d;
  input [2:0] inp_1d;
  output [2:0] out_0d;
  wire select_0n, nselect_0n;
  SDN_MUX2 I0 (out_0d[0], inp_0d[0], i
  SDN_MUX2 I1 (out_0d[1], inp_0d[1], i
  SDN_MUX2 I2 (out_0d[2], inp_0d[2], i
  SDN_MUX2 I3 (out_0r, inp_0r, inp_1r,
  SDN_AN2 I4 (inp_0a, nselect_0n, out_
  SDN_AN2 I5 (inp_1a, select_0n, out_0
  srff I6 (inp_1r, inp_0r, select_0n,
   initialise);
  assign select_0n = inp_1r;
endmodule
```

Agenda

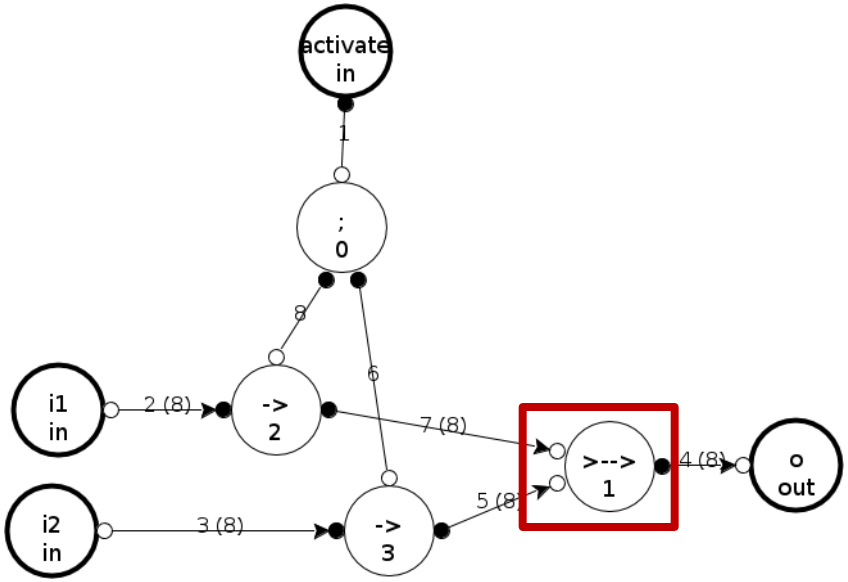
1. Overview
2. Resynthesis idea
3. **Resynthesis design flow**
4. Results

Resynthesis design flow



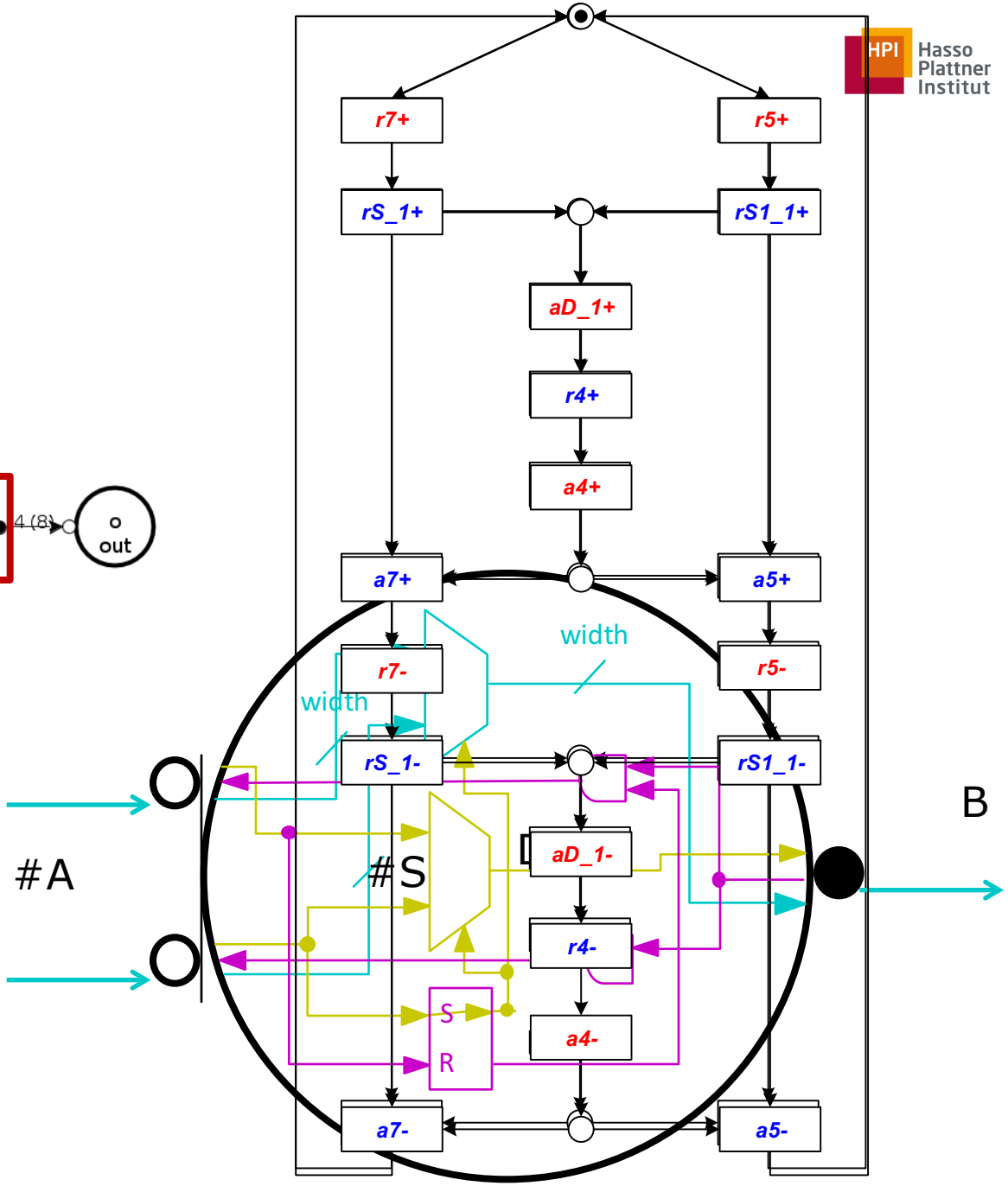


Resynthesis design flow

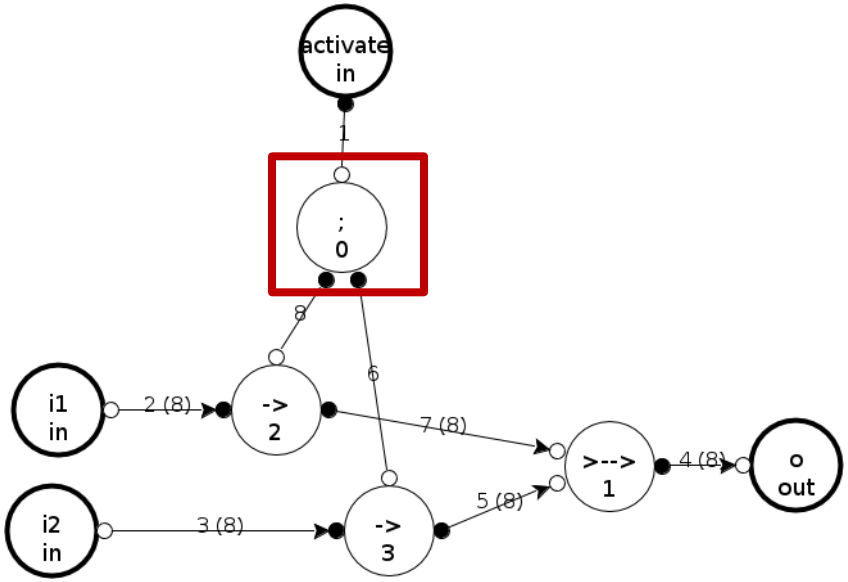


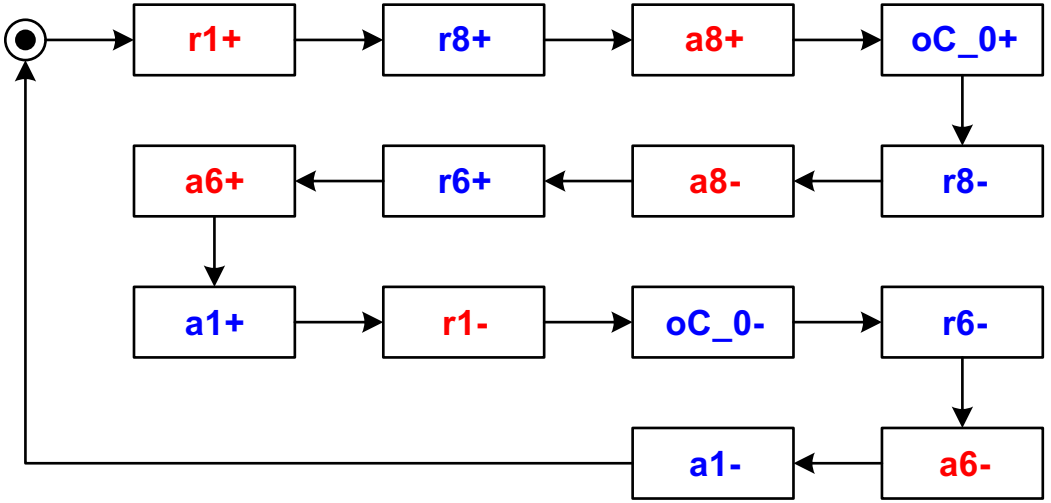
A0 → Channel 7
A1 → Channel 5
B → Channel 4

S0 → Component #1
S1 → Component #1
D → Component #1

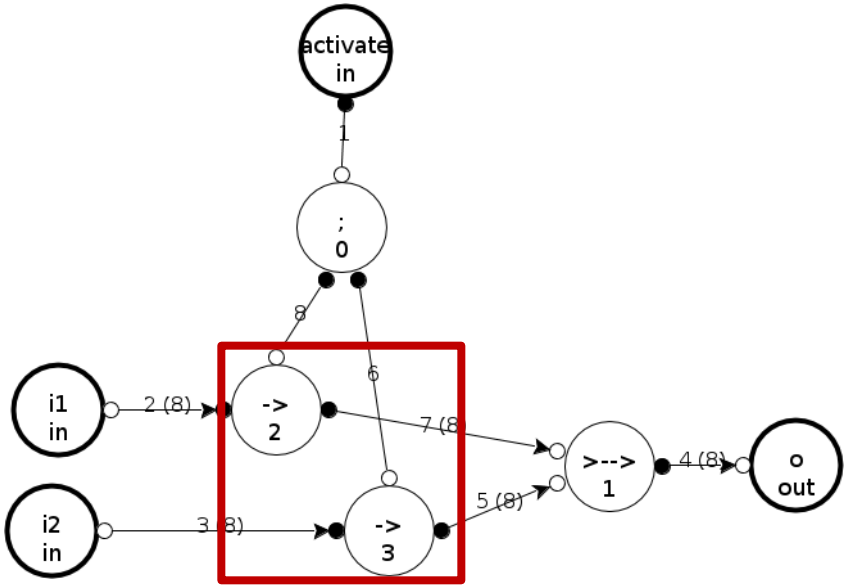


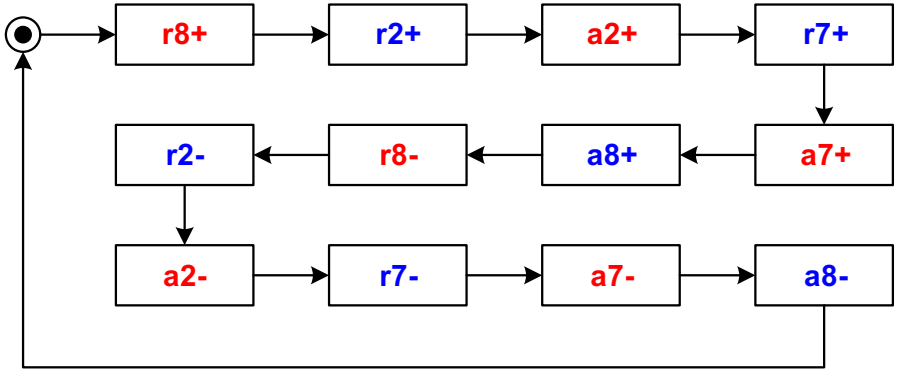
Resynthesis design flow

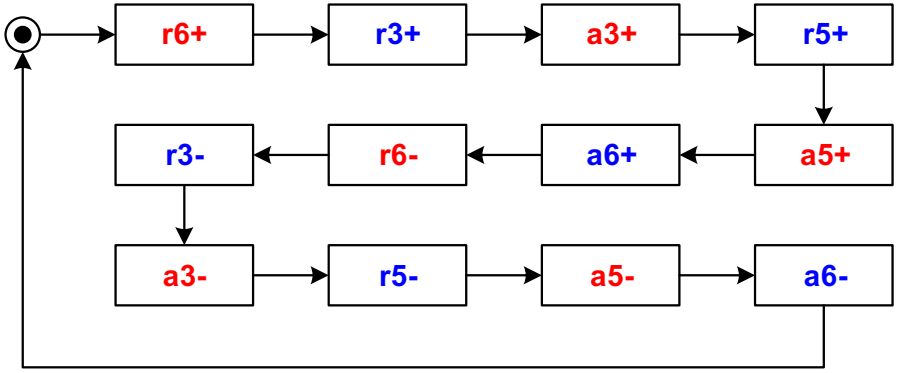




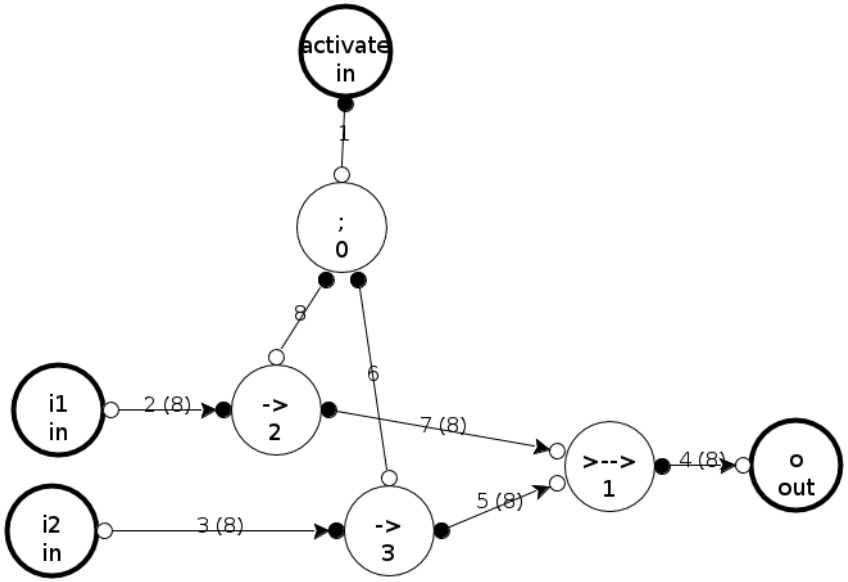
Resynthesis design flow

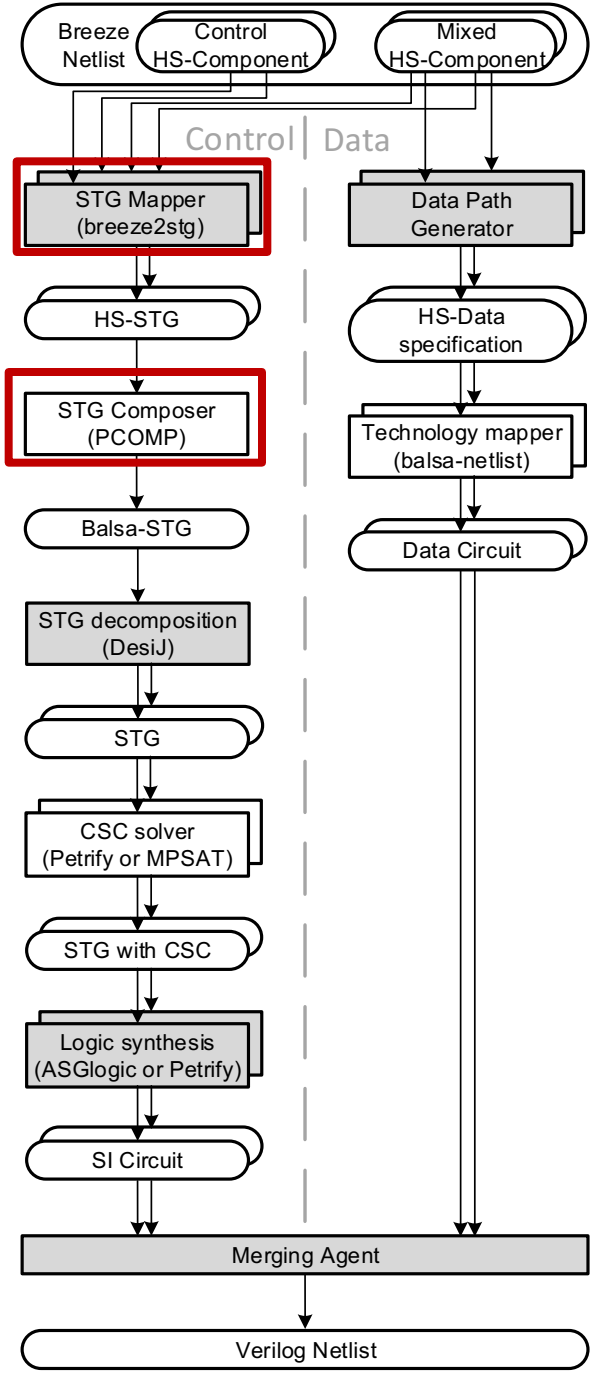




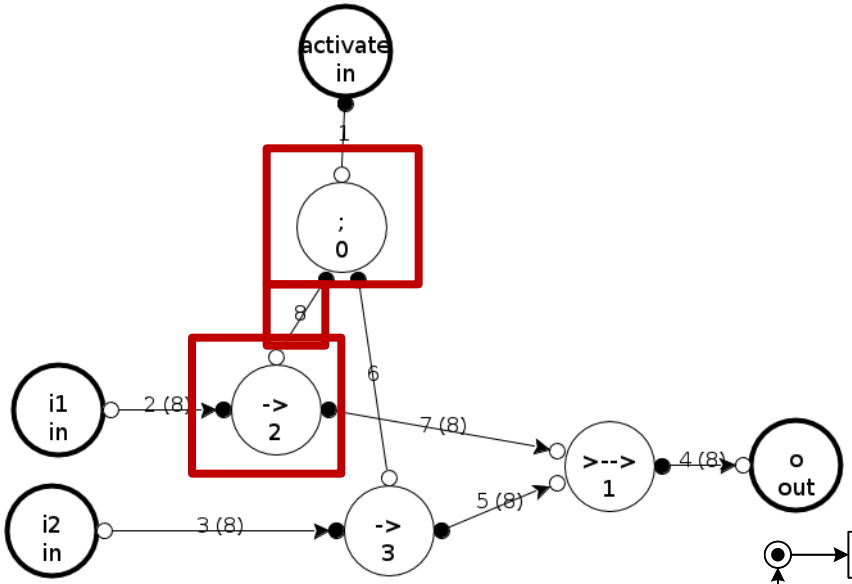


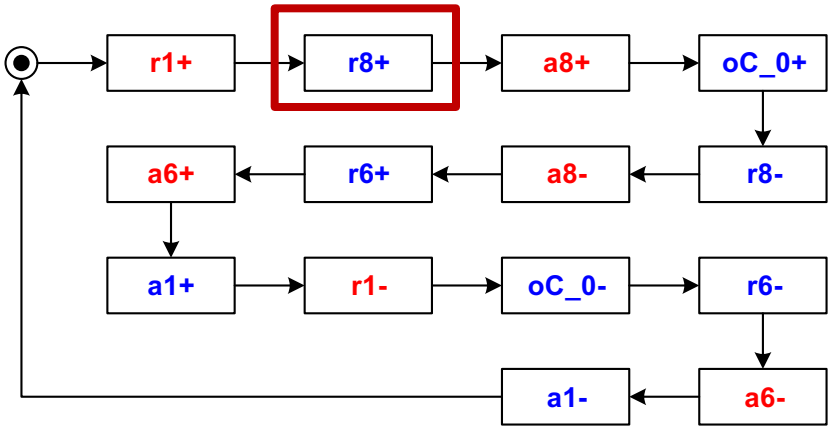
Resynthesis design flow

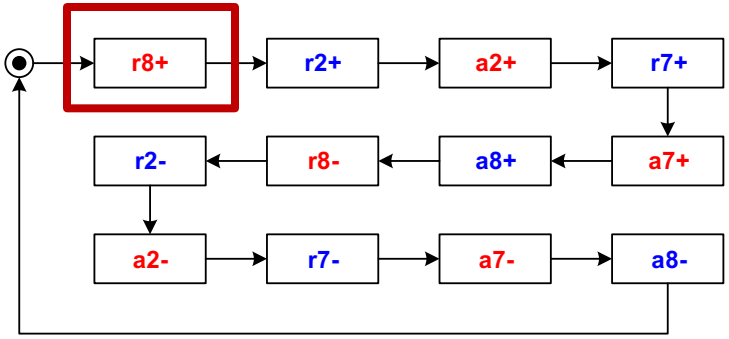


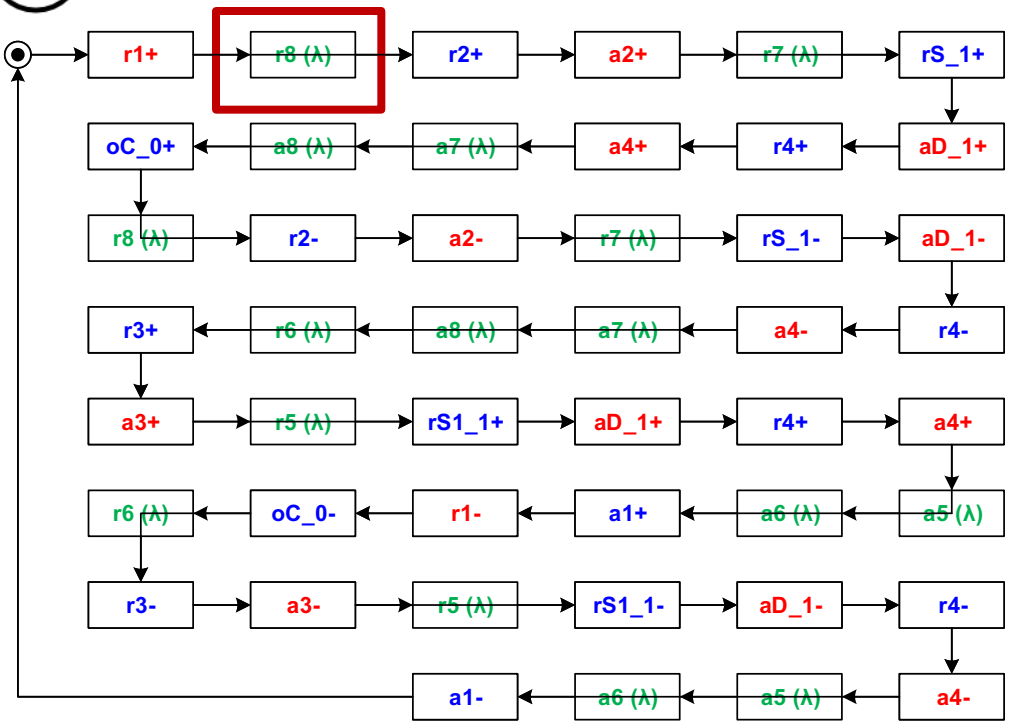


Resynthesis design flow

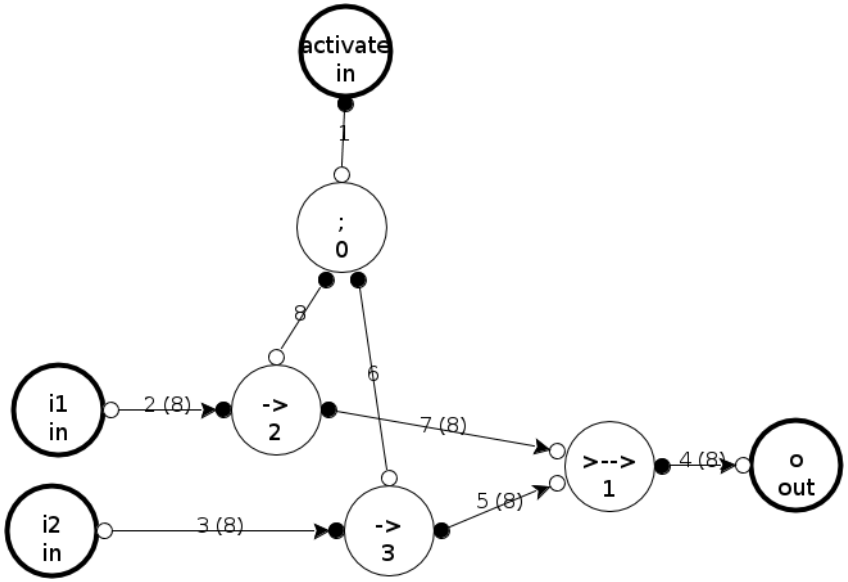


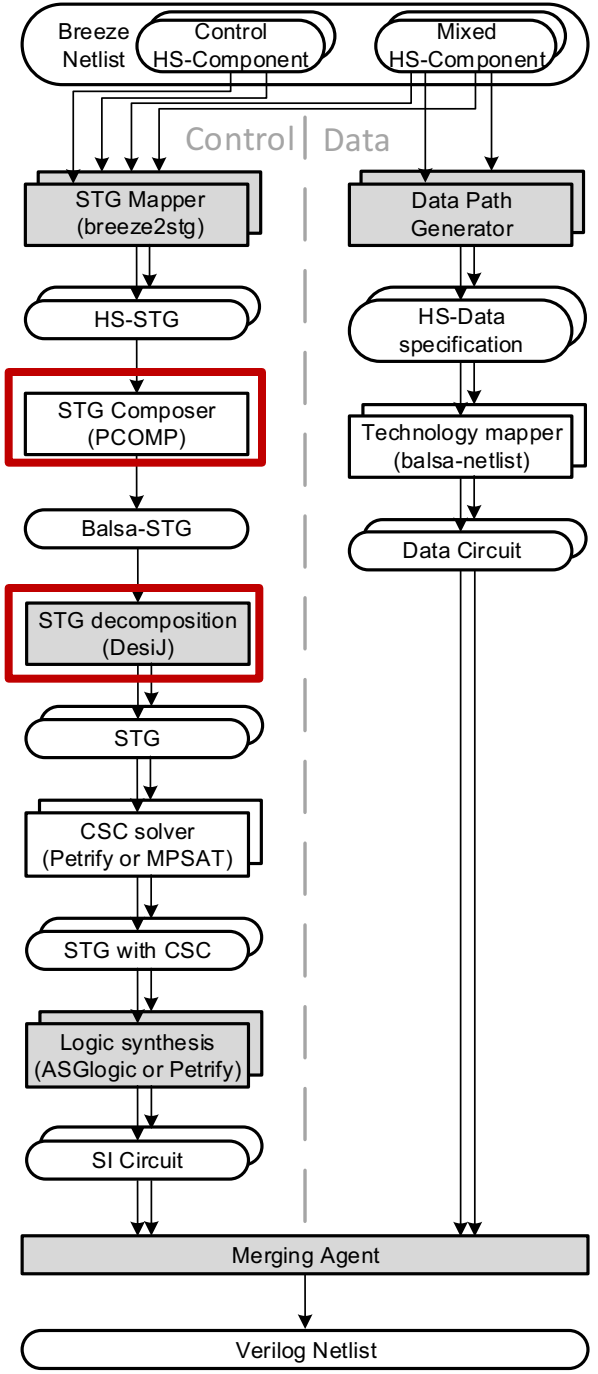




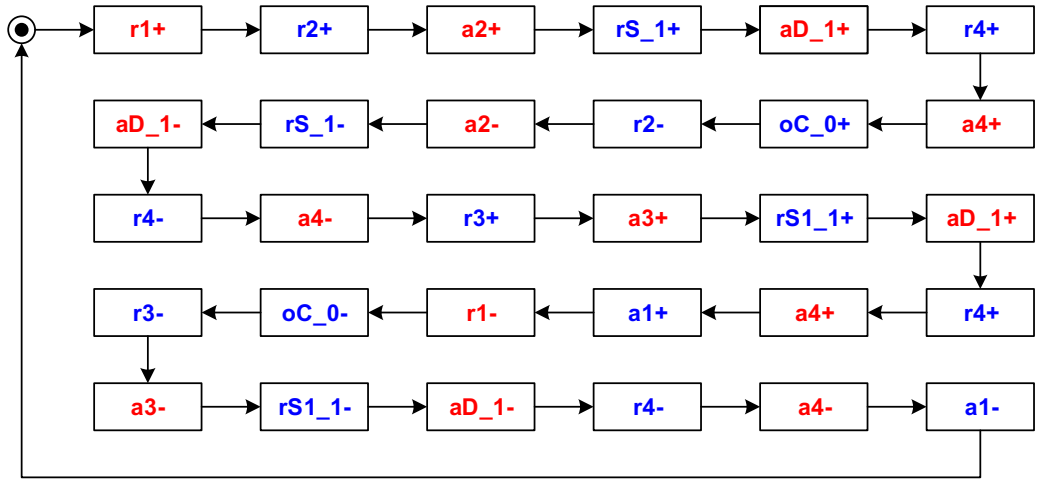


Resynthesis design flow



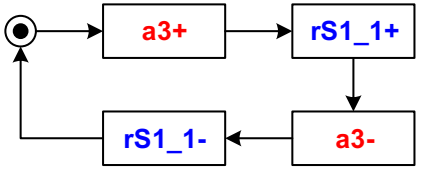


Resynthesis design flow

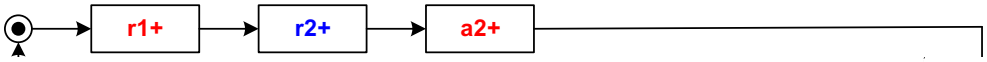


30 Transitions

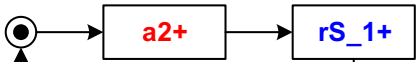
State space explosion

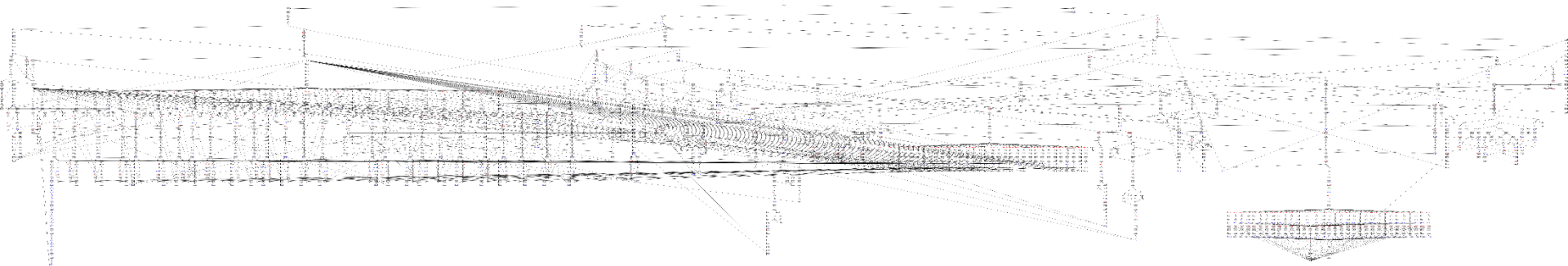


18 Transitions



4 Transitions

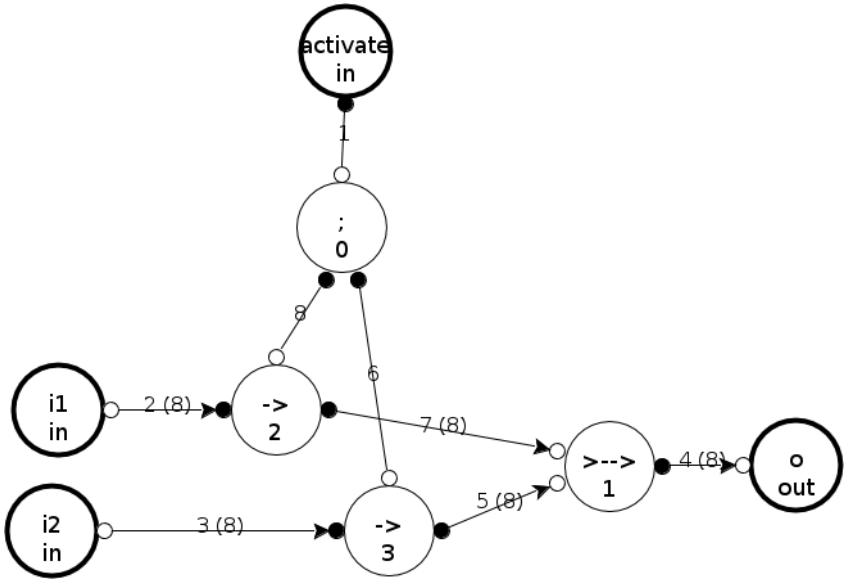


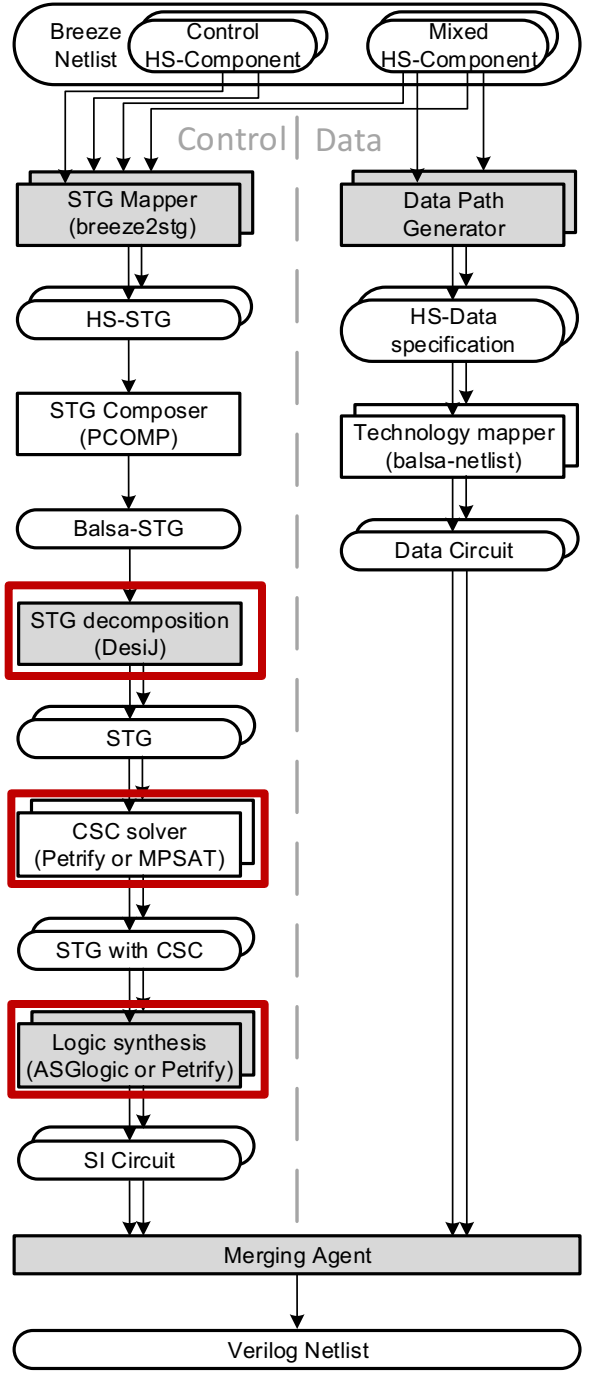


2373 Transitions > 400,000 States (~1.2 GB main memory)

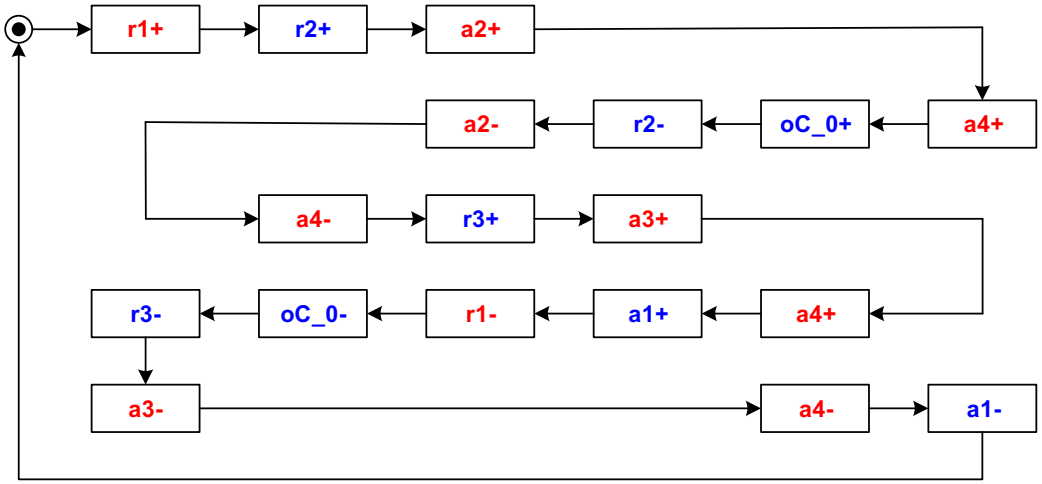
4 (8) Transitions

Resynthesis design flow

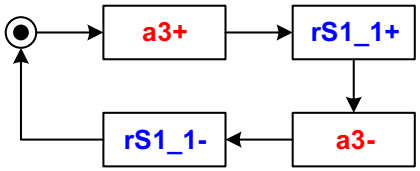




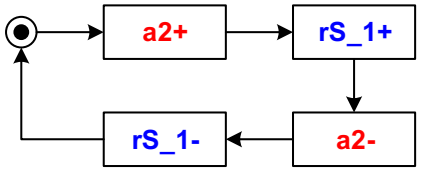
Resynthesis design flow



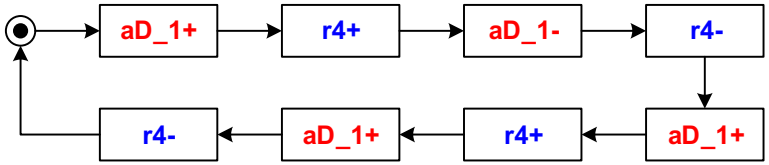

```
SDN_NR2B_1 U0 (.B(a4), .A(oC_0), .X(r3__set_1));
SDN_AN2_1 U1 (.A2(r2), .A1(a4), .X(oC_0__set_1));
SDN_OA21_1 U2 (.A2(oC_0__set_1), .B(r1), .A1(oC_0), .X(oC_0));
SDN_OA21_1 U3 (.A2(r3), .B(a4), .A1(a1), .X(a1));
SDN_OA21_1 U4 (.A2(r3__set_1), .B(oC_0), .A1(r3), .X(r3));
SDN_NR2B_1 U5 (.B(oC_0), .A(r1), .X(r2));
```



assign rS1_1 = a3;

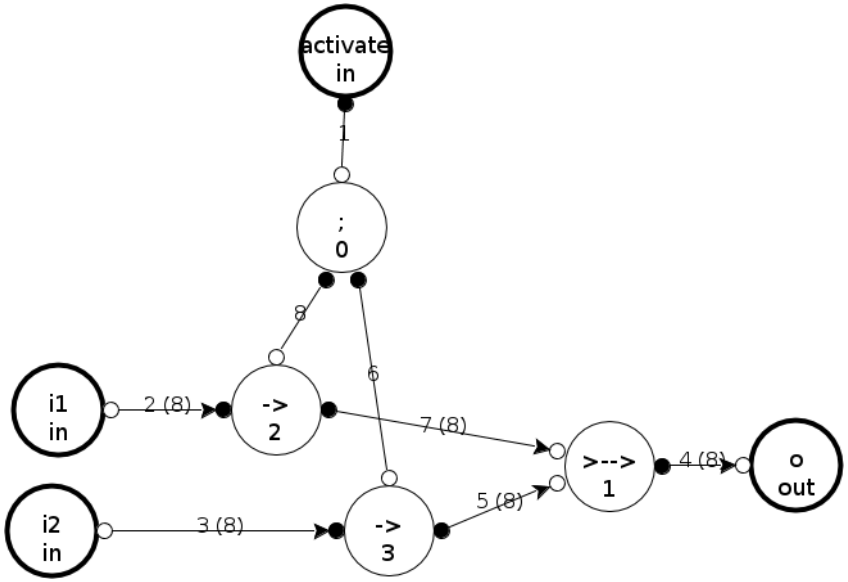


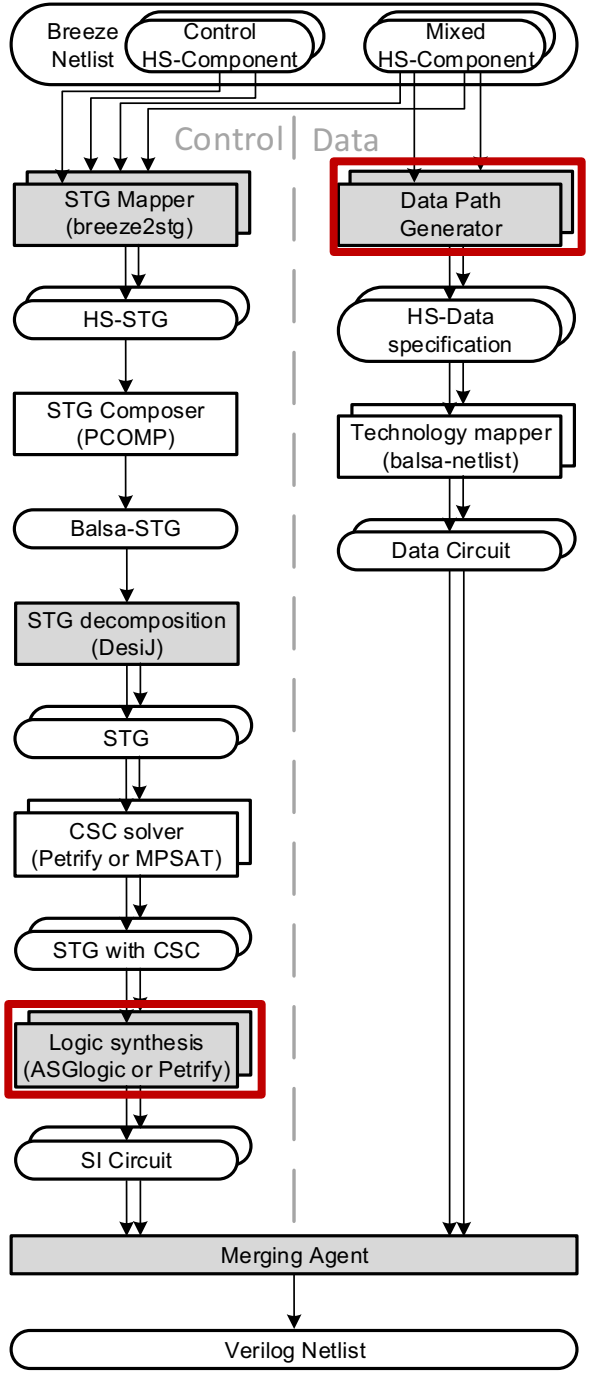
assign rS_1 = a2;



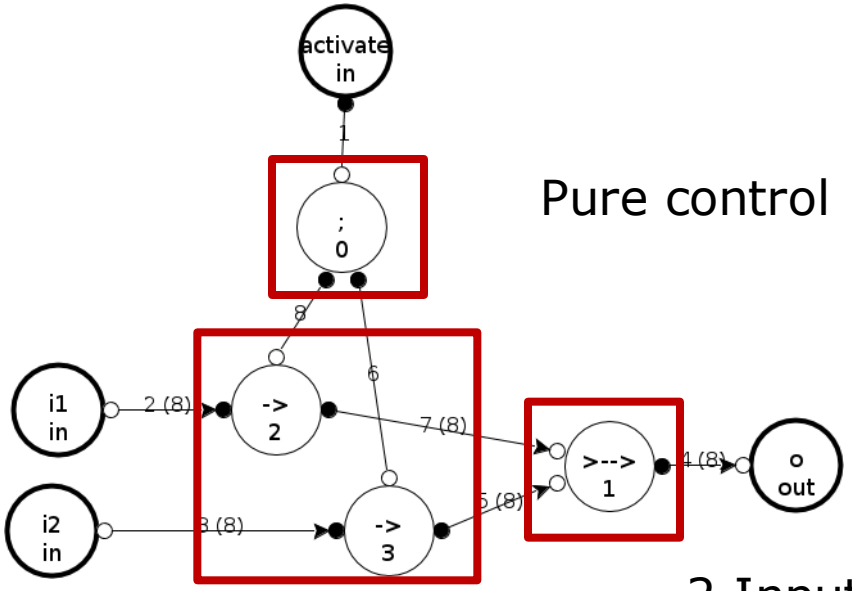
assign r4 = aD_1;

Resynthesis design flow





Resynthesis design flow

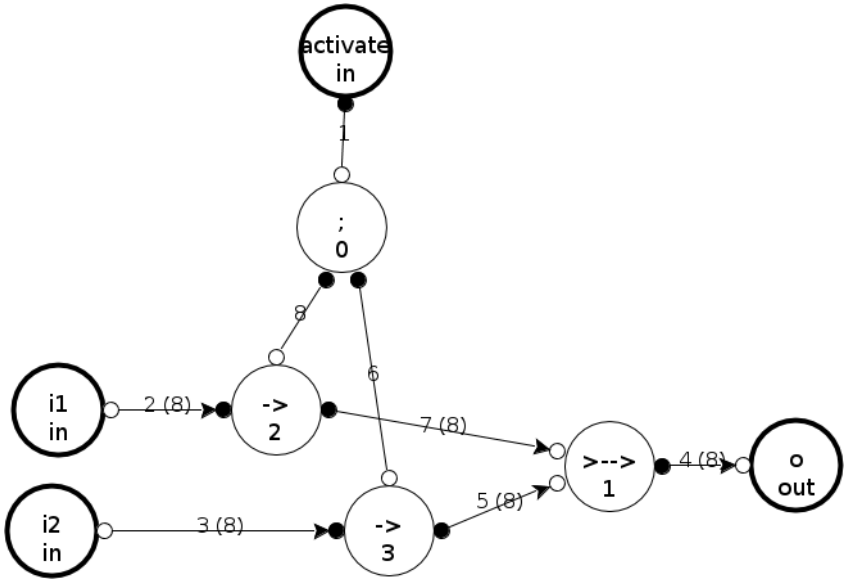


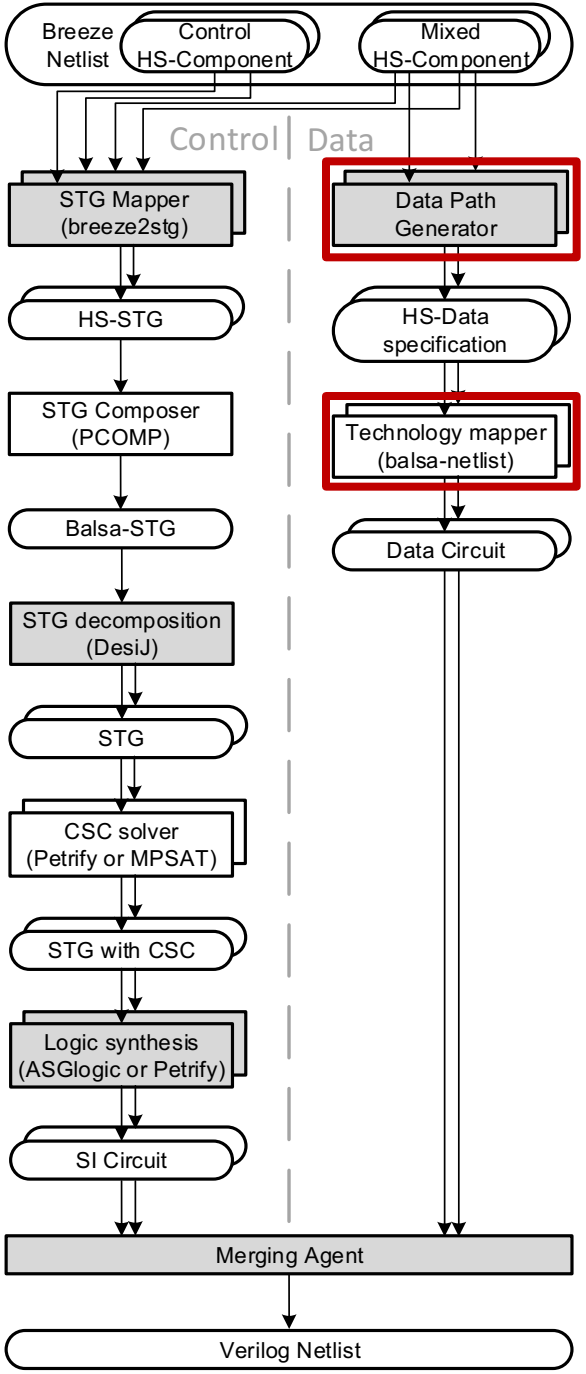
8 bit data width

Pure control → No data path

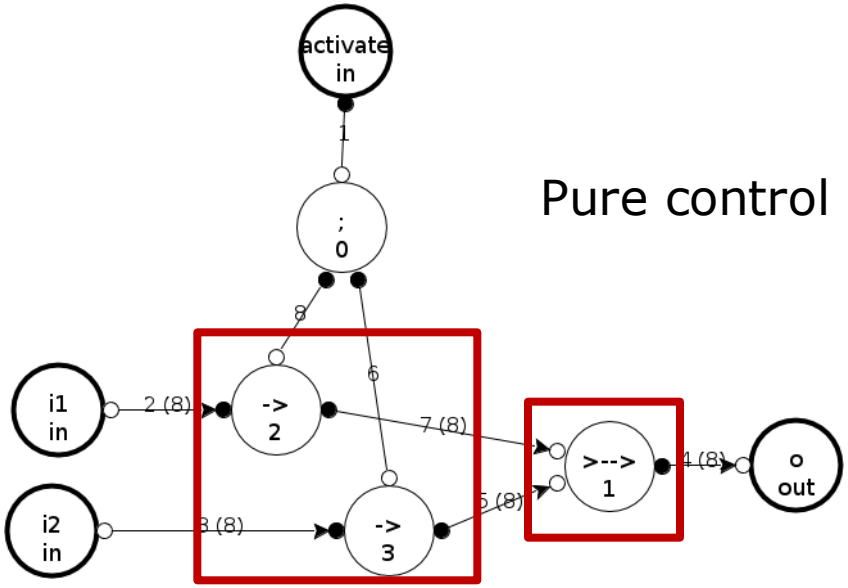
2 Input channels
8 bit data width

Resynthesis design flow





Resynthesis design flow



Pure control → No data path

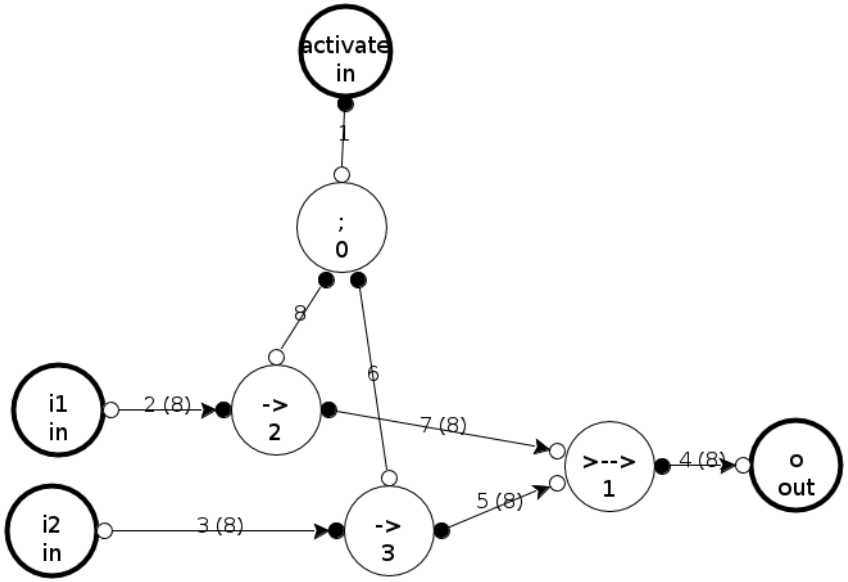
2 Input channels
8 bit data width

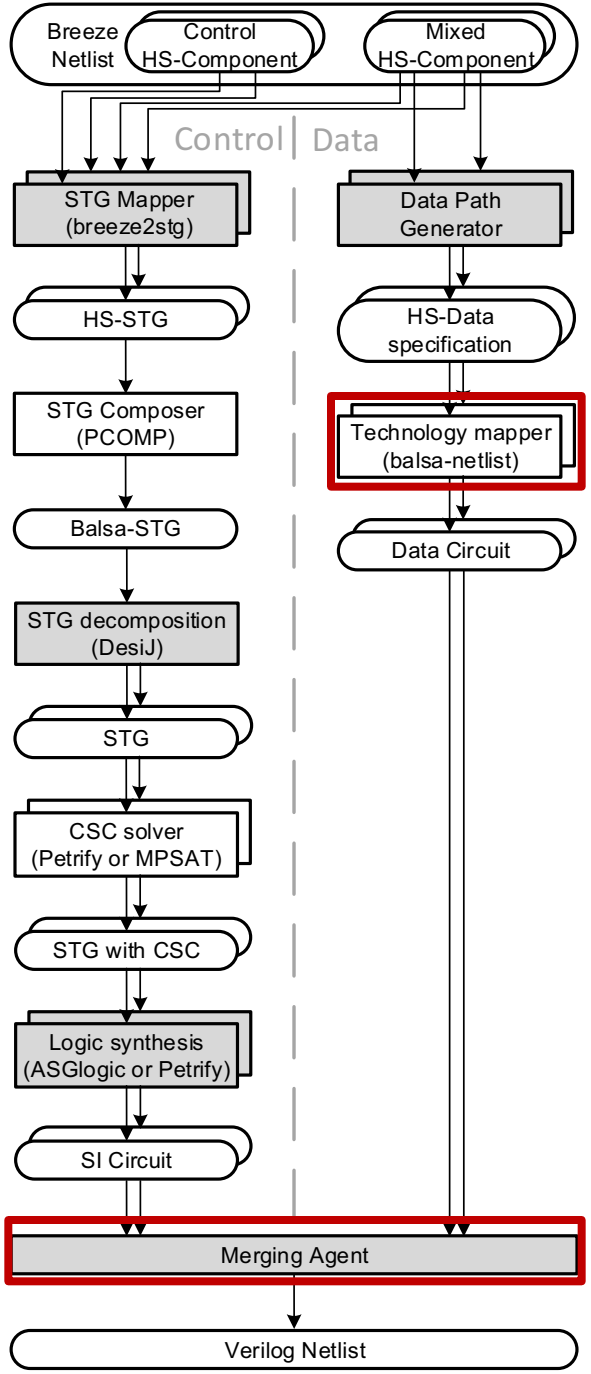
8 bit data width

```
module ResynFetch_0 (
  inp_0d,
  out_0d
);
  input [7:0] inp_0d;
  output [7:0] out_0d;
  assign out_0d[0] = inp_0d[0];
  assign out_0d[1] = inp_0d[1];
  assign out_0d[2] = inp_0d[2];
  assign out_0d[3] = inp_0d[3];
  assign out_0d[4] = inp_0d[4];
  assign out_0d[5] = inp_0d[5];
  assign out_0d[6] = inp_0d[6];
  assign out_0d[7] = inp_0d[7];
endmodule
```

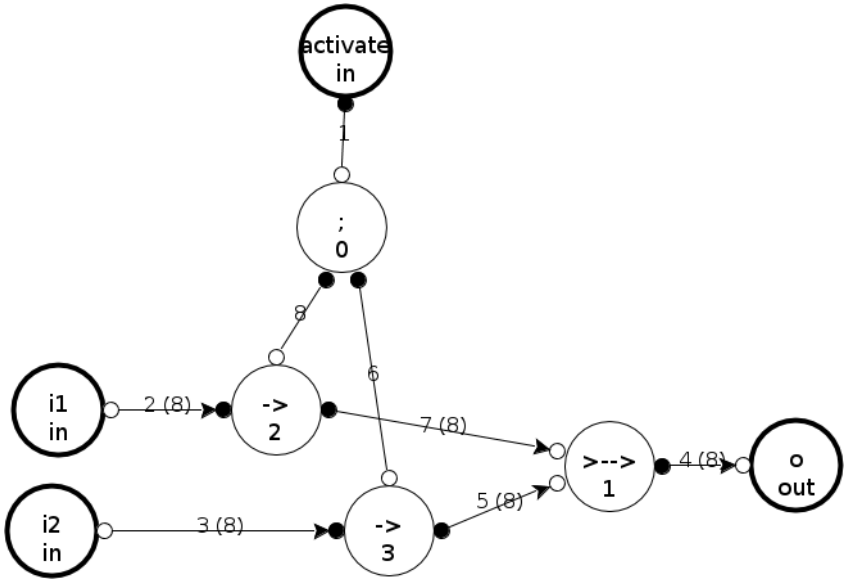
```
module ResynCallMux_0 (
   inp_0r, inp_0d,
   inp_1r, inp_1d,
   out_0r, out_0d,
);
   input inp_0r, inp_1r;
   output out_0r;
   input [2:0] inp_0d;
   input [2:0] inp_1d;
   output [2:0] out_0d;
   wire select_0n;
   SDN_MUX2 I0 (out_0d[0], inp_0d[0], inp_1d[0], select_0n);
   SDN_MUX2 I1 (out_0d[1], inp_0d[1], inp_1d[1], select_0n);
   SDN_MUX2 I2 (out_0d[2], inp_0d[2], inp_1d[2], select_0n);
   SDN_MUX2 I3 (out_0r, inp_0r, inp_1r, select_0n);
   assign select_0n = inp_1r;
endmodule
```

Resynthesis design flow





Resynthesis design flow




```
// Control
ResynControl_muxtest0 S0 (a3,_reset,rS1_1);
ResynControl_muxtest1 S1 (a2,a3,a4,r1,_reset,a1,oC_0,r2,r3);
ResynControl_muxtest2 S2 (a2,_reset,rS_1);
ResynControl_muxtest3 S3 (aD_1,_reset,r4);

// Data
ResynFetch_0 I2 (d2, d7);
ResynFetch_0 I3 (d3, d5);
ResynCallMux_0 I1 (rS_1, d7, rS1_1, d5, aD_1, d4);
```

Agenda

1. Overview
2. Resynthesis idea
3. Resynthesis design flow
4. **Results**

Experimental results 130nm (Post synthesis)
Latency in ps

	Balsa	Resynthesis			
		w/o STG decomposition			
		Petrify		ASGlogic	
gcd	949	N/A		792	-17%
counter	43	N/A		34	-19%
counter alt.	50	36	-29%	39	-23%
shifter	16	13	-13%	15	-4%
buffer	8	8	-4%	8	-5%

- Petrify sometimes fails to generate proper reset logic
- However in most cases Petrifys solutions are faster (Todo for us)
- Applying STG decomposition will (in general) result in performance overhead

Impact of STG decomposition

	# HS comp	w/o STG deco.	
		Petrify	ASGlogic
SAMIPS	24	no	no
CP0	252	no	no
EX	474	no	no
FWUnit	36	no	no
ID	3	yes	yes
ADD4	15	yes	no

- All benchmarks could be decomposed and for all their components CSC could be obtained
- Main problem for ASGlogic: AND gate decomposition

Summary & future work

- Summary
  - Fully automated resynthesis flow for Balsa
  - Download: https://github.com/hpiasg
  - Performance improvement of up to 23%
  - STG decomposition tackles state space explosion
  - Logic synthesis with proper reset insertion

- Future work
  - AND gate decomposition
  - Data path optimisation
  - Delay matching
  - Area / Power analysis
  - …

Thank you!

Questions?

Backup slides

Backup slide: Clustering problem?

Backup slide: Petrify reset

Backup slide: Data path optimsation

Backup slide: AND deco