A Fast Logic Simulator Using a Look Up Table Cascade Emulator

H.Nakahara T.Sasao M.Matsuura

Department of Computer Science and Electronics

Kyushu Institute of Technology, Japan

Outline of the Talk

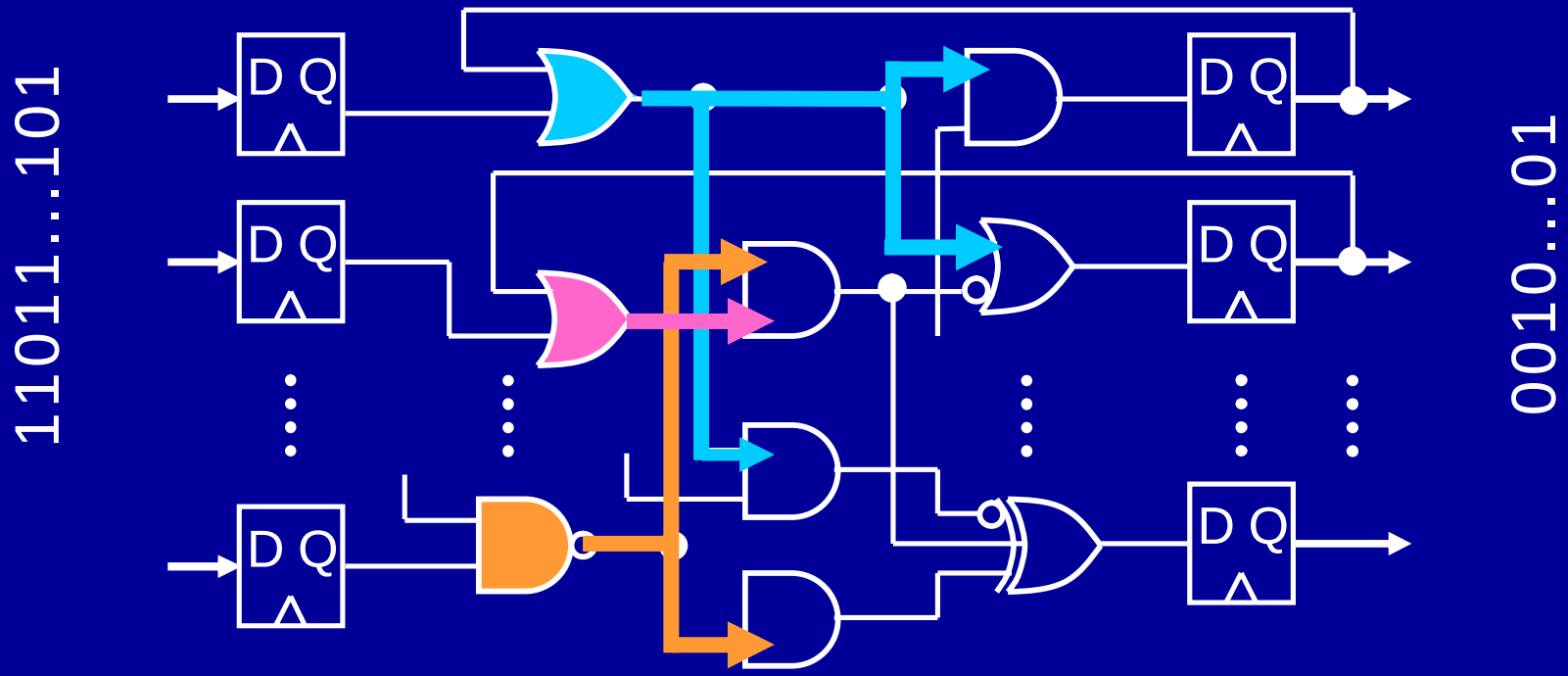
- Background
- Look Up Table (LUT) Cascade Emulator
- Logic Simulation using the LUT Cascade Emulator
- Experimental Results
- Conclusion

Background

- Verification time of the LSI design increases
- High-speed logic simulators are needed
  - Event-driven simulator
  - Cycle-based simulator

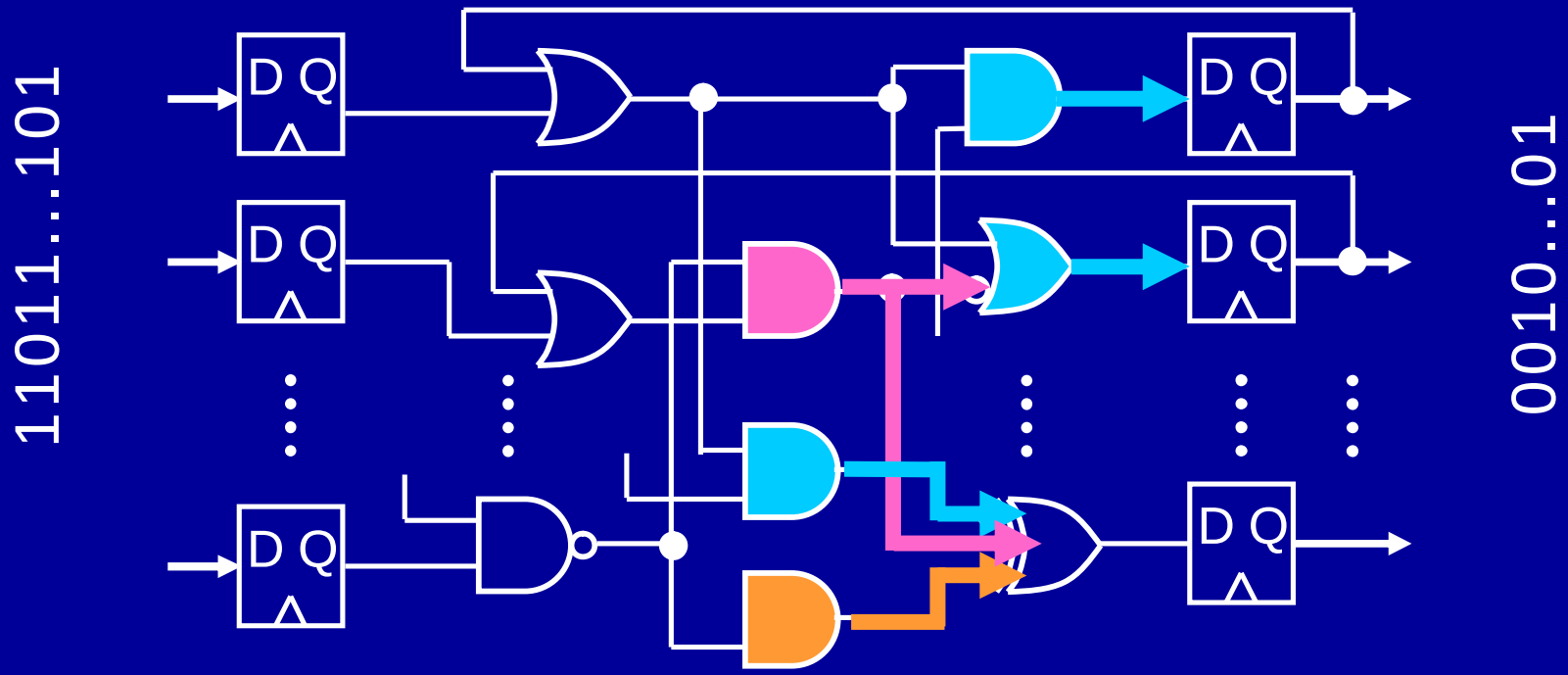
Event Driven Simulator

- Evaluates only gates whose input signal change
  - ○ Performs the timing simulation
  - X Is slow



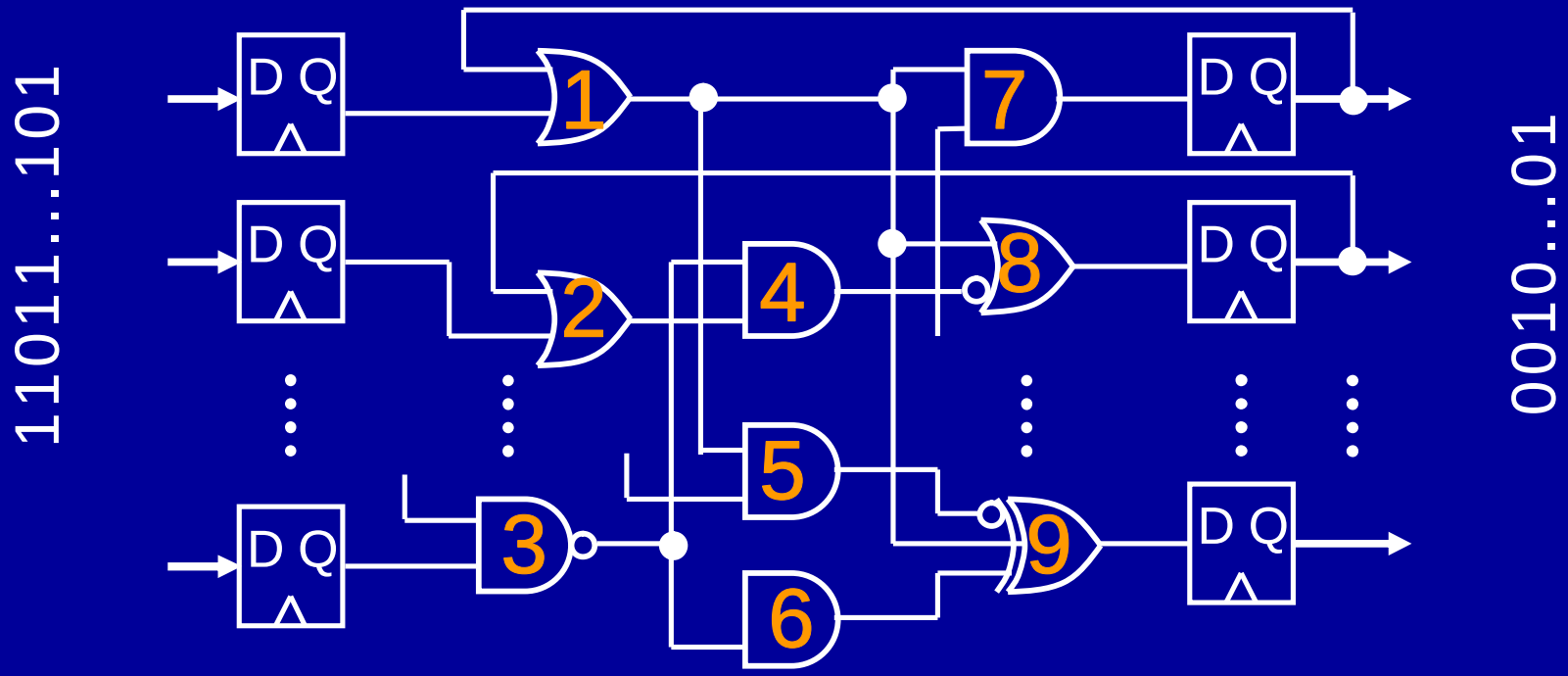
Event Driven Simulator

- Evaluates only gates whose input signal change
  - ○ Performs the timing simulation
  - X Is slow



Cycle Based Simulator

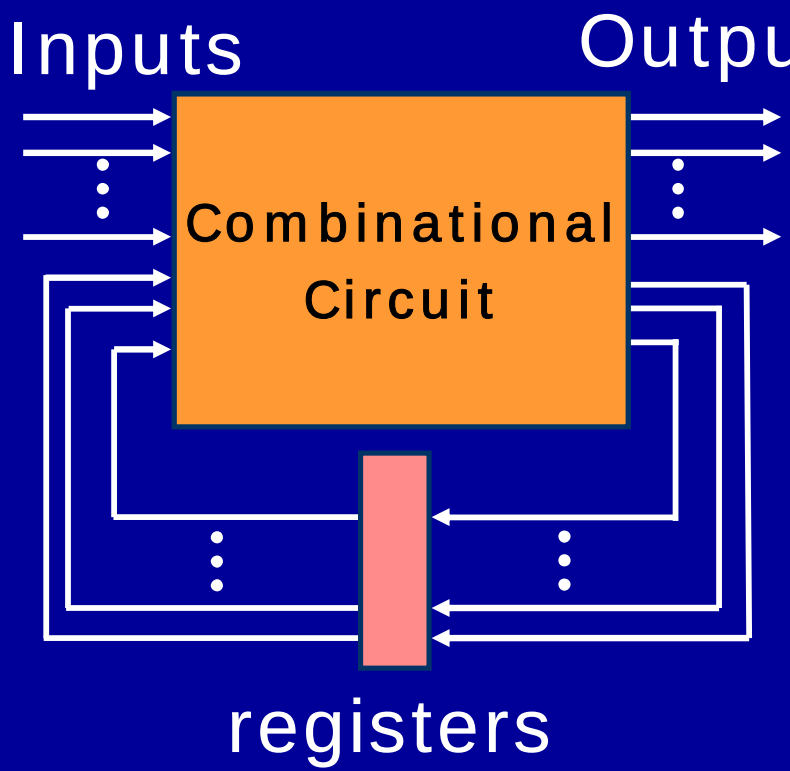
- Evaluates all gates for each clock cycle
  - ○ Is fast
  - X does not perform timing simulation

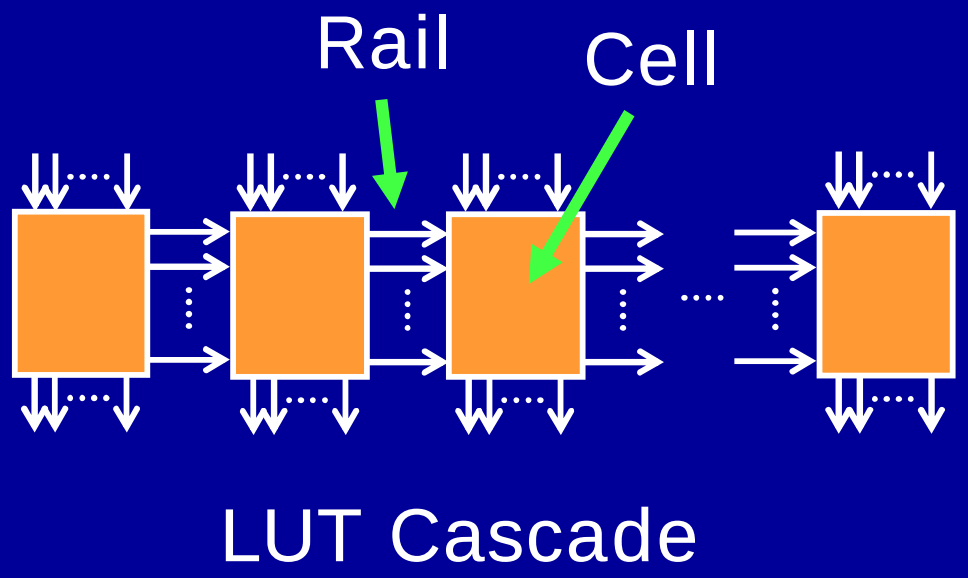


Feature of the Proposed Simulator

- Uses an LUT cascade emulator
- Cycle-based logic simulator
- Runs on a standard PC
  - Cheaper than hardware accelerator
  - The performance can be enhanced with the improvement of the PCs

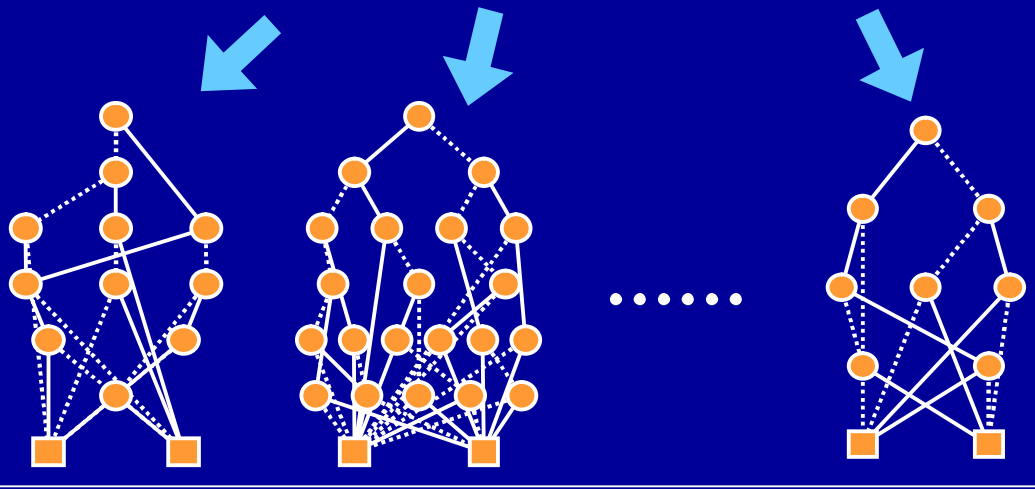
LUT Cascade


Inputs
Outputs
Combinational Circuit
registers


Rail
Cell
LUT Cascade

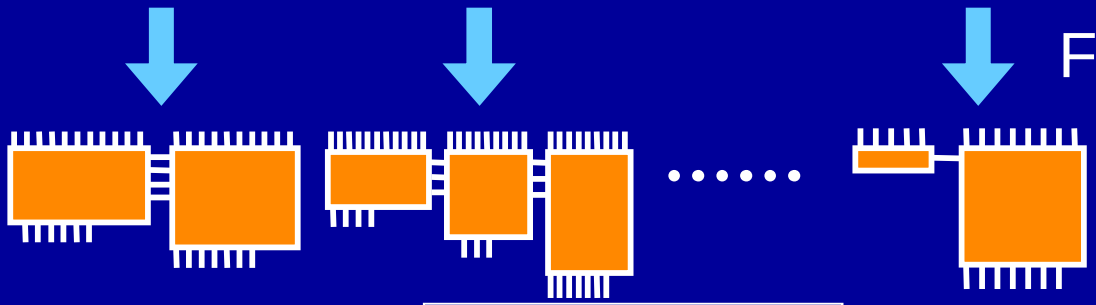
Synthesis for LUT Cascades

Multiple-Output Logic Function



Partition outputs
Construct BDD_for_CFs

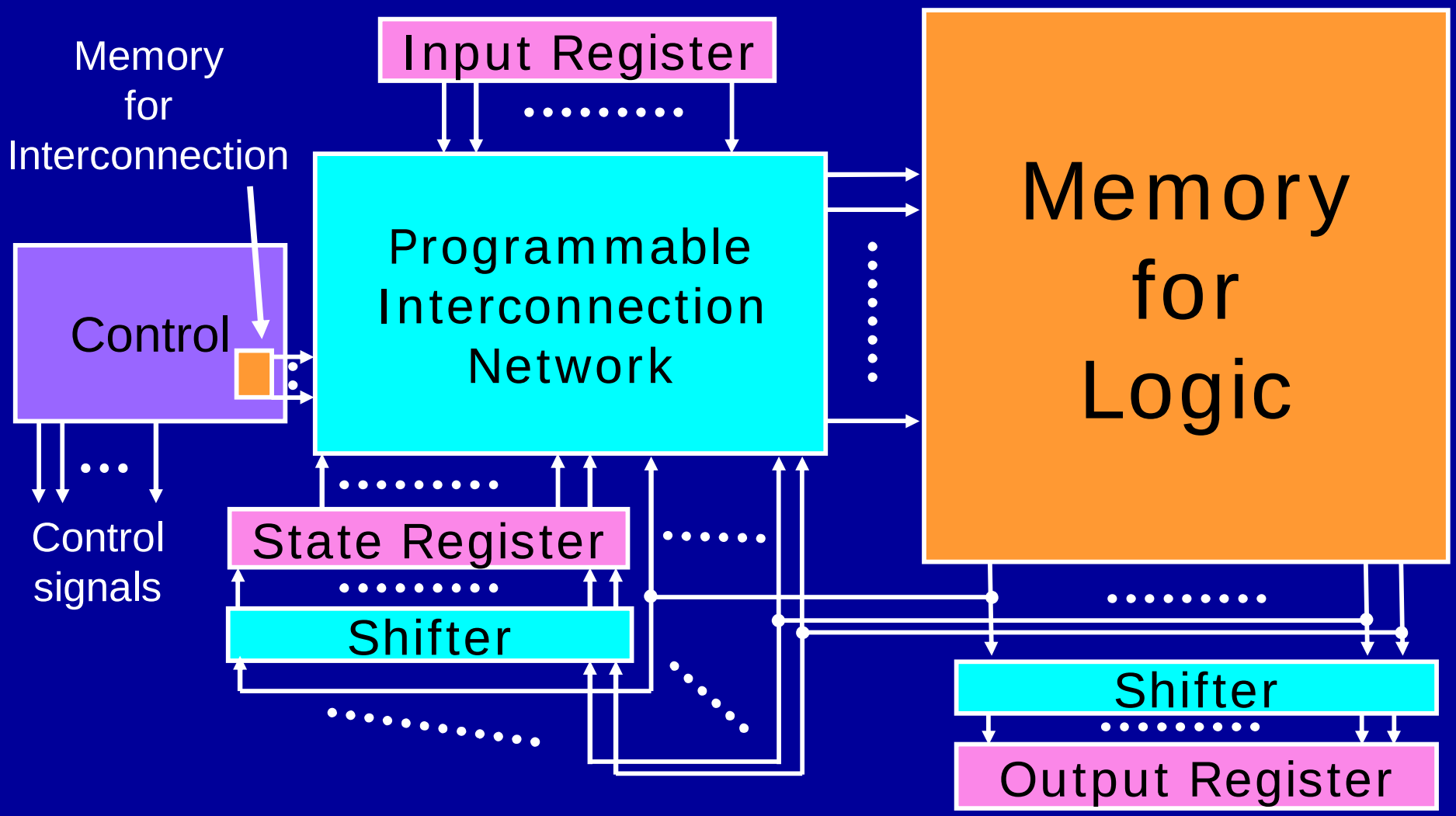
BDD_for_CF (Binary Decision Diagram for Characteristic Function)



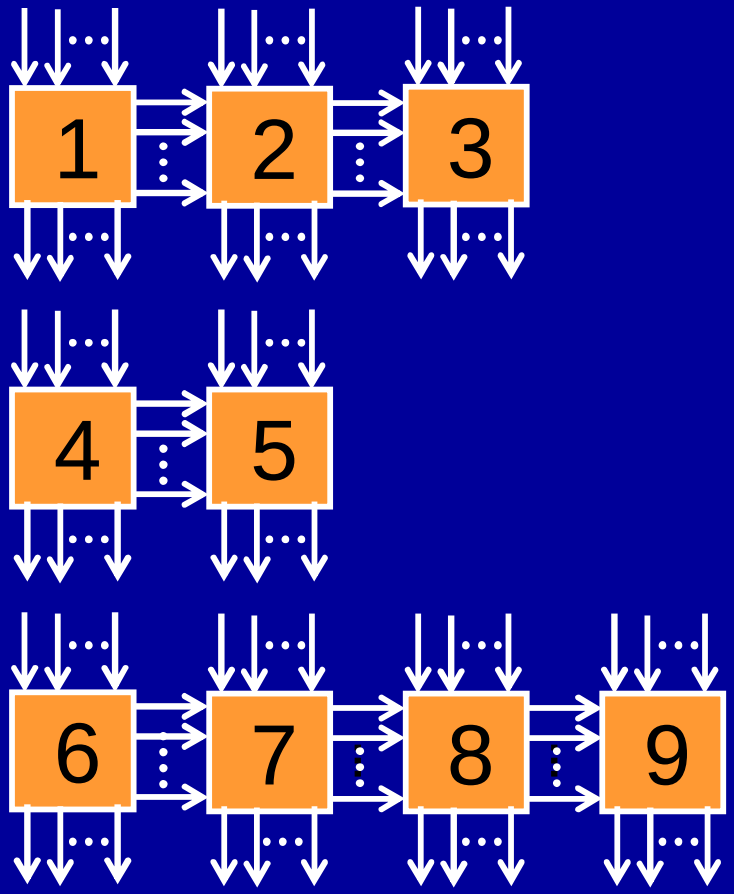
Functional Decomposition

LUT Cascade

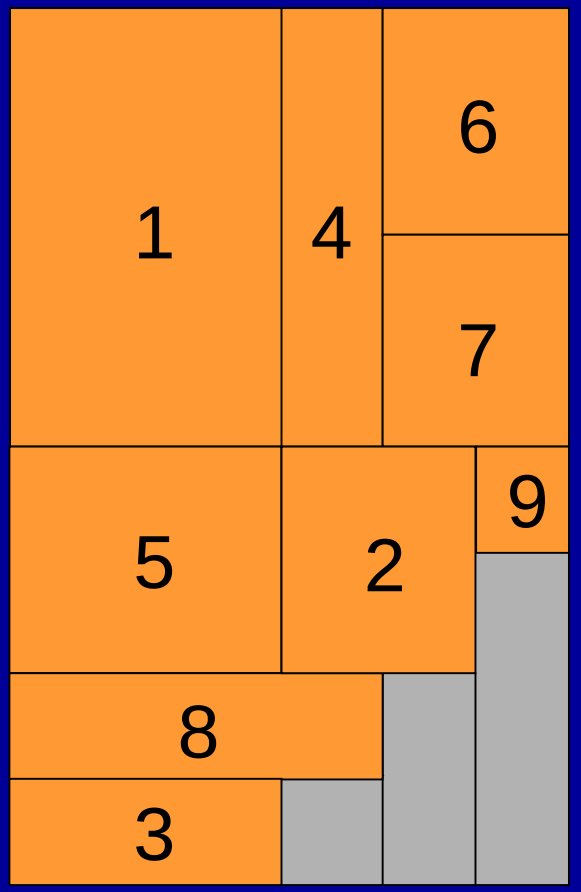
LUT Cascade Emulator



Memory-Packing

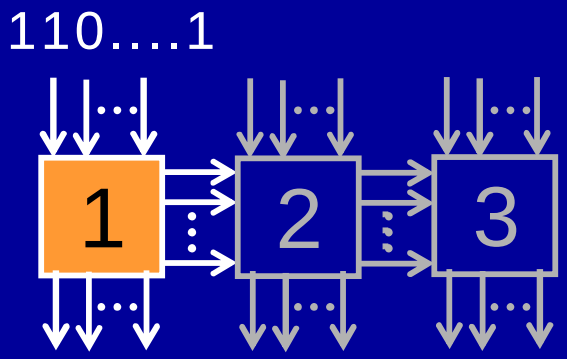


LUT cascades



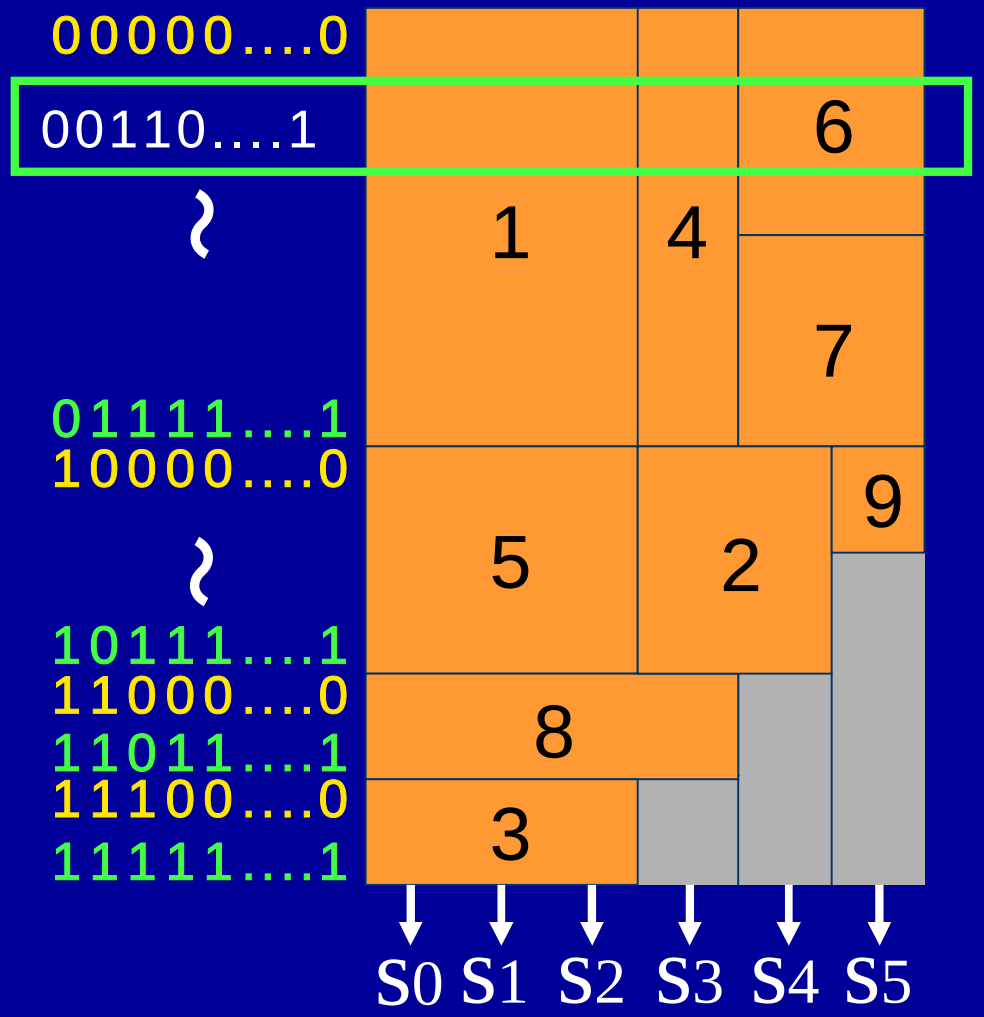
Memory for Logic

Example – Setting Cell Address

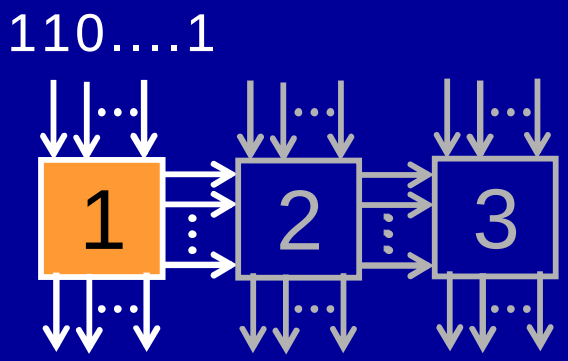


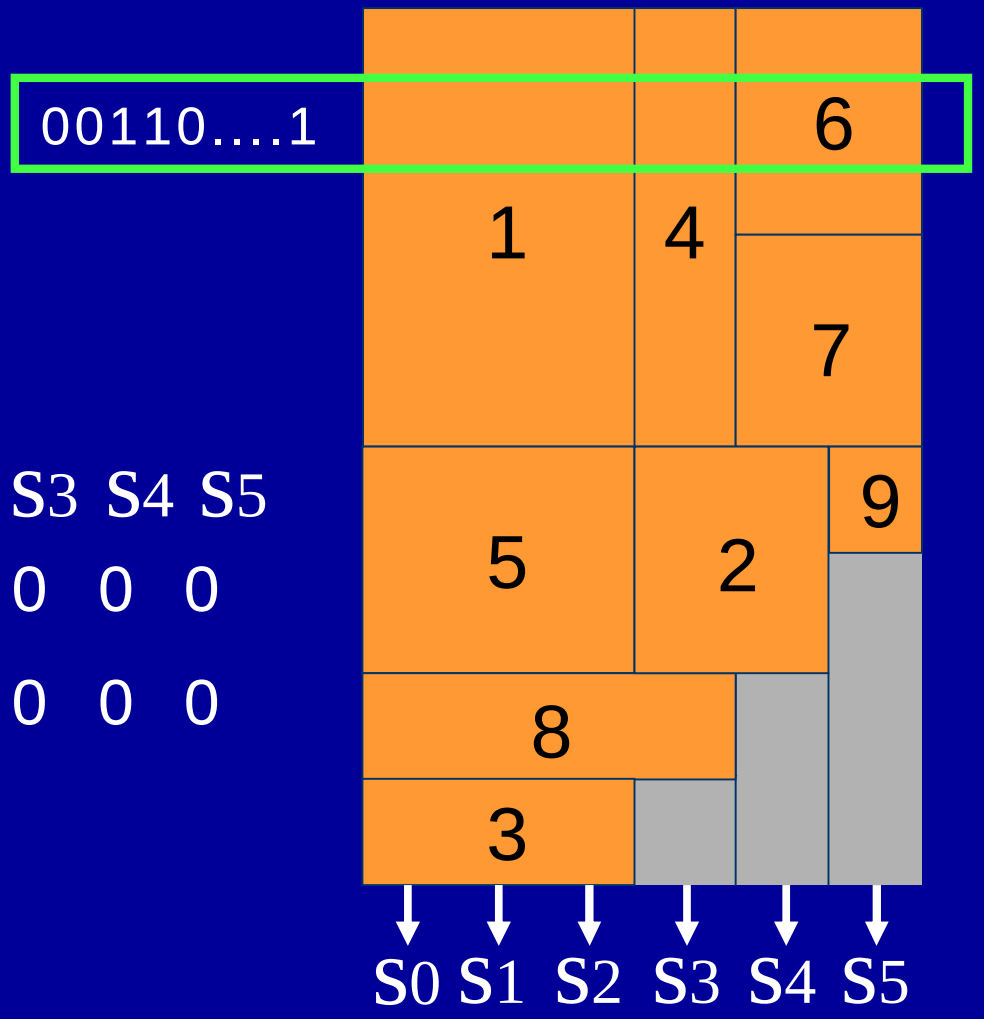
00000....0
OR
110....1

Address



Example – Reading cell data



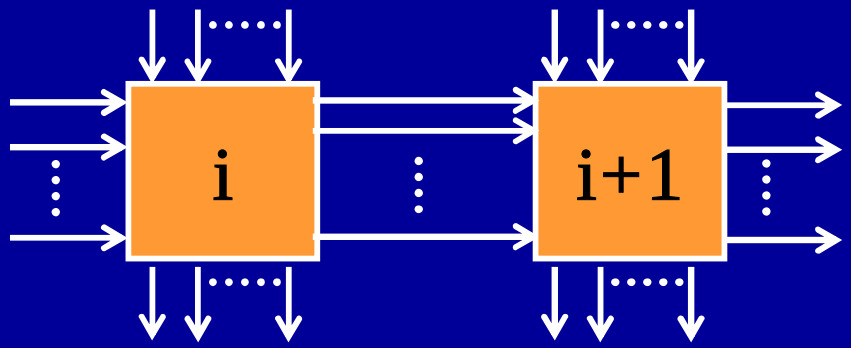


	S0	S1	S2	S3	S4	S5
Rail Outputs	1	1	0	0	0	0
Primary Outputs	0	0	1	0	0	0

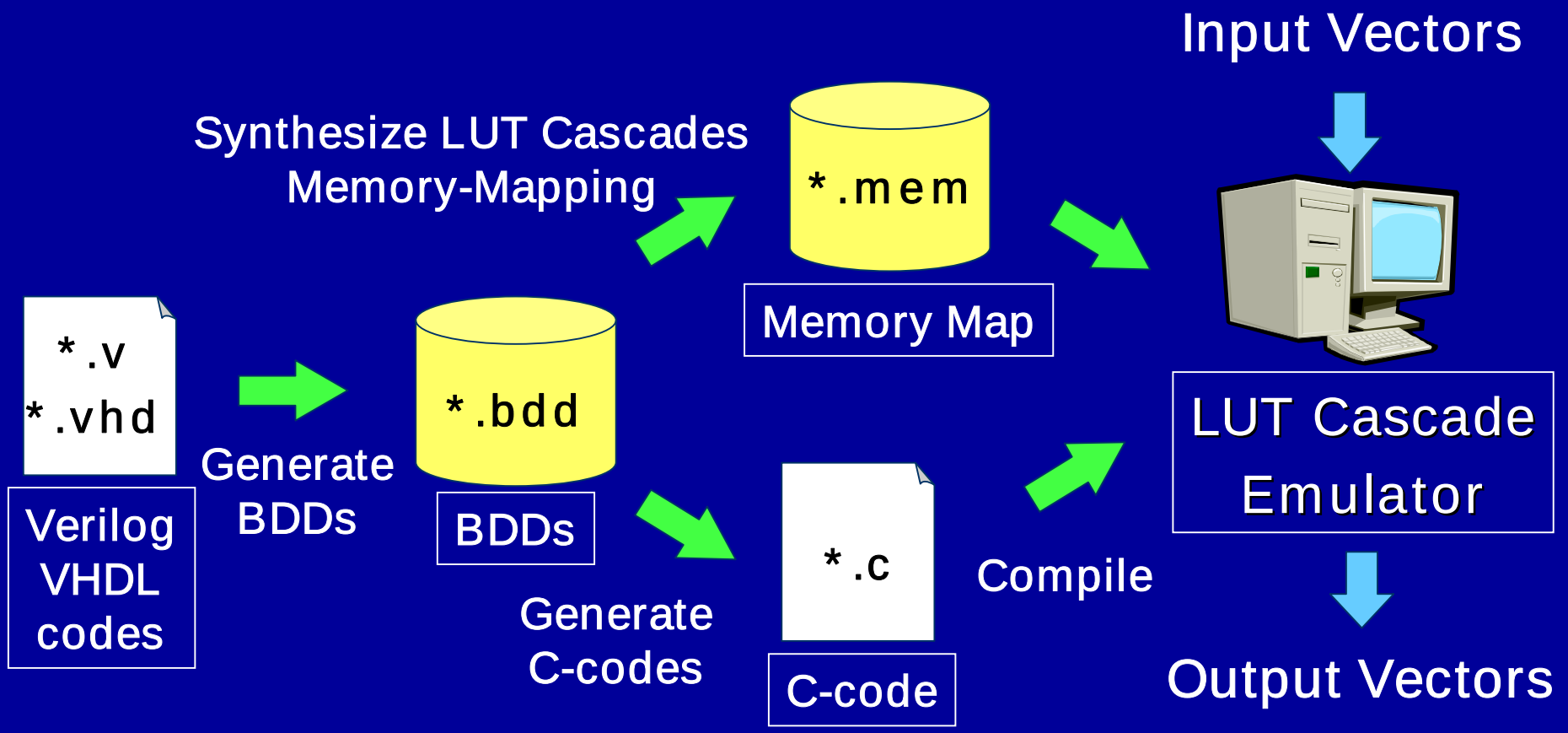
Pseudo-Code

```
/* i th-Cell */
Address ← Page_address[i] | ( Rail_outputs[i-1] | Cell_inputs[i]);
Read_data ← Lut[ Address];
Rail_outputs[i] ← Read_data & Mask_rail[i];
Primary_outputs ← Read_data & Mask_ext[i] << Shift_ext[i];
State_outputs ← Read_data & Mask_state[i] >> Shift_state[i];
```

```
/* i+1 th-Cell */
```



Data Flow of the Logic Simulation System



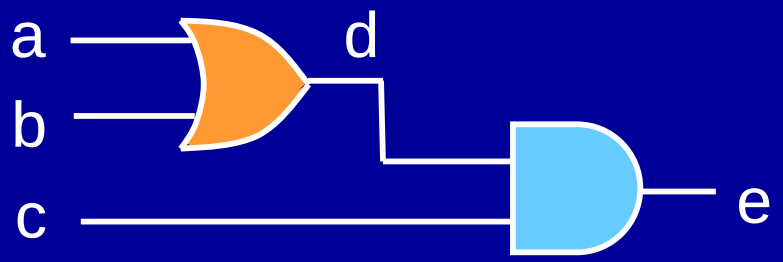
Experiment: Comparison with LCC

- Selected from MCNC benchmark functions
  - Simulation Time
  - Simulation Setup Time
    (BDD+Cascade+Compilation+Memory-Mapping)
- Environment
  - CPU: Pentium4 Xeon 2.8GHz
    (L1 Cache: 12kμops, L2 Cache: 512KB)
  - Memory: 4GByte
  - OS: Redhat Linux 7.3
  - Compiler: gcc 3.2.2
  - Optimize option: -O3

Levelized Complied Code (LCC)

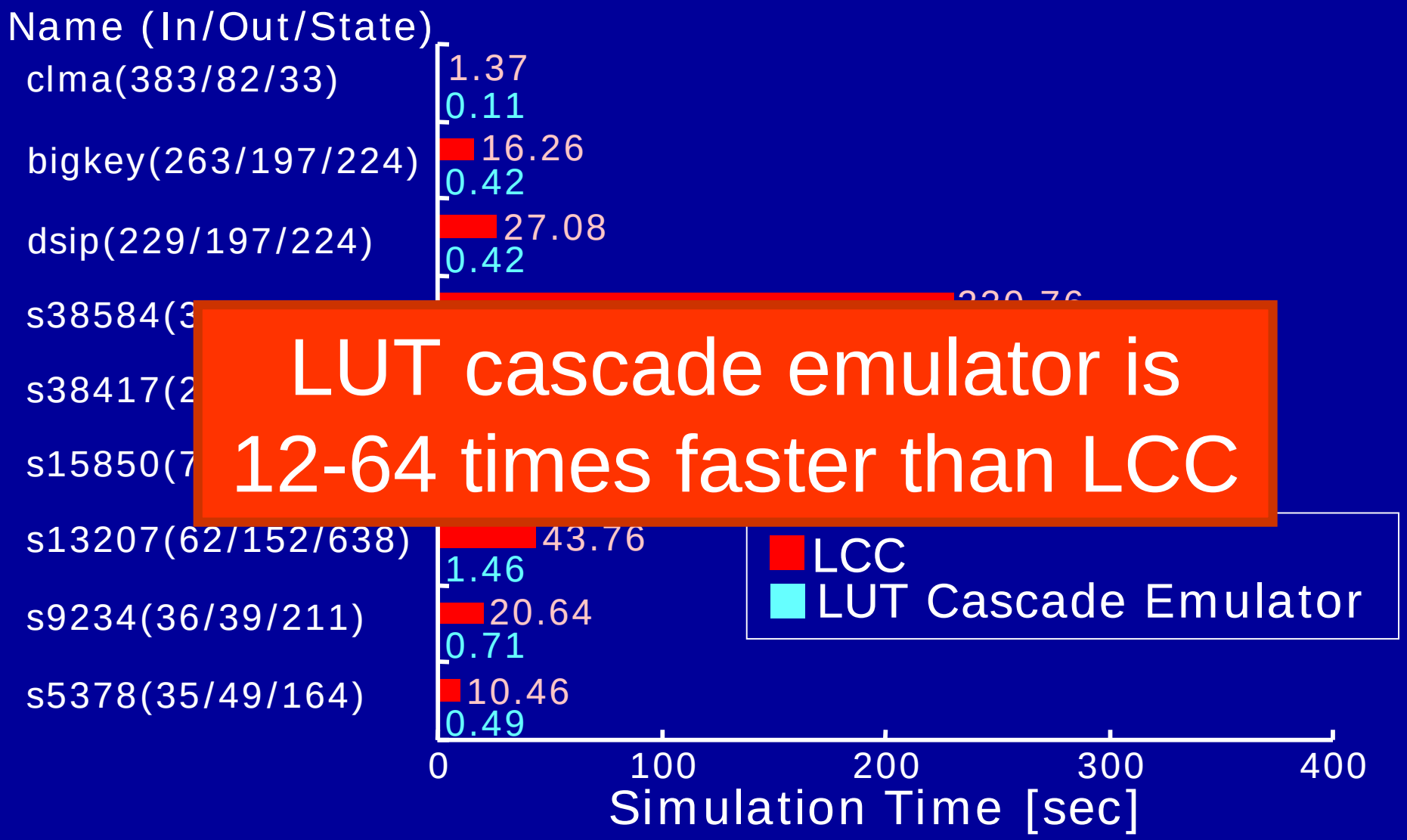
- Cycle-based logic simulator
- LCC generates a C-code for each gate
- Evaluates the circuit in a topological order
- Netlist → C-code → Execution-code

LCC Compile



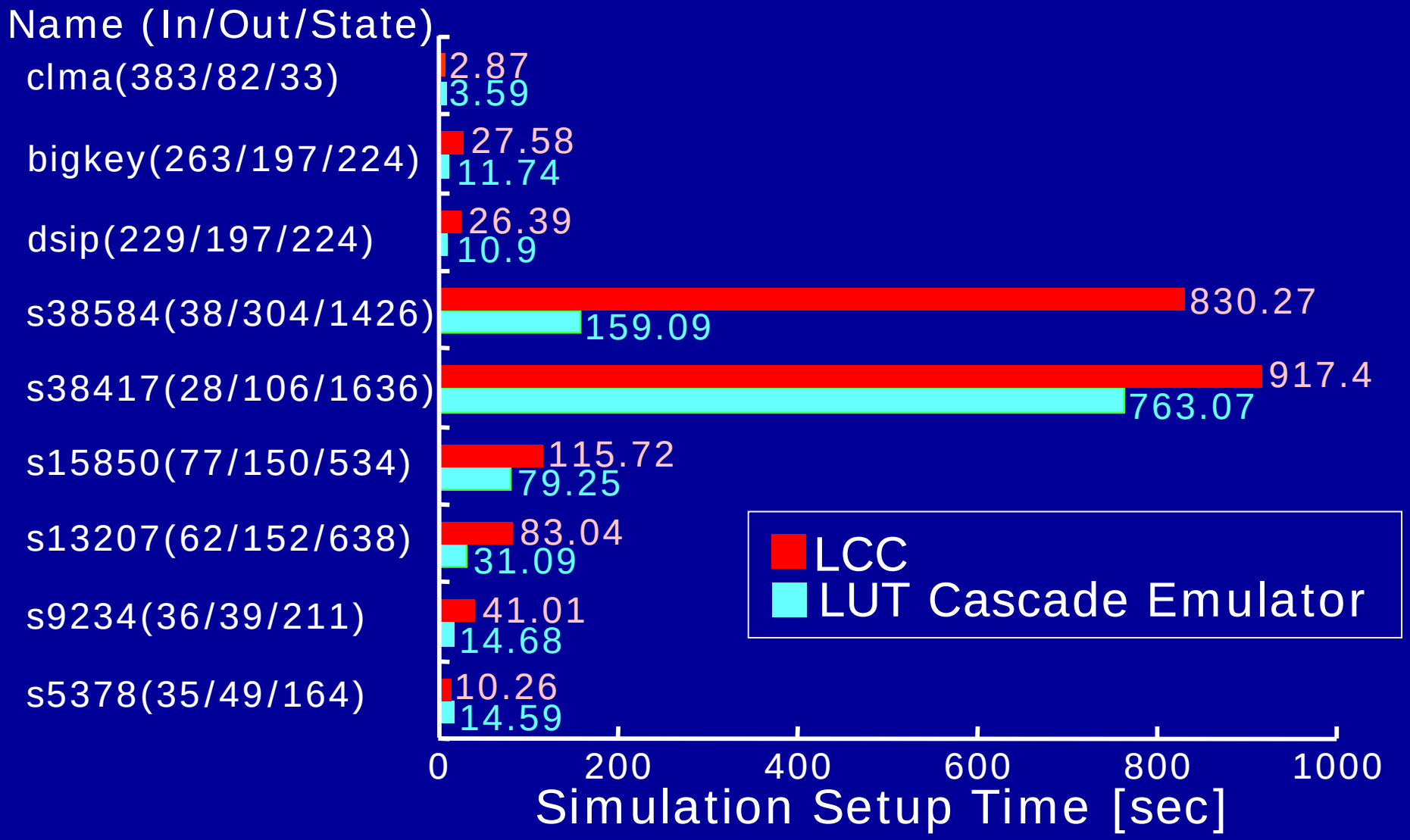

```
d = a | b;
e = d & c;
```

Simulation Time



LUT cascade emulator is
12-64 times faster than LCC

Simulation Setup Time



Estimation of Simulation Time for LUT Cascade Emulator

EST.LUT = EXT.in×Cell+Cell+P.out+S.out+Rail

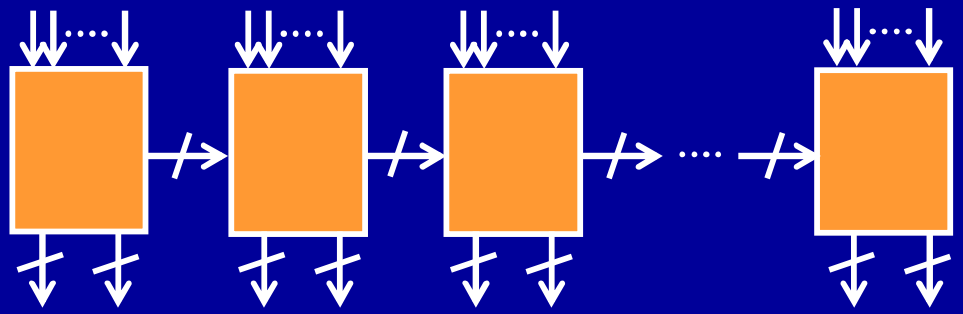
EXT.in :average number of external inputs

Cell :number of cells

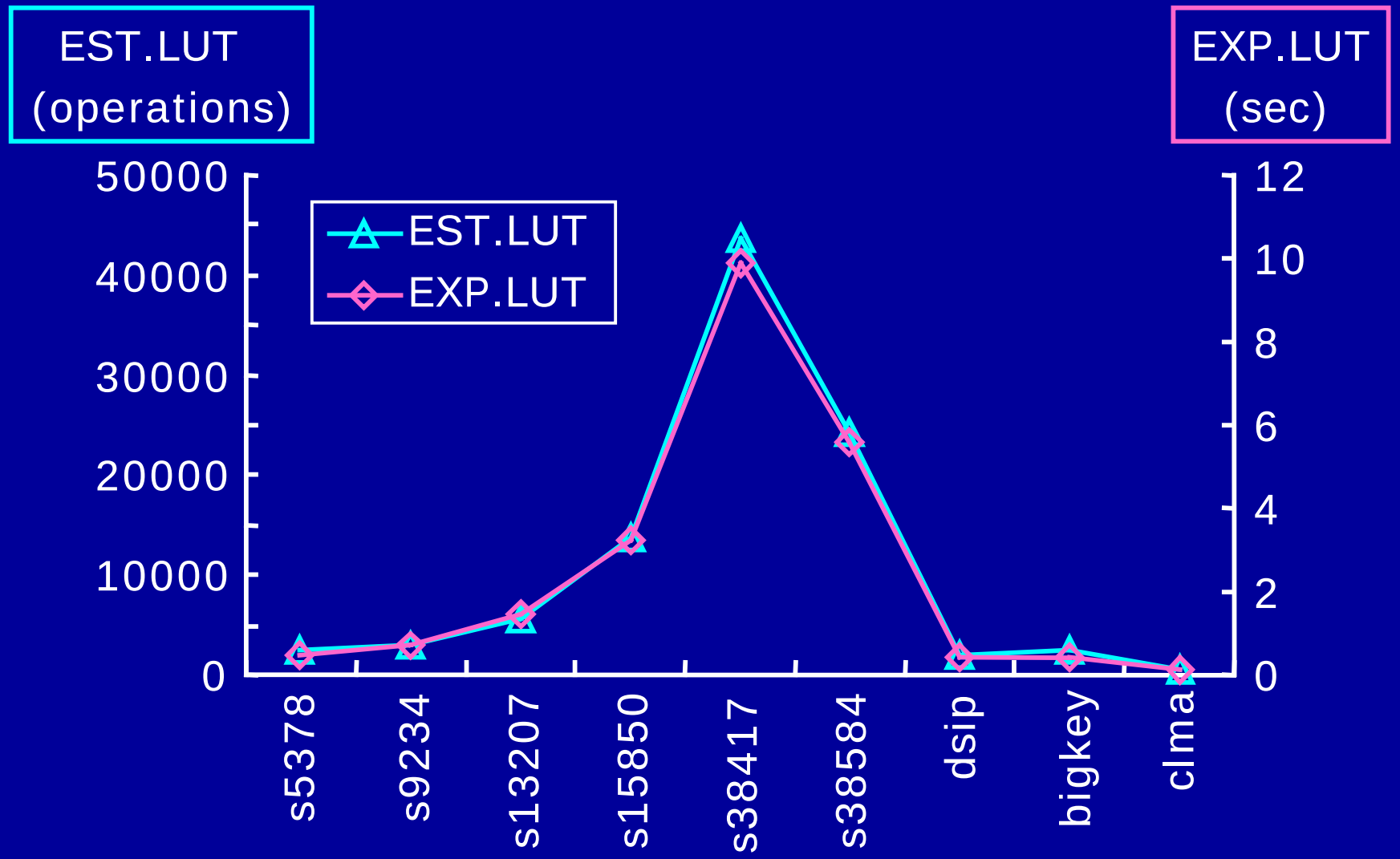
P.out :number of cells with external outputs

S.out :number of cells with state outputs

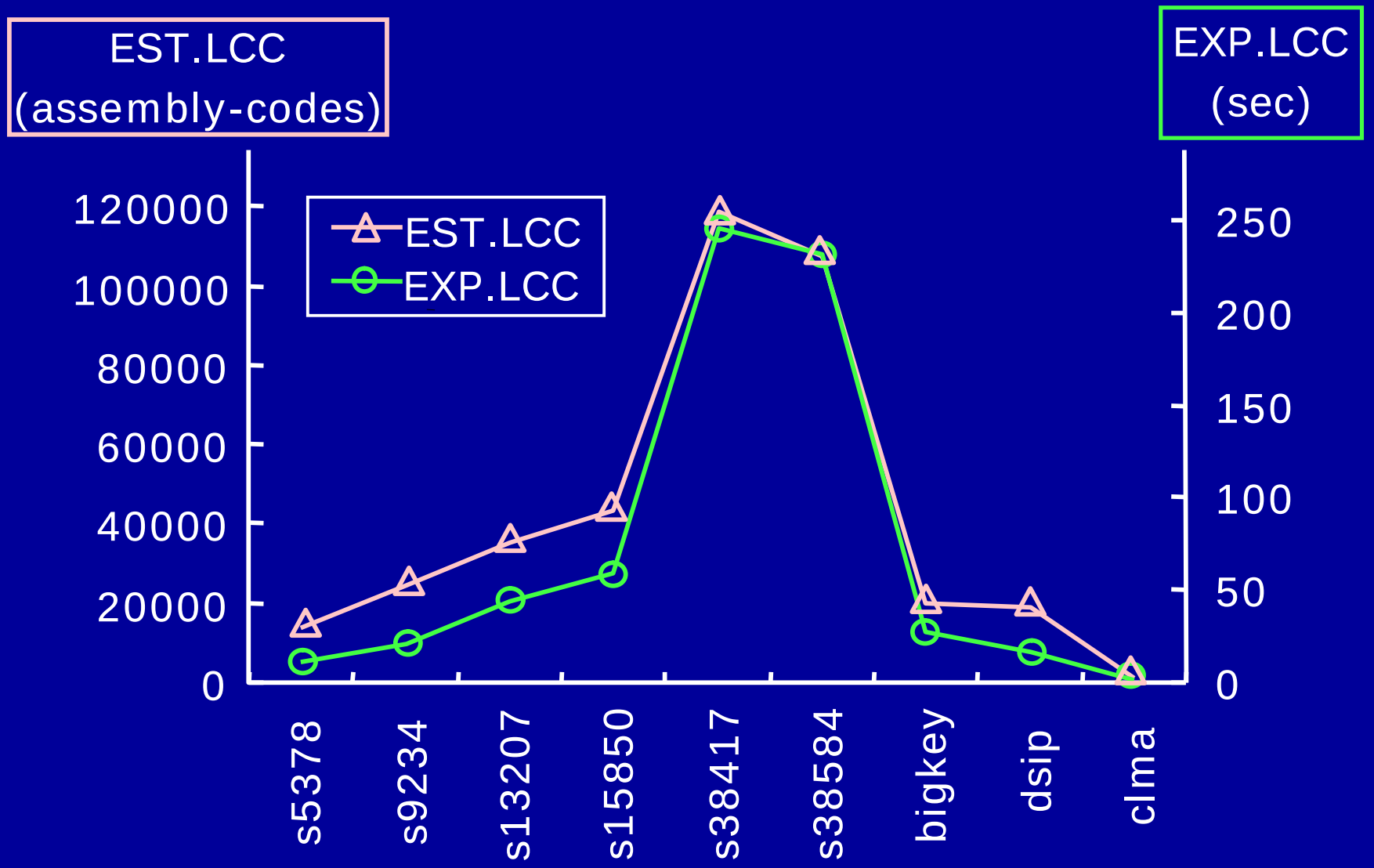
Rail :number of rail outputs



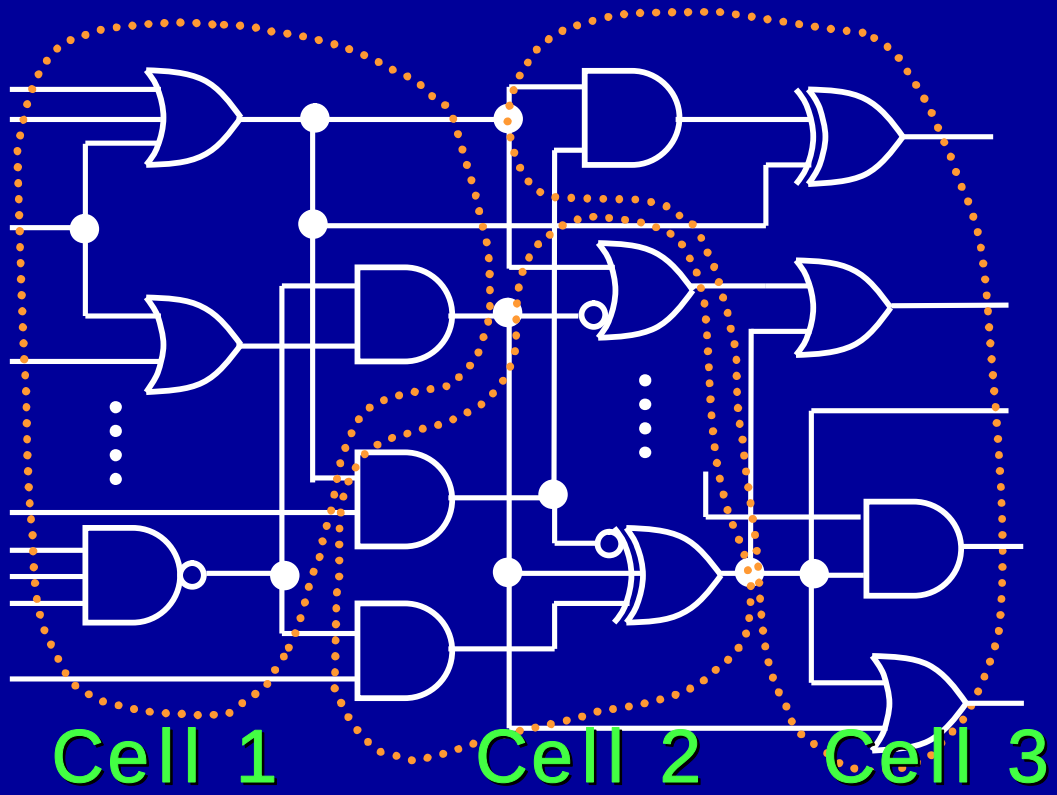
Simulation Time for LUT Cascade Emulator



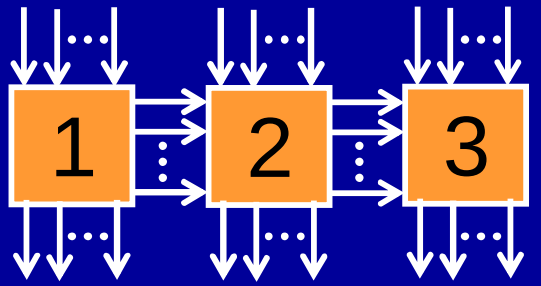
Simulation Time for LCC



Difference of the Representations



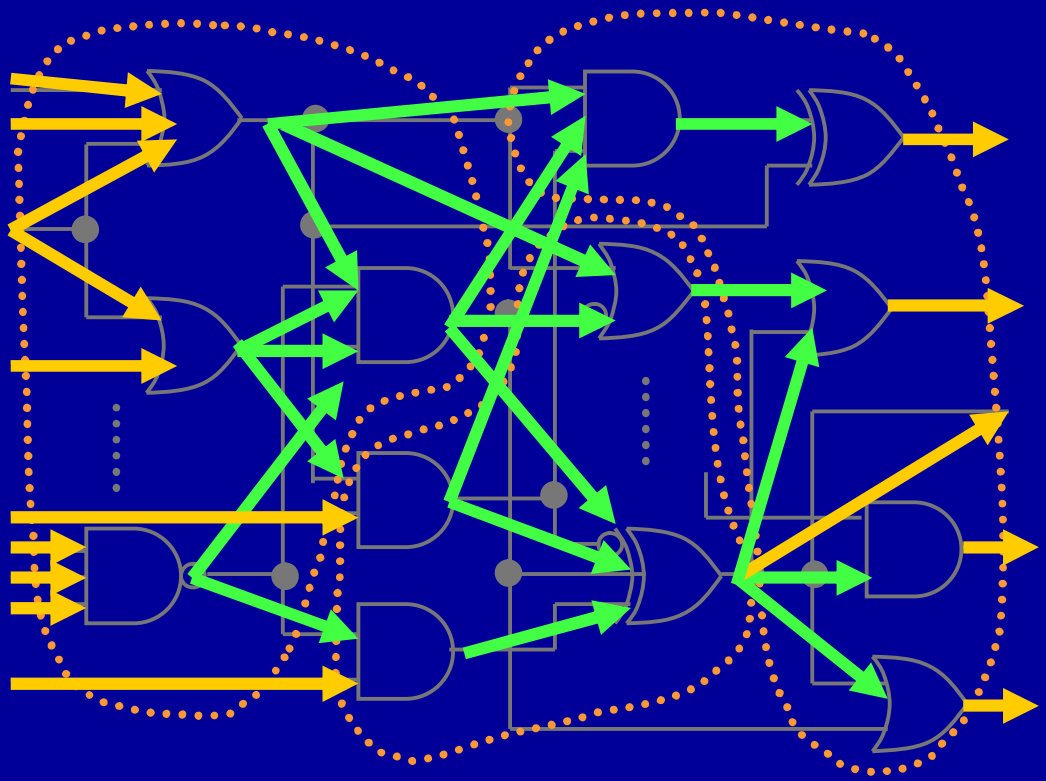
LCC

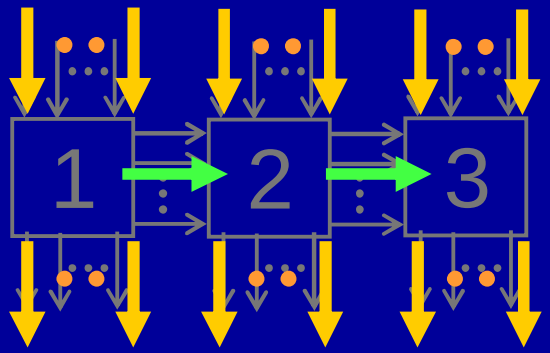


LUT Cascade

Advantage of the Cascade Structure

- Cascade structure has fewer signals than the random logic network
  - Fewer cache miss in the LUT cascade emulator



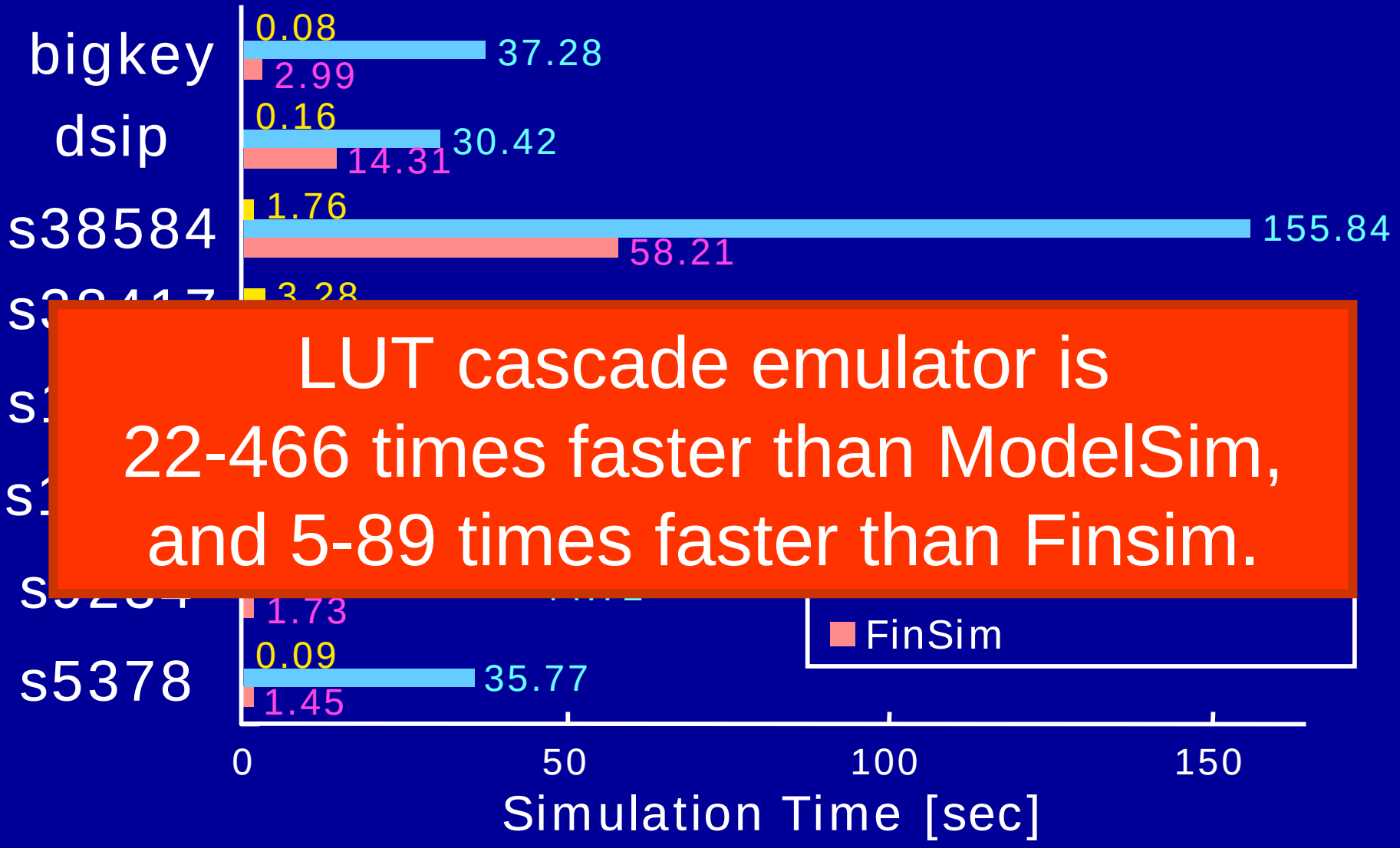


...correspond to variables for program codes

Comparison with Commercial Tools

- ModelSim 6.0c (AE) Menter Graphics Corp.
- Super-Finsim 6.2.9 Fintronic USA, Inc.
- LUT Cascade Emulator
  (on the cygwin.dll 1.3.22)
- Environment
  - CPU: Pentium4 3.06GHz
    (L1 Cache: 12kμops, L2 Cache: 512KB)
  - Memory: 2GByte
  - OS: Windows XP Professional SP2

Simulation Time

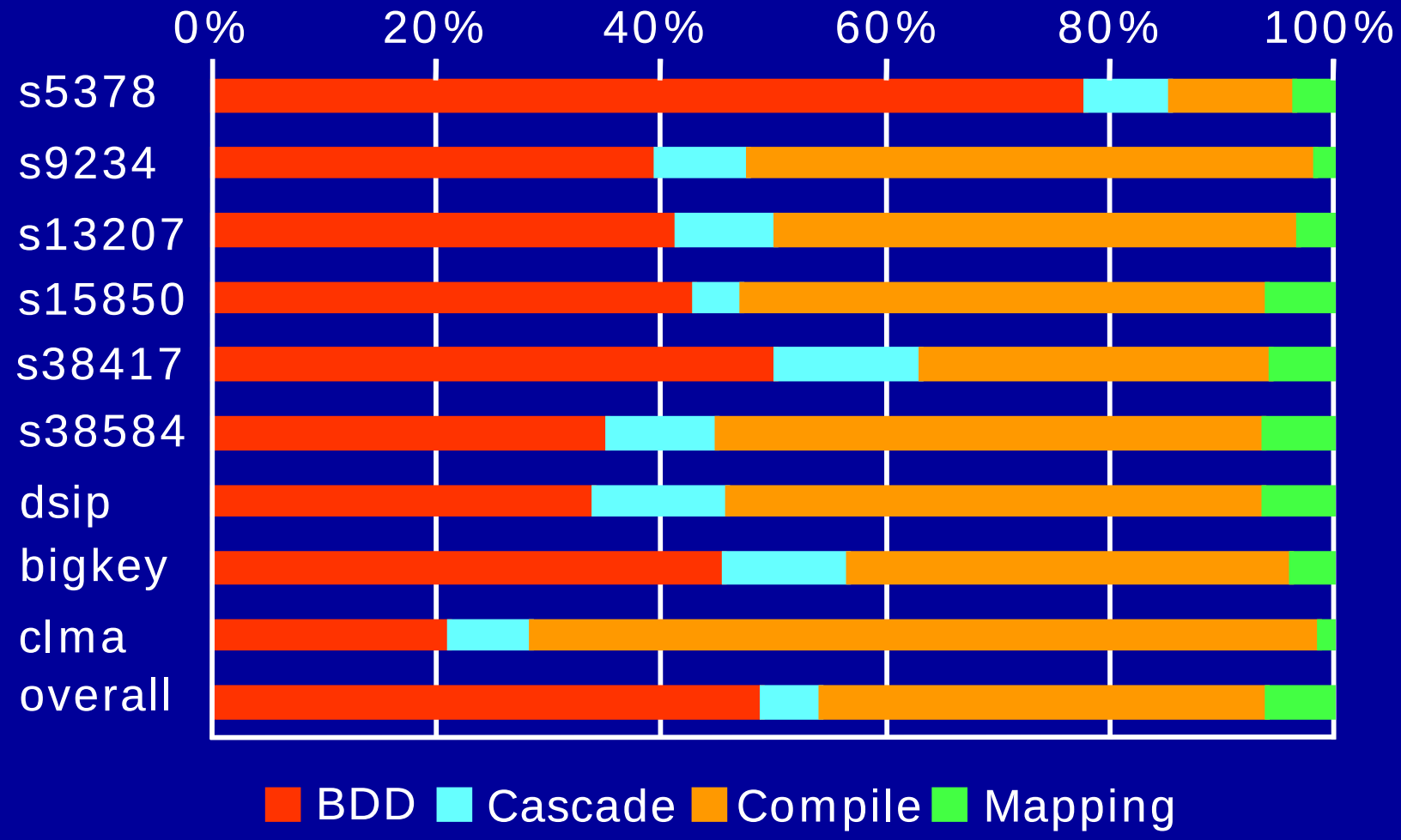


LUT cascade emulator is
22-466 times faster than ModelSim,
and 5-89 times faster than Finsim.

Conclusion

- Logic simulator using the LUT cascade emulator
  - Cycle-based
  - Runs on a standard PC
- Our method is 20-40 times faster than conventional cycle-based logic simulator
  - Suitable for a PC
- Future projects
  - Develop a partition method for a large circuit and to represent the circuits by smaller BDDs

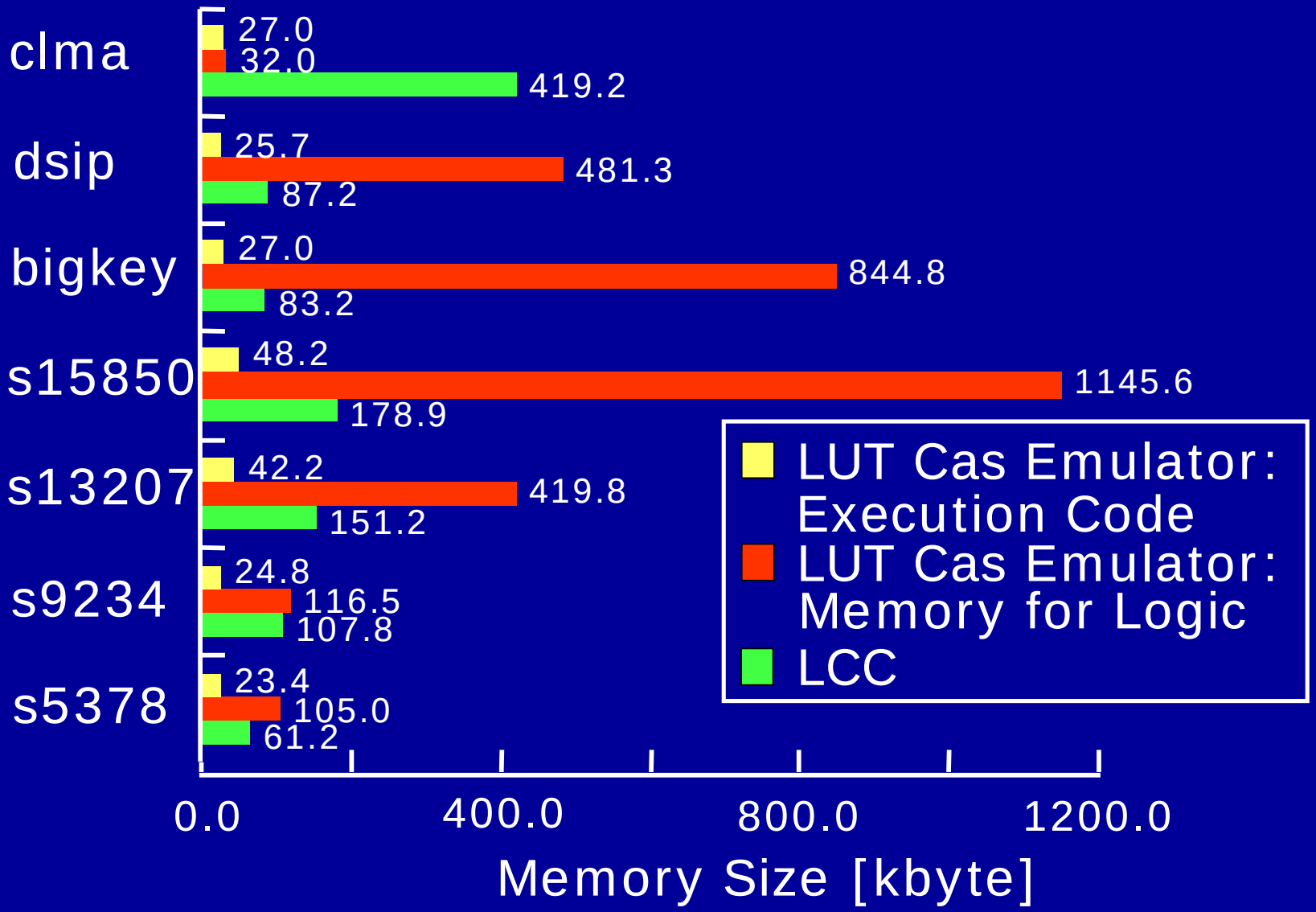
Setup Time for the LUT Cascade Emulator



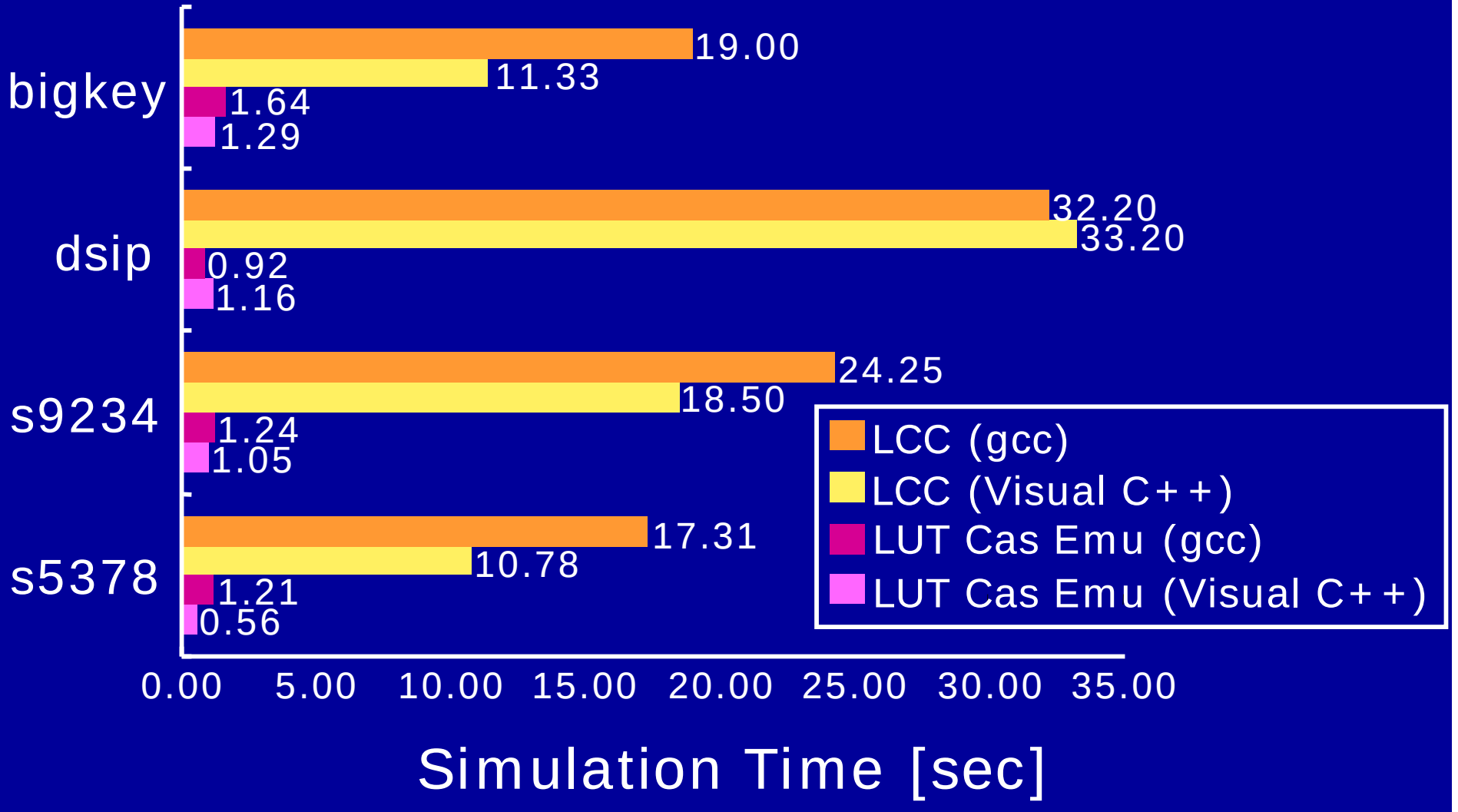
Realization of Benchmark Functions

Name	In/Out/State	LUT Cascade Emulator			LCC	
		Mem [Mbit]	Setup [sec]	Sim [sec]	Setup [sec]	Sim [sec]
s5378	35/49/164	0.82	14.59	0.49	10.26	10.46
s9234	36/39/211	0.91	14.68	0.71	41.01	20.64
s13207	62/152/638	3.28	31.09	1.46	83.04	43.76
s15850	77/150/534	8.95	79.25	3.25	115.72	58.71
s38417	28/106/1636	45.36	763.07	9.87	917.40	245.63
s38584	38/304/1426	16.90	159.09	7.61	830.27	230.76
dsip	229/197/224	6.60	10.90	0.42	26.39	27.08
bigkey	263/197/224	3.76	11.74	0.42	27.58	16.26
clma	383/82/33	0.25	3.64	0.13	207.76	340.75
total			1.00	1.00	2.35	33.46

Memory Size


clma
27.0
32.0
419.2
dsip
25.7
481.3
87.2
bigkey
27.0
844.8
83.2
s15850
48.2
1145.6
178.9
s13207
42.2
419.8
151.2
s9234
24.8
116.5
107.8
s5378
23.4
105.0
61.2
LUT Cas Emulator: Execution Code
LUT Cas Emulator: Memory for Logic
LCC
0.0
400.0
800.0
1200.0
Memory Size [kbyte]

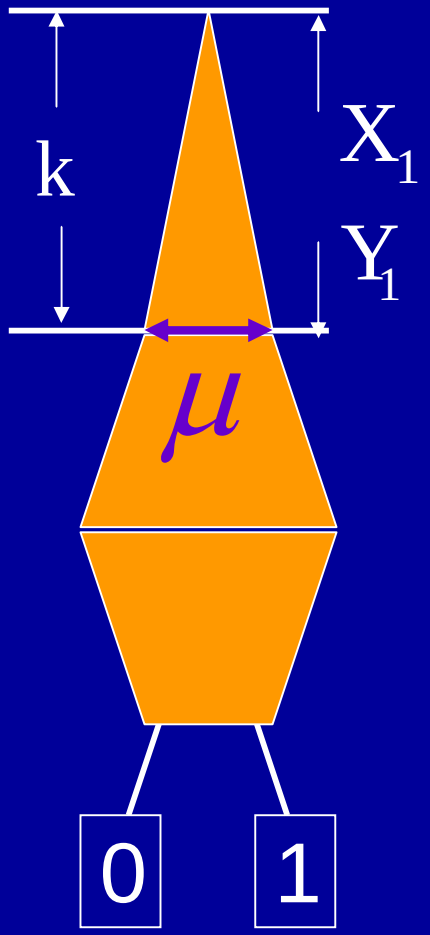
Difference of Compilers



Comparison with Commercial Tools

Name	Simulation Time [sec]		
	FinSim	ModelSim	LUT Cas Emu
s5378	1.45	35.77	0.09
s9234	1.73	44.72	0.11
s13207	6.59	46.50	0.17
s15850	14.84	68.63	0.58
s38417	17.93	72.70	3.28
s38584	58.21	155.84	1.76
dsip	14.31	30.42	0.16
bigkey	2.99	37.28	0.08
ratio	32.69	245.33	1.00

Functional Decomposition

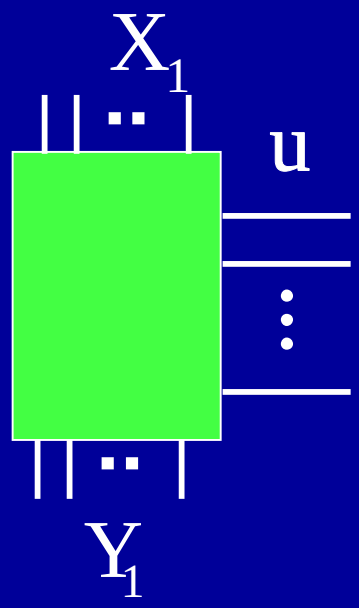


# of external inputs:k

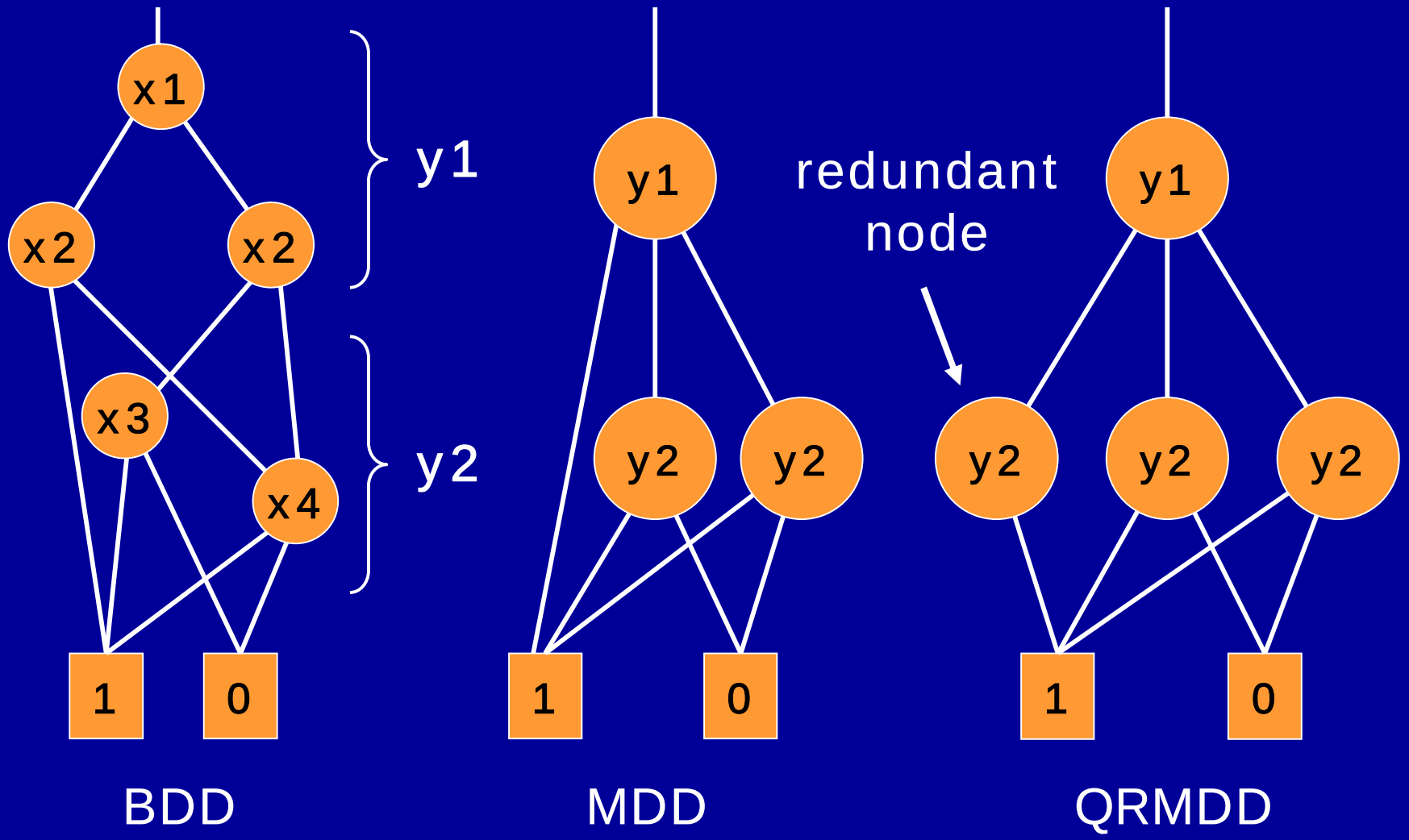
Width of BDD: μ

# of rail outputs: $\lceil \log_2 \mu \rceil = u$

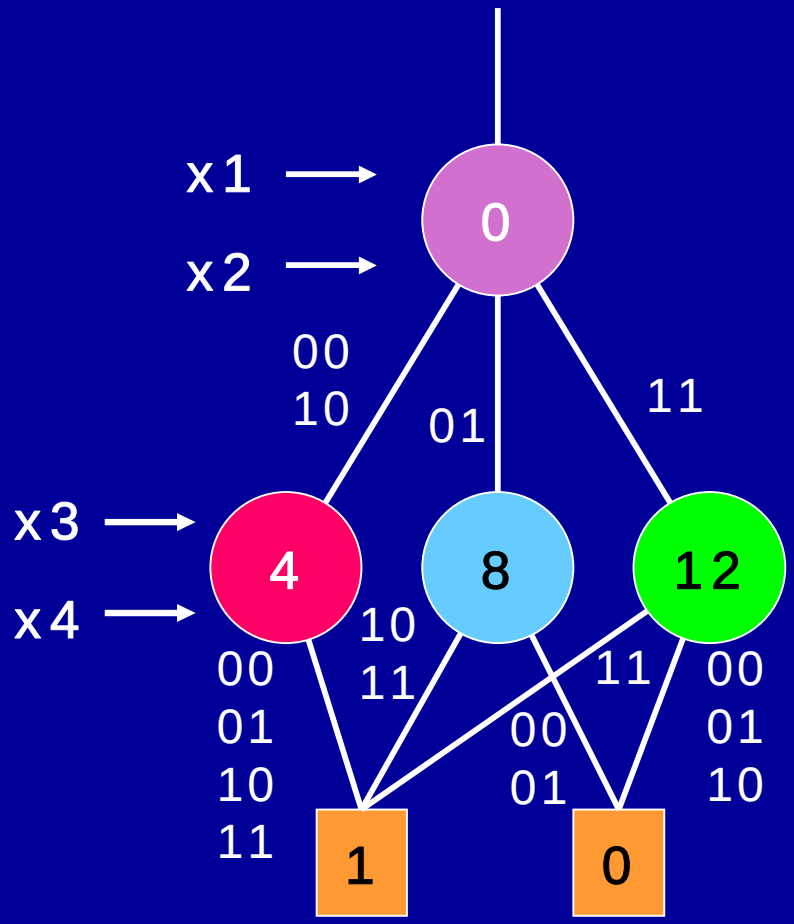
Output variables: Y_1



Comparison with Decision Diagrams



Reference Table for QRMDD



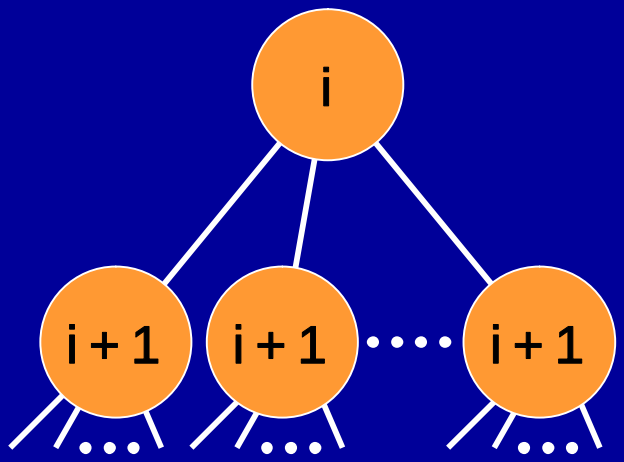
inputs for each node

	00	01	10	11
0	4	8	4	12
4	1	1	1	1
8	0	0	1	1
12	0	0	0	1

Pseudo-Code for QRMDD

```
/* i th-Node */
mdd_input ← input[i] & mask[i];
mdd_var ← Table[ ptr + mdd_input];
ptr ← mdd_var & Mask_ptr[i];
Primary_outputs ← mdd_var & Mask_out[i] << Shift_out[i];
State_outputs ← mdd_var & Mask_state[i] >> Shift_state[i];

/* i+1 th-Node */
```



Results (1) : Simulation Time [sec]

Benchmark			Simulation Time [sec]			Ratio	
Name	In	Out	LCC	QRMDD	LUT Cas Emu	v.s. LCC	v.s. QRMDD
C880	60	26	2.83	1.41	1.16	2.44	1.22
C1908	33	23	4.95	0.76	0.55	9.00	1.38
C3540	50	22	13.43	6.32	1.13	11.88	5.59
apex1	45	45	24.97	0.38	0.28	89.18	1.36
rot	135	107	103.61	3.17	2.23	46.46	1.42
too large	38	3	8.66	0.08	0.06	144.33	1.33
vda	17	39	8.68	0.14	0.10	86.80	1.40
x1	51	35	7.79	0.44	0.37	21.05	1.19

Results (2) : Memory Size [kbyte]

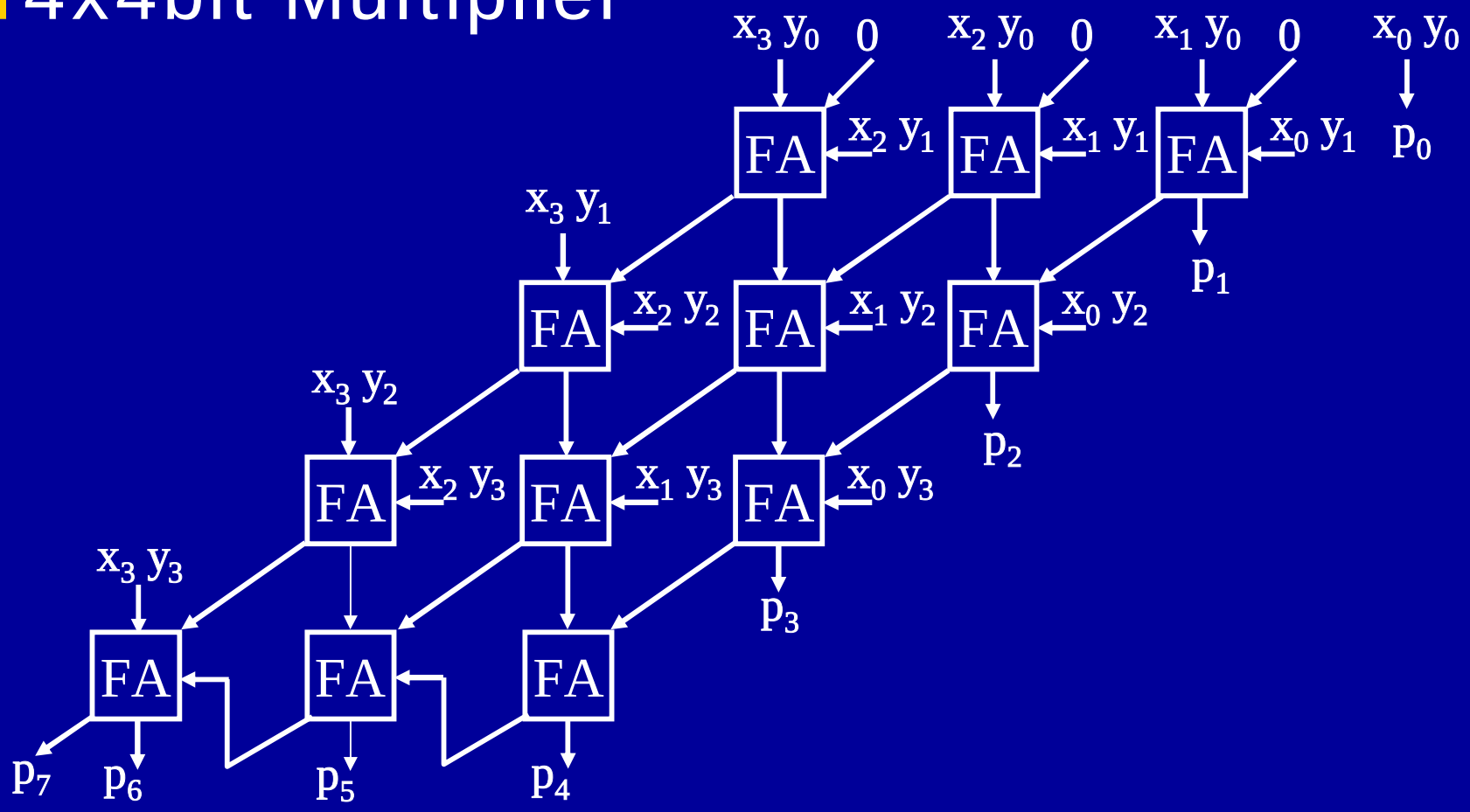
Benchmark			LCC	QRMDD		LUT Cas Emu	
Name	In	Out		code	data	code	data
C880	60	26	27.1	22.7	193.4	18.8	128.0
C1908	33	23	31.1	18.9	115.8	18.8	64.0
C3540	50	22	51.1	23.1	1275.7	18.8	1024.0
apex1	45	45	59.1	18.7	73.8	14.8	32.0
rot	135	107	183.1	26.7	395.3	22.8	512.0
too large	38	3	35.4	14.7	5.6	14.8	8.0
vda	17	39	35.1	14.7	10.2	14.8	8.0
x1	51	35	35.1	18.7	132.7	15.6	64.0

Results (3) : Parameters

Benchmark			QRMDD		LUT Cascade Emulator	
Name	In	Out	average Width	#Path	average rail	# Cell
C880	60	26	65.1	190	5.4	88
C1908	33	23	56.0	132	5.6	72
C3540	50	22	335.9	243	8.1	128
apex1	45	45	52.7	73	6.3	43
rot	135	107	92.9	272	4.1	181
too large	38	3	18.8	19	4.5	12
vda	17	39	28.9	26	4.7	16
x1	51	35	94.2	68	5.8	39

Partition of the Outputs

- 4x4bit Multiplier



Simulation Time for LUT Cascade Emulator