Using the Minimum Set of Input Combinations to Minimize the Area of Local Routing Networks in Logic Clusters Containing Logically Equivalent I/Os in FPGAs

Andy Ye
Ryerson University

Minimum-area IIBs – how to minimize the area of a logic cluster without losing any functionality?

FPGA Logic Clusters

- A Collection of Look-Up Tables and Flip-Flops
- Share a Common Set of Inputs and Outputs
- **Logically Equivalent** Logic Cluster Inputs
  - Any input signal can enter the cluster through any of the logic cluster input pins
- **Logically Equivalent** Logic Cluster Outputs
  - Any output signal can exit the cluster through any of the logic cluster output pins

Benefit and Cost of Logic Equivalency

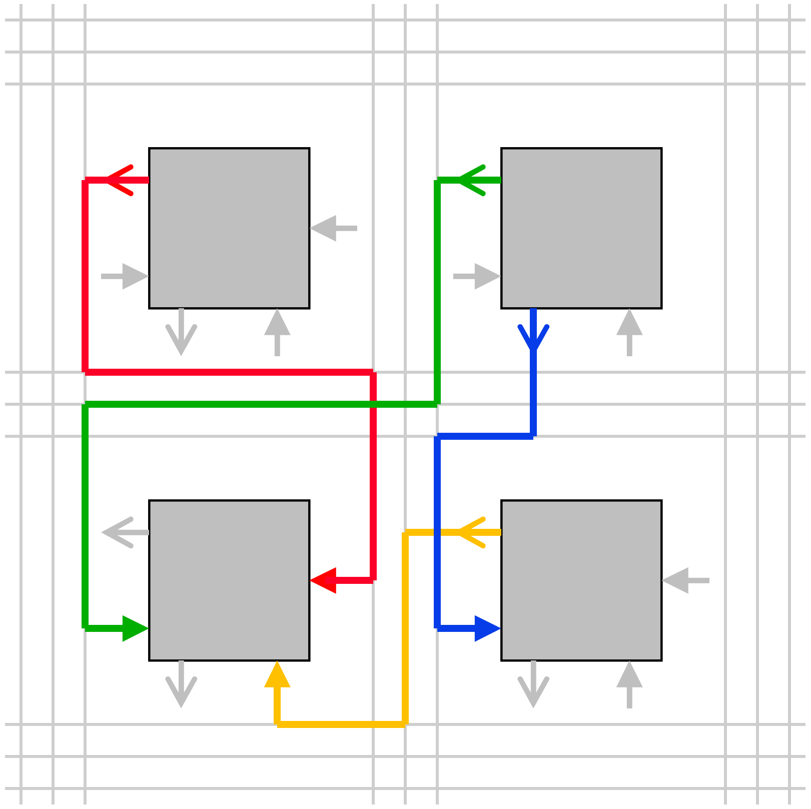
- Benefit: Reduces the Global Routing Area
  - Increases the flexibility of global routing network
  - Less routing tracks
  - Less global routing area

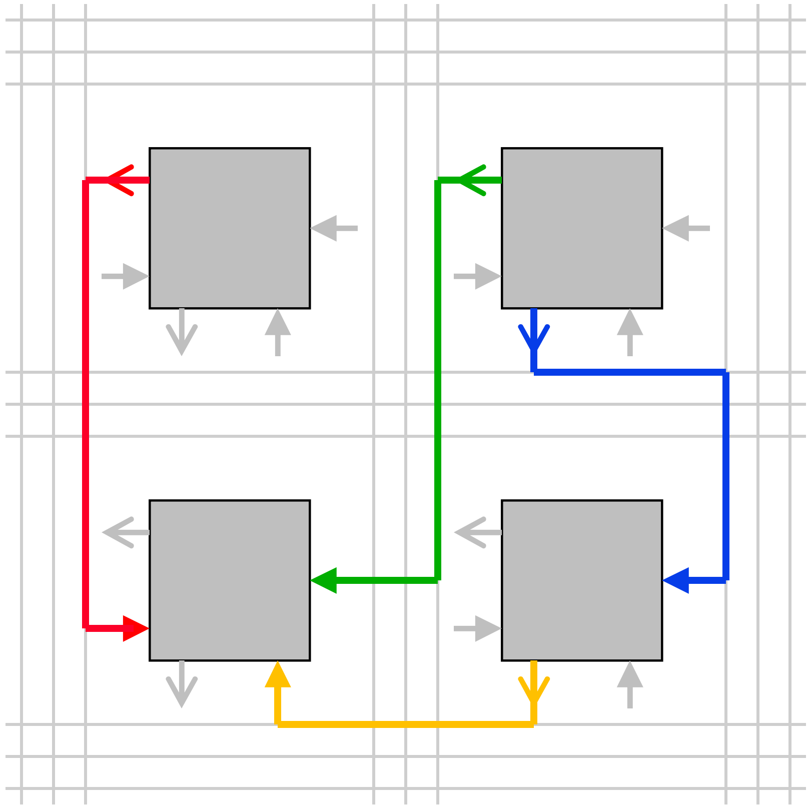
- Cost: Increases Logic Cluster Area
  - Needs specialized local routing networks in every logic cluster
  - Local routing networks cost area

What is a minimum local routing network design that can achieve full logic equivalency?

Benefit: Routing Track Reduction

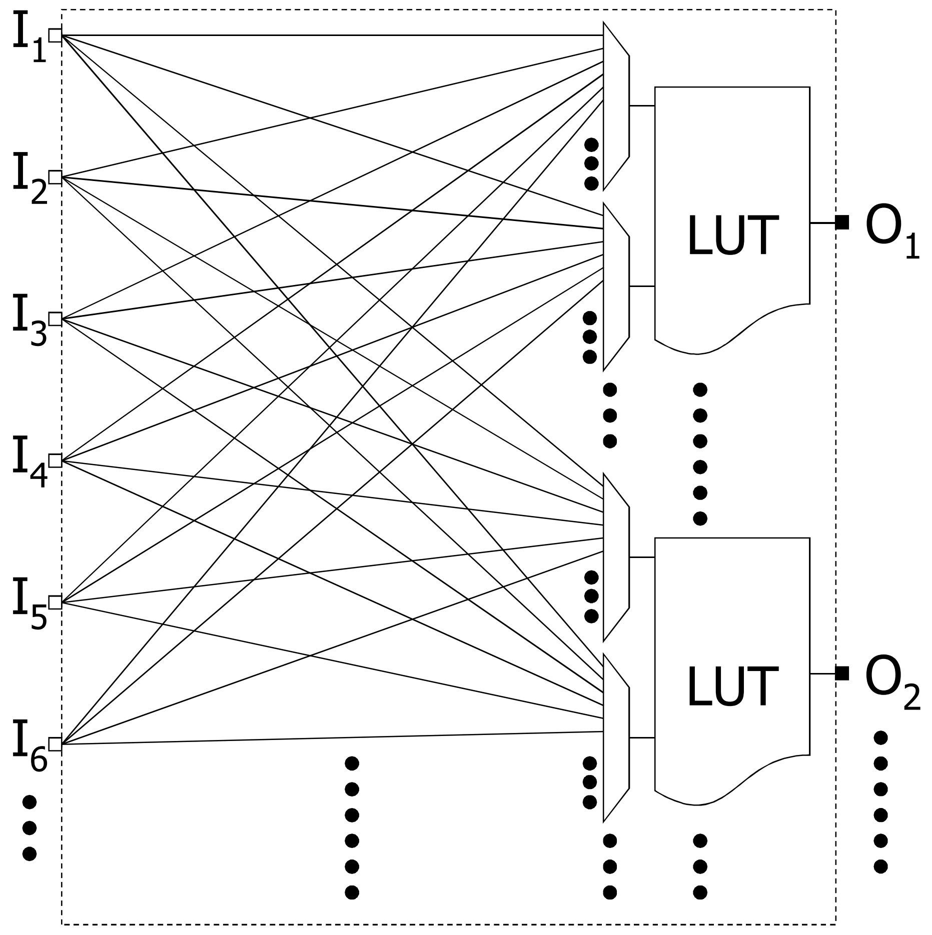
- Consider a set of logic clusters each with 2 output pins and 3 input pins


Logically
Non-Equivalent
I/O Pins


Logically
Equivalent
I/O Pins

Cost: Local Routing Networks

- A Fully Connected Local Routing Network



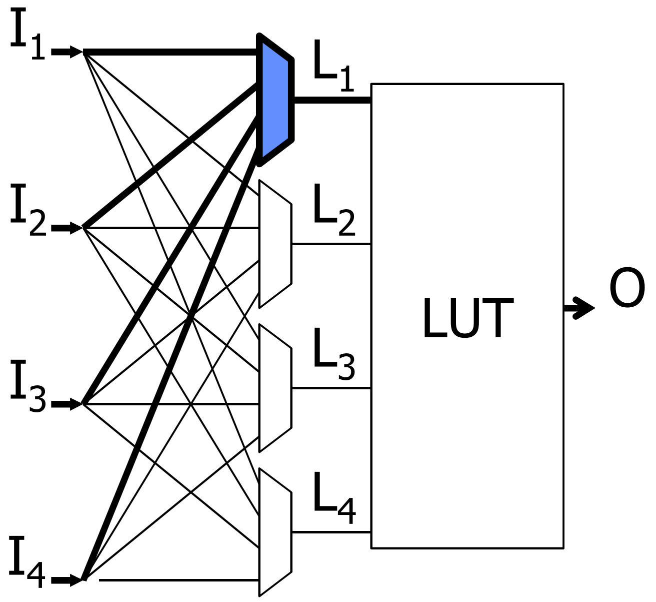
Less Than Full Connectivity?

- Yes!

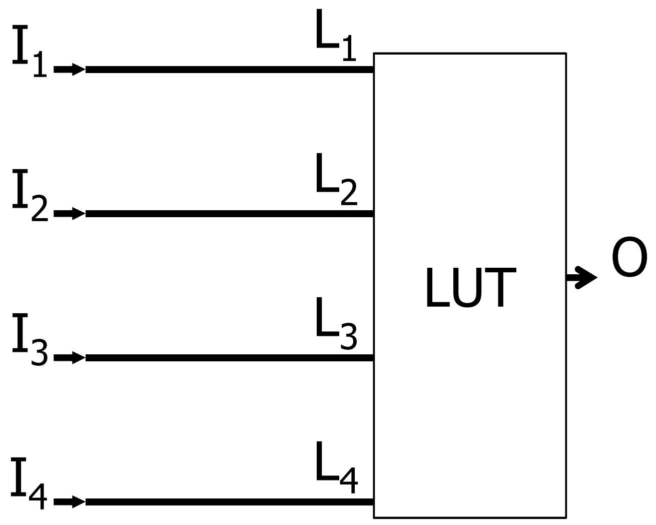
- A Simple Example [Betz and Rose 98]
  - Logic clusters containing just **one** LUT each
  - The number of inputs per cluster = the number of LUT inputs
  - Can completely eliminate the local routing network

- Key Mechanism that Enables the Elimination of the Local Routing Network
  - LUT reconfiguration – reconfigure a LUT as the logic cluster input assignment changes

Logic Equivalency Through LUT Reconfiguration – 1 LUT Cluster



Logical Equivalency Through Local Routing Network



Logical Equivalency Through LUT Reconfiguration

Logic Equivalency Through LUT Reconfiguration – 1 LUT Cluster

L1	L2	L3	L4	**O**
0	0	0	0	**f0**
0	0	0	1	**f1**
0	0	1	0	**f2**
0	0	1	1	**f3**
0	1	0	0	**f4**
0	1	0	1	**f5**
0	1	1	0	**f6**
0	1	1	1	**f7**
1	0	0	0	**f8**
1	0	0	1	**f9**
1	0	1	0	**f10**
1	0	1	1	**f11**
1	1	0	0	**f12**
1	1	0	1	**f13**
1	1	1	0	**f14**
1	1	1	1	**f15**

LUT Config. for Connection to Cluster Input 1

L1	**L2**	L3	L4	**O**
0	**0**	0	0	**f0**
0	**0**	0	1	**f1**
0	**0**	1	0	**f2**
0	**0**	1	1	**f3**
0	**1**	0	0	**f8**
0	**1**	0	1	**f9**
0	**1**	1	0	**f10**
0	**1**	1	1	**f11**
1	**0**	0	0	**f4**
1	**0**	0	1	**f5**
1	**0**	1	0	**f6**
1	**0**	1	1	**f7**
1	**1**	0	0	**f12**
1	**1**	0	1	**f13**
1	**1**	1	0	**f14**
1	**1**	1	1	**f15**

LUT Config. for Connection to Cluster Input 2

L1	L2	**L3**	L4	**O**
0	0	**0**	0	**f0**
0	0	**0**	1	**f1**
0	0	**1**	0	**f8**
0	0	**1**	1	**f9**
0	1	**0**	0	**f4**
0	1	**0**	1	**f5**
0	1	**1**	0	**f12**
0	1	**1**	1	**f13**
1	0	**0**	0	**f2**
1	0	**0**	1	**f3**
1	0	**1**	0	**f10**
1	0	**1**	1	**f11**
1	1	**0**	0	**f6**
1	1	**0**	1	**f7**
1	1	**1**	0	**f14**
1	1	**1**	1	**f15**

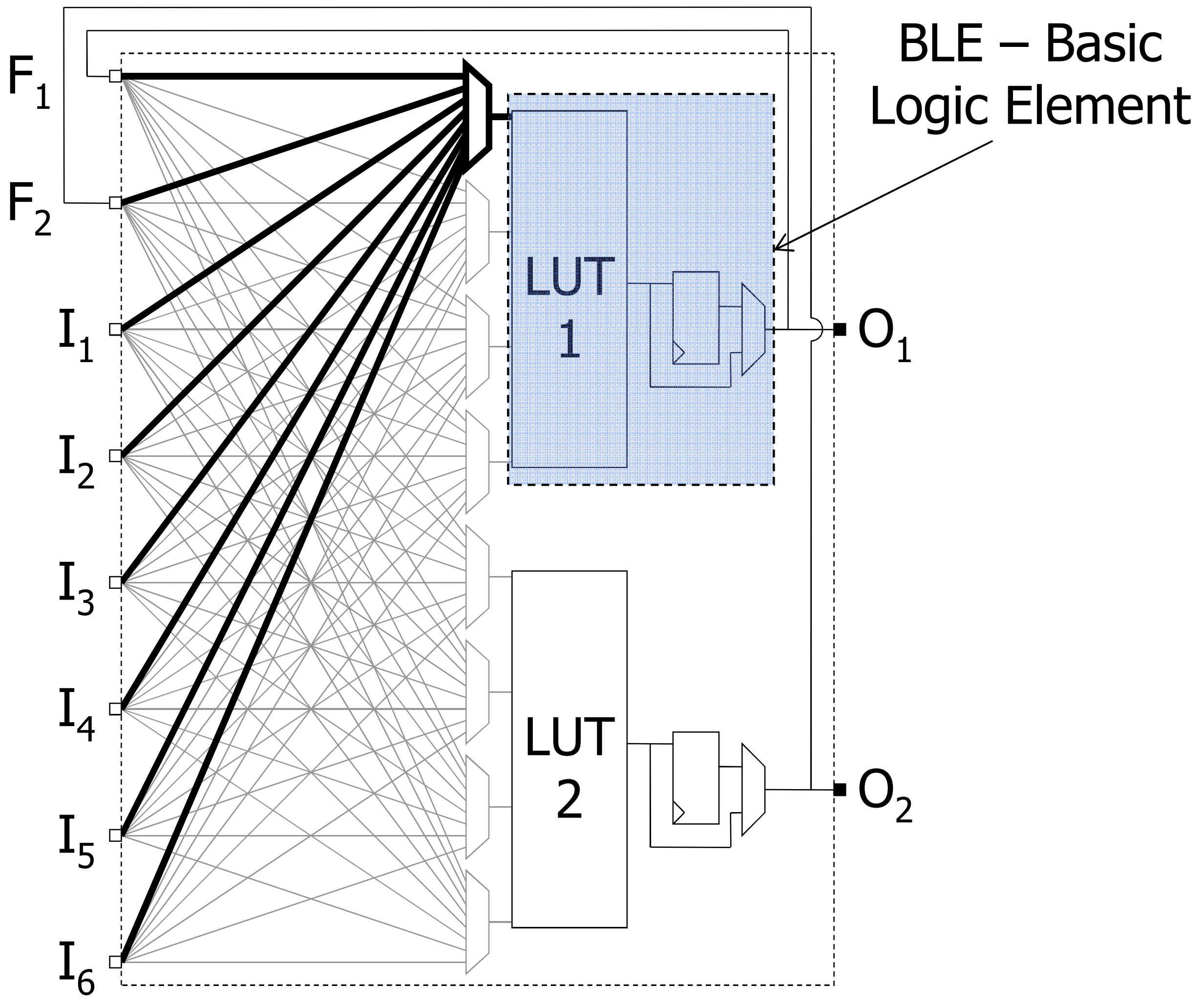
LUT Config. for Connection to Cluster Input 3

L1	L2	L3	**L4**	**O**
0	0	0	**0**	**f0**
0	0	0	**1**	**f8**
0	0	1	**0**	**f2**
0	0	1	**1**	**f10**
0	1	0	**0**	**f4**
0	1	0	**1**	**f12**
0	1	1	**0**	**f6**
0	1	1	**1**	**f14**
1	0	0	**0**	**f1**
1	0	0	**1**	**f9**
1	0	1	**0**	**f3**
1	0	1	**1**	**f11**
1	1	0	**0**	**f5**
1	1	0	**1**	**f13**
1	1	1	**0**	**f7**
1	1	1	**1**	**f15**

LUT Config. for Connection to Cluster Input 4

Question Addressed in This Research

How about clusters with more than 1 LUT?

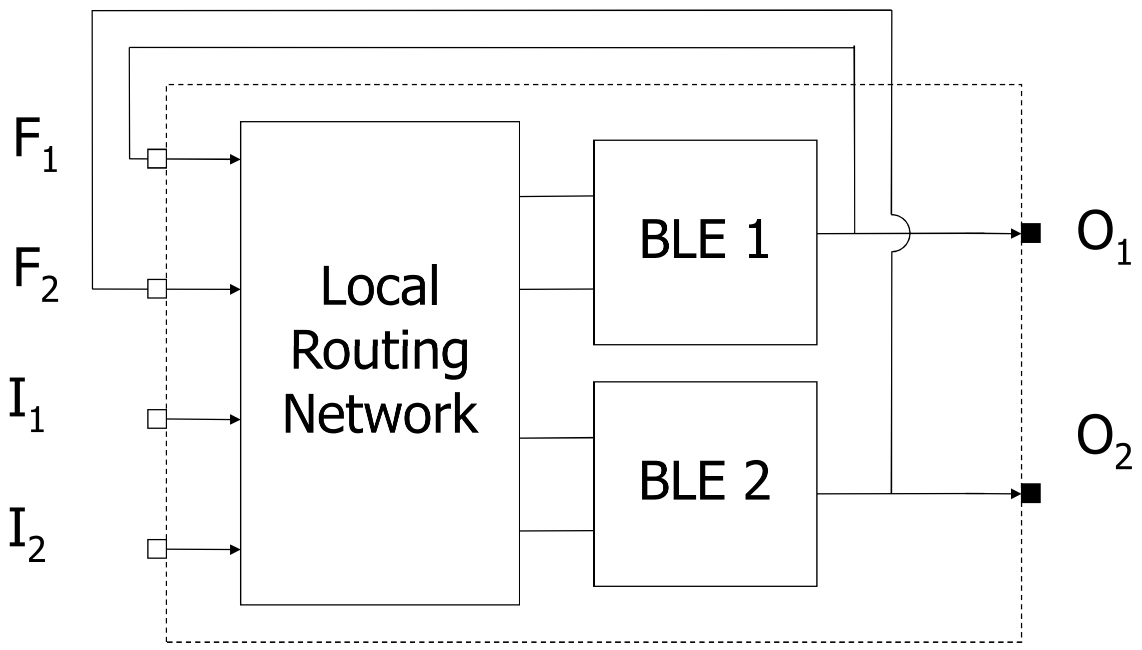


Functions that can be Generated by a Local Routing Network

- A Logic Cluster with
  - k-input LUT
  - I logic cluster inputs
  - N feedbacks
- Fix LUT Configuration
- Reconfigure the Local Routing Network
- Maximum Number of Functions that the LUT can Generate – $(I + N)^k$
- Many of these functions can be made redundant through LUT reconfiguration.

An Example

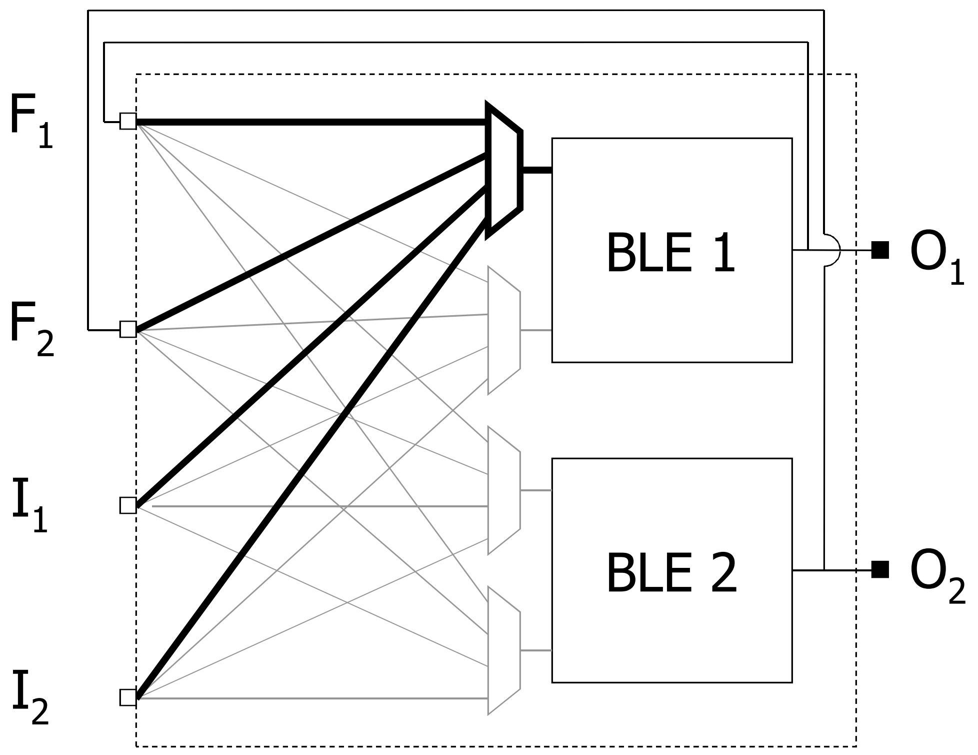
- k = 2, N = 2, I = 2: $(2+2)^2$ = 16 functions



f(F1,F1), f(F1,F2), f(F1,I1), f(F1,I2)
f(F2,F1), f(F2,F2), f(F2,I1), f(F2,I2)
f(I1,F1), f(I1,F2), f(I1,I1), f(I1,I2)
f(I2,F1), f(I2,F2), f(I2,I1), f(I2,I2)

Local Routing Network – 4:1 Muxes

f(F1,F1), f(F1,F2), f(F1,I1), f(F1,I2)
f(F2,F1), f(F2,F2), f(F2,I1), f(F2,I2)
f(I1,F1), f(I1,F2), f(I1,I1), f(I1,I2)
f(I2,F1), f(I2,F2), f(I2,I1), f(I2,I2)



Commutative Property of LUT Inputs

- Given a k-Input LUT configured with the function:
  - $f(a_1, a_2, \ldots, a_k)$
- Connect the LUT to k independent Boolean inputs:
  - $i_1, i_2, \ldots, i_k$
- There exists another function, f', such that:
  - $f'(i_1, i_2, \ldots, i_{x-1}, \boldsymbol{i_y}, i_{x+1}, i_{x+2}, \ldots, i_{y-1}, \boldsymbol{i_x}, i_{y+1}, i_{y+2}, \ldots, i_k)$
  - =
  - $f(i_1, i_2, \ldots, i_{x-1}, \boldsymbol{i_x}, i_{x+1}, i_{x+2}, \ldots, i_{y-1}, \boldsymbol{i_y}, i_{y+1}, i_{y+2}, \ldots, i_k)$

Commutative Property Continued

L1	L2	**L3**	L4	**O**
0	0	**0**	0	**f0**
0	0	**0**	1	**f1**
0	0	**1**	0	**f2**
0	0	**1**	1	**f3**
0	1	**0**	0	**f4**
0	1	**0**	1	**f5**
0	1	**1**	0	**f6**
0	1	**1**	1	**f7**
1	0	**0**	0	**f8**
1	0	**0**	1	**f9**
1	0	**1**	0	**f10**
1	0	**1**	1	**f11**
1	1	**0**	0	**f12**
1	1	**0**	1	**f13**
1	1	**1**	0	**f14**
1	1	**1**	1	**f15**

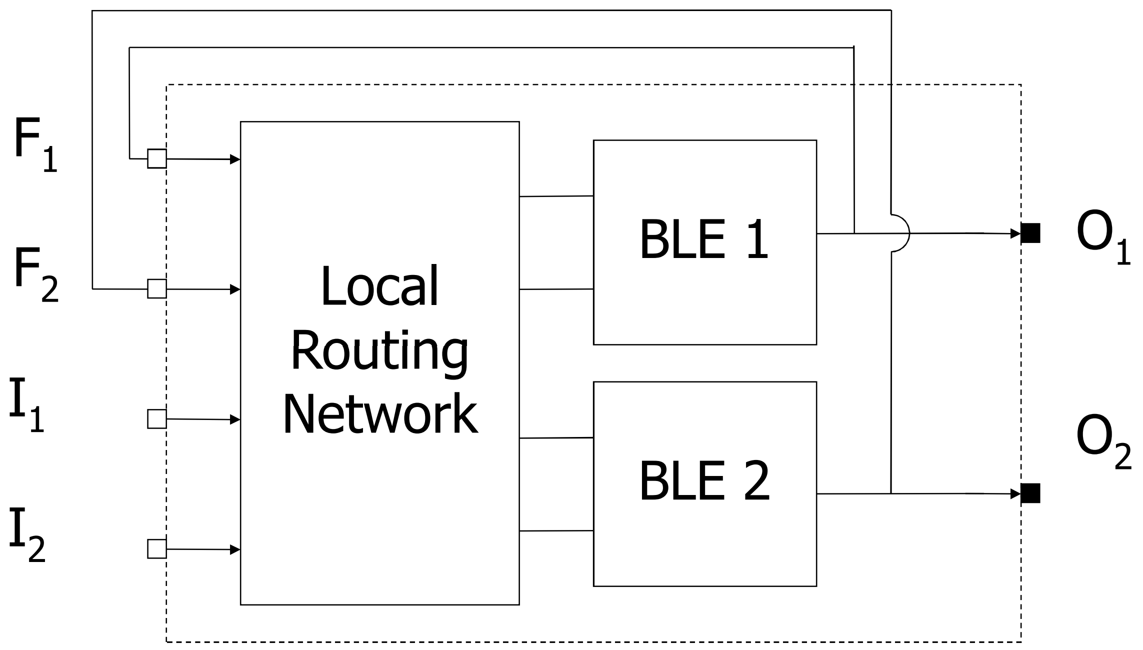
Before Exchanging L1 and L3

L3	L2	**L1**	L4	**O**
0	0	**0**	0	**f0**
0	0	**0**	1	**f1**
0	0	**1**	0	**f8**
0	0	**1**	1	**f9**
0	1	**0**	0	**f4**
0	1	**0**	1	**f5**
0	1	**1**	0	**f12**
0	1	**1**	1	**f13**
1	0	**0**	0	**f2**
1	0	**0**	1	**f3**
1	0	**1**	0	**f10**
1	0	**1**	1	**f11**
1	1	**0**	0	**f6**
1	1	**0**	1	**f7**
1	1	**1**	0	**f14**
1	1	**1**	1	**f15**

After Exchanging L1 and L3

Commutative Property Continued

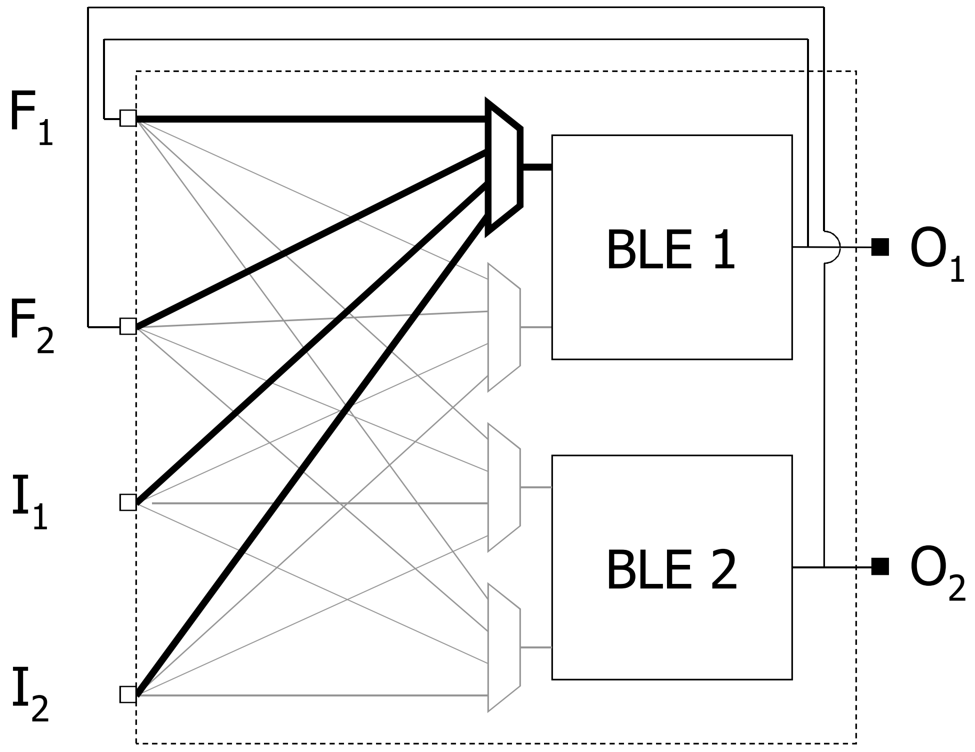
- k = 2, N = 2, I = 2: 16 functions => 10 functions



f(F1,F1),	f(F1,F2),	f(F1,I1),	f(F1,I2)
~~f(F2,F1),~~	f(F2,F2),	f(F2,I1),	f(F2,I2)
~~f(I1,F1),~~	~~f(I1,F2),~~	f(I1,I1),	f(I1,I2)
~~f(I2,F1),~~	~~f(I2,F2),~~	~~f(I2,I1),~~	f(I2,I2)

Local Routing Network – 4:1 Muxes

f(F1,F1),	f(F1,F2),	f(F1,I1),	f(F1,I2)
f(F2,F1),	f(F2,F2),	f(F2,I1),	f(F2,I2)
f(I1,F1),	f(I1,F2),	f(I1,I1),	f(I1,I2)
f(I2,F1),	f(I2,F2),	f(I2,I1),	f(I2,I2)



Duplicated-Constant Input Equivalence

- Given a k-Input LUT configured with the function:
  - $f(a_1, a_2, \ldots, a_k)$
- Connect the LUT to k independent Boolean inputs:
  - $i_1, i_2, \ldots, i_k$
- If $i_x = i_y$, then here exists another function, f', such that:
  - $f'(i_1, i_2, \ldots, i_{x-1}, \boldsymbol{i_x}, i_{x+1}, i_{x+2}, \ldots, i_{y-1}, \mathbf{0}, i_{y+1}, i_{y+2}, \ldots, i_k)$
  - =
  - $f(i_1, i_2, \ldots, i_{x-1}, \boldsymbol{i_x}, i_{x+1}, i_{x+2}, \ldots, i_{y-1}, \boldsymbol{i_y}, i_{y+1}, i_{y+2}, \ldots, i_k)$

Duplicated-Constant Input Equivalence Continued

L1	L2	L3	L4	**O**
-	0	0	0	**f0**
-	0	0	1	**f1**
-	0	1	0	**f2**
-	0	1	1	**f3**
-	1	0	0	**f4**
-	1	0	1	**f5**
-	1	1	0	**f6**
-	1	1	1	**f7**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**

A Three Input Boolean Function

L1	**L2**	L3	L4	**O**
0	**0**	0	0	**f0**
0	**0**	0	1	**f1**
0	**0**	1	0	**f2**
0	**0**	1	1	**f3**
0	1	0	0	**x**
0	1	0	1	**x**
0	1	1	0	**x**
0	1	1	1	**x**
1	0	0	0	**x**
1	0	0	1	**x**
1	0	1	0	**x**
1	0	1	1	**x**
1	**1**	0	0	**f4**
1	**1**	0	1	**f5**
1	**1**	1	0	**f6**
1	**1**	1	1	**f7**

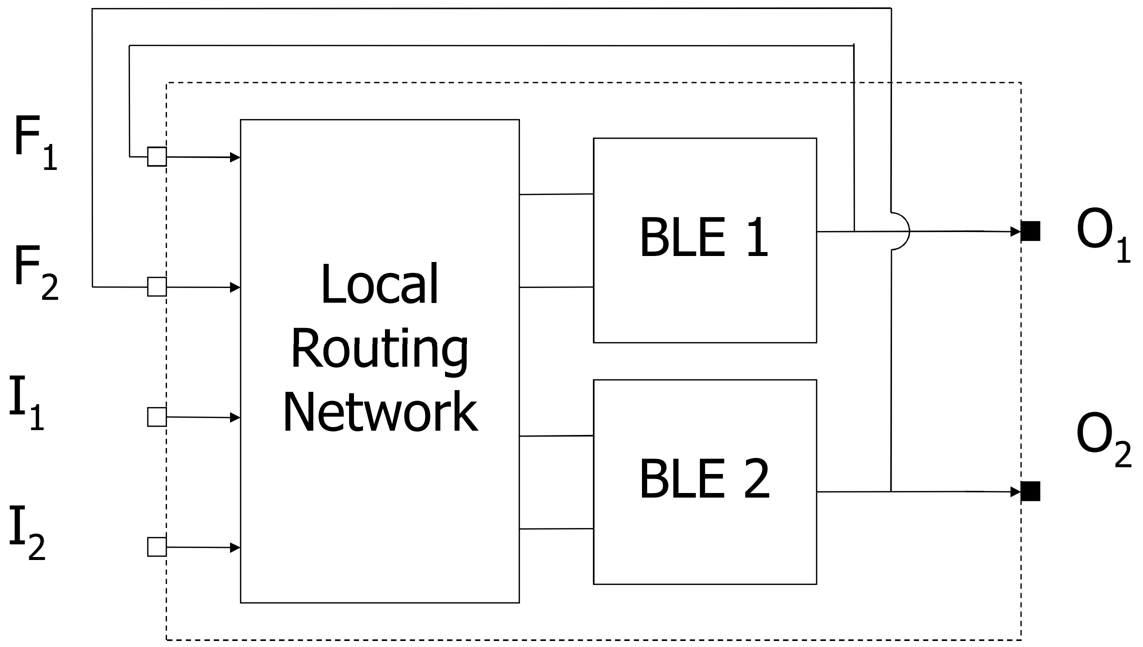
Duplicated Input Implementation

L1	L2	L3	L4	**O**
0	0	0	0	**f0**
0	0	0	1	**f1**
0	0	1	0	**f2**
0	0	1	1	**f3**
0	1	0	0	**f4**
0	1	0	1	**f5**
0	1	1	0	**f6**
0	1	1	1	**f7**
1	0	0	0	**x**
1	0	0	1	**x**
1	0	1	0	**x**
1	0	1	1	**x**
1	1	0	0	**x**
1	1	0	1	**x**
1	1	1	0	**x**
1	1	1	1	**x**

Constant Input ('0') Implementation

Duplicated-Constant Input Equivalence Continued

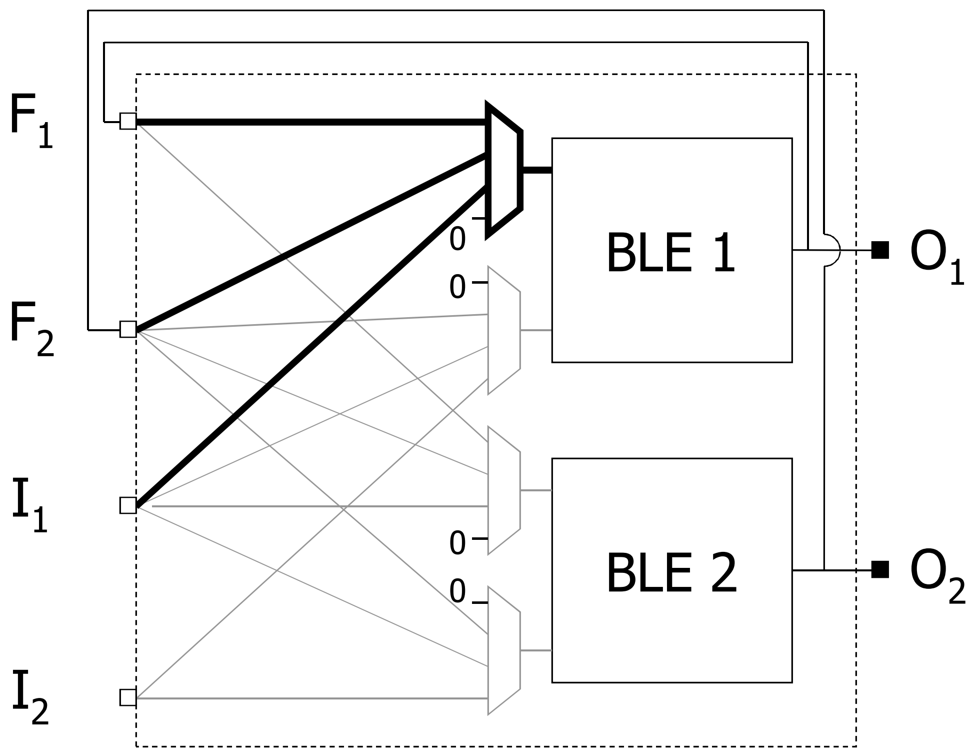
- k = 2, N = 2, I = 2: 16 functions => 10 functions



f(F1,0),	f(F1,F2),	f(F1,I1),	f(F1,I2)
f(F2,F1),	f(F2,0),	f(F2,I1),	f(F2,I2)
f(I1,F1),	f(I1,F2),	f(I1,0),	f(I1,I2)
f(I2,F1),	f(I2,F2),	f(I2,I1),	f(I2,0)

Local Routing Network – 4:1 Muxes

f(F1,0), f(F1,F2), f(F1,I1), f(F1,I2)
f(F2,F1), f(F2,0), f(F2,I1), f(F2,I2)
f(I1,F1), f(I1,F2), f(I1,0), f(I1,I2)
f(I2,F1), f(I2,F2), f(I2,I1), f(0,I2)



Constant-New Input Equivalence (Shannon Decomposition)

- Given a k-Input LUT configured with the function:
  - $f(a_1, a_2, \ldots, a_k)$
- Connect the LUT to k independent Boolean inputs:
  - $i_1, i_2, \ldots, i_k$
- If $i_x = 0$, then here exists another function, f', such that:
  - $f'(i_1, i_2, \ldots, i_{x-1}, \mathbf{i_z}, i_{x+1}, i_{x+2}, \ldots, i_k)$
  - =
  - $f(i_1, i_2, \ldots, i_{x-1}, \mathbf{0}, i_{x+1}, i_{x+2}, \ldots, i_k)$
- where i_z is a new arbitrary input

Constant-New Input Equivalence Continued

L1	L2	L3	L4	**O**
-	0	0	0	**f0**
-	0	0	1	**f1**
-	0	1	0	**f2**
-	0	1	1	**f3**
-	1	0	0	**f4**
-	1	0	1	**f5**
-	1	1	0	**f6**
-	1	1	1	**f7**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**
-	-	-	-	**--**

A Three Input Boolean Function

L1	**L2**	L3	L4	**O**
0	**0**	0	0	**f0**
0	**0**	0	1	**f1**
0	**0**	1	0	**f2**
0	**0**	1	1	**f3**
0	1	0	0	**x**
0	1	0	1	**x**
0	1	1	0	**x**
0	1	1	1	**x**
1	0	0	0	**x**
1	0	0	1	**x**
1	0	1	0	**x**
1	0	1	1	**x**
1	**1**	0	0	**f4**
1	**1**	0	1	**f5**
1	**1**	1	0	**f6**
1	**1**	1	1	**f7**

Duplicated Input Implementation

L1	L2	L3	L4	**O**
0	0	0	0	**f0**
0	0	0	1	**f1**
0	0	1	0	**f2**
0	0	1	1	**f3**
0	1	0	0	**f4**
0	1	0	1	**f5**
0	1	1	0	**f6**
0	1	1	1	**f7**
1	0	0	0	**x**
1	0	0	1	**x**
1	0	1	0	**x**
1	0	1	1	**x**
1	1	0	0	**x**
1	1	0	1	**x**
1	1	1	0	**x**
1	1	1	1	**x**

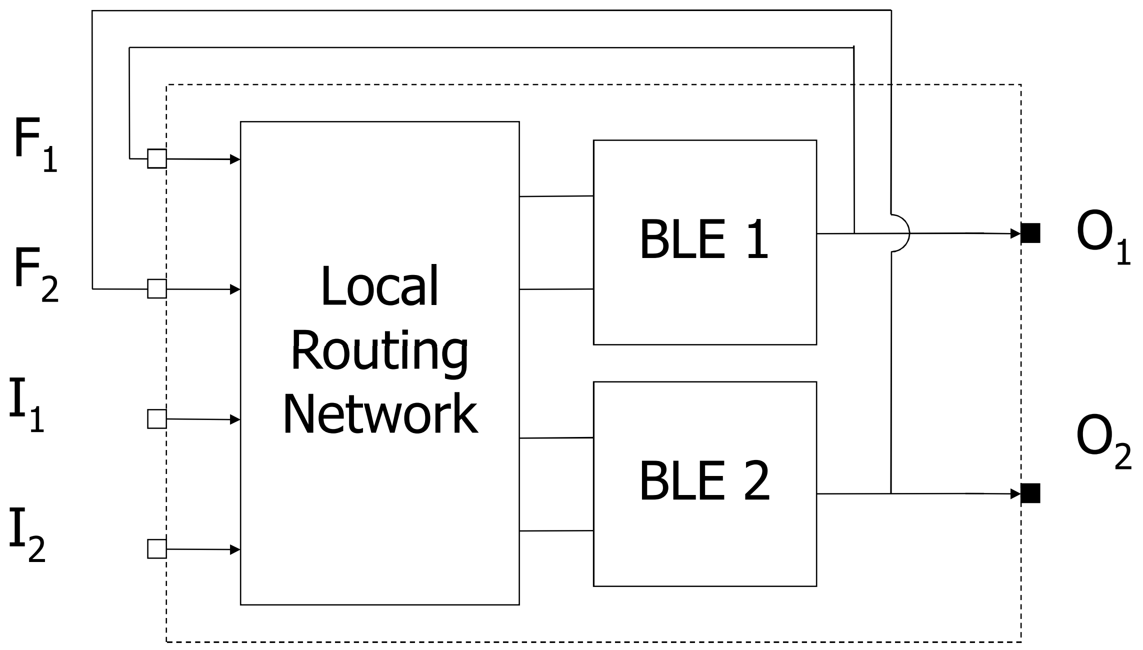
Constant Input ('0') Implementation

L1	L2	L3	L4	**O**
0	0	0	0	**f0**
0	0	0	1	**f1**
0	0	1	0	**f2**
0	0	1	1	**f3**
0	1	0	0	**f4**
0	1	0	1	**f5**
0	1	1	0	**f6**
0	1	1	1	**f7**
1	0	0	0	**f0**
1	0	0	1	**f1**
1	0	1	0	**f2**
1	0	1	1	**f3**
1	1	0	0	**f4**
1	1	0	1	**f5**
1	1	1	0	**f6**
1	1	1	1	**f7**

New Input Implementation

Constant-New Input Equivalence Continued

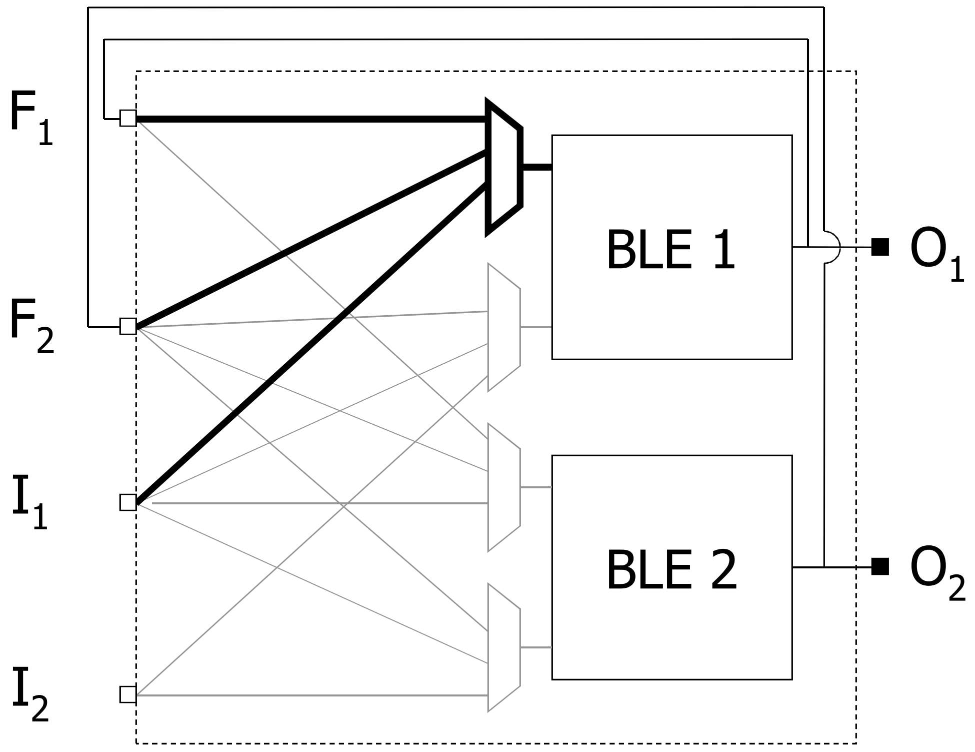
- k = 2, N = 2, I = 2: 16 functions => 10 => 6 functions



~~f(F1,0),~~	f(F1,F2),	f(F1,I1),	f(F1,I2)
f(F2,F1),	~~f(F2,0),~~	f(F2,I1),	f(F2,I2)
f(I1,F1),	f(I1,F2),	~~f(I1,0),~~	f(I1,I2)
f(I2,F1),	f(I2,F2),	f(I2,I1),	~~f(I2,0)~~

Local Routing Network – 3:1 Muxes

f(F1,0),	f(F1,F2),	f(F1,I1),	f(F1,I2)
f(F2,F1),	f(F2,0),	f(F2,I1),	f(F2,I2)
f(I1,F1),	f(I1,F2),	f(I1,0),	f(I1,I2)
f(I2,F1),	f(I2,F2),	f(I2,I1),	f(I2,0)



Summary – Function Reduction for Local Routing Network

- Without LUT Reconfiguration – $(I+N)^k$ Functions
- Commutative Property – $\sum_{j=1}^{k}\binom{n}{j}$ Functions ($n = I + N$)
- Duplicated-Constant Input Equivalency – $\sum_{j=1}^{k}\binom{n}{j}$ Functions
- Constant-New Input Equivalency – $\binom{n}{k}$ Functions

What is the effect of function reduction on the design of the local routing network?

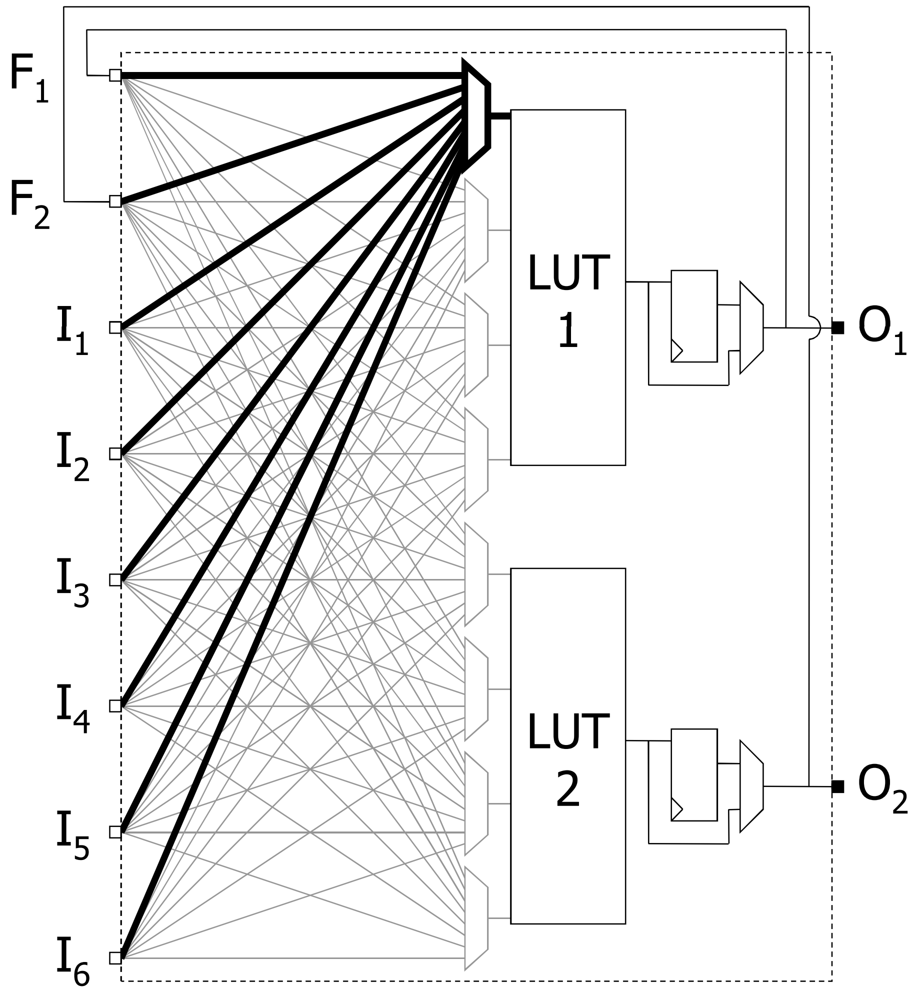
Summary – Local Routing Network Design

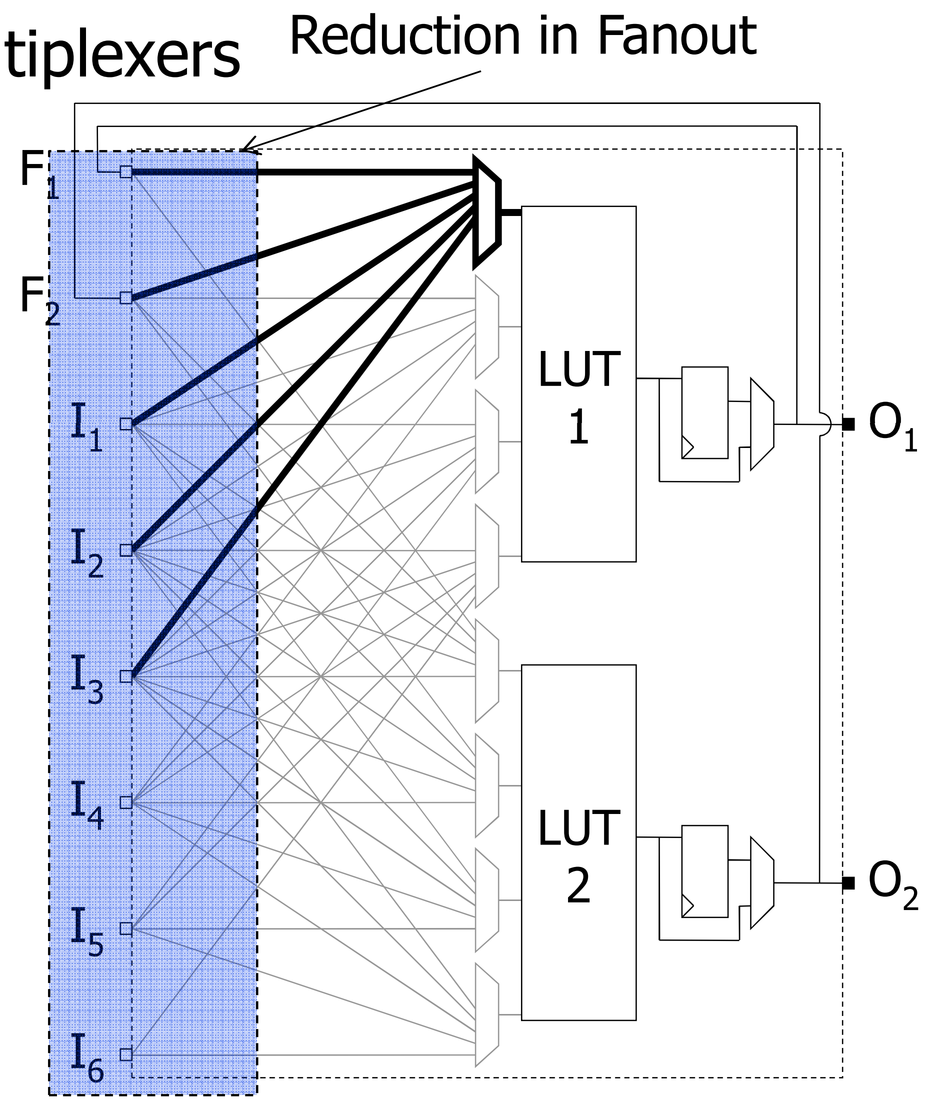
f(F1,F1),	$f(\underline{F1},\underline{F2})$,	$f(\underline{F1},\underline{I1})$,	$f(\underline{F1},\underline{I2})$
f(F2,F1),	f(F2,F2),	$f(\underline{F2},\underline{I1})$,	$f(\underline{F2},\underline{I2})$
f(I1,F1),	f(I1,F2),	f(I1,I1),	$f(\underline{I1},\underline{I2})$
f(I2,F1),	f(I2,F2),	f(I2,I1),	f(I2,I2)

- LUT Input 1 = {F1, F2, I1}
- LUT Input 2 = {F2, I1, I2}
- 3:1 Multiplexers instead of 4:1 Multiplexers
- In General: From (I+N):1 Multiplexers to (I+N-k+1):1 Multiplexers

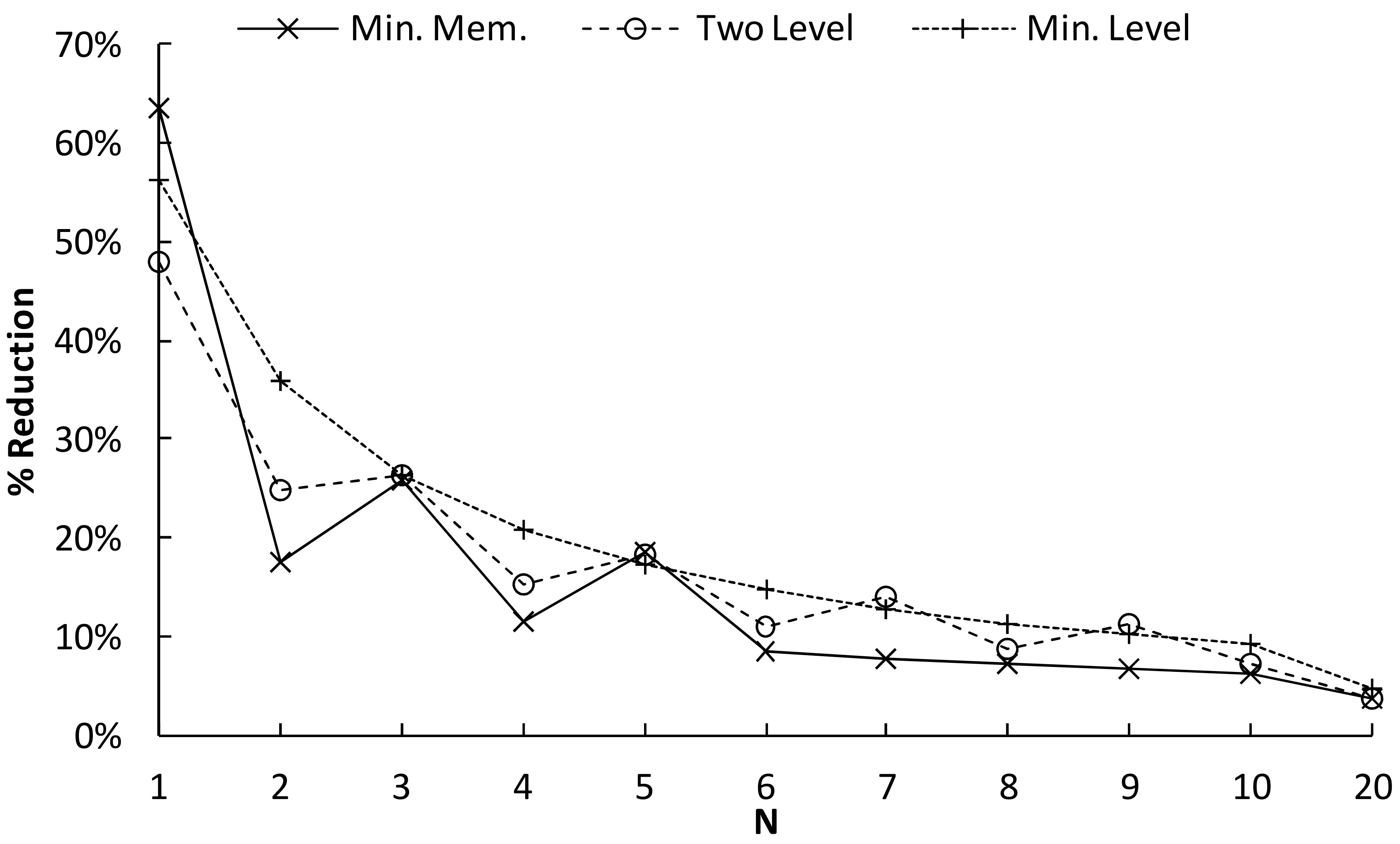
A More Complex Example

- k = 4, N = 2, I = 6: 4096 Functions => 70 Functions
- 8:1 Multiplexers => 5:1 Multiplexers

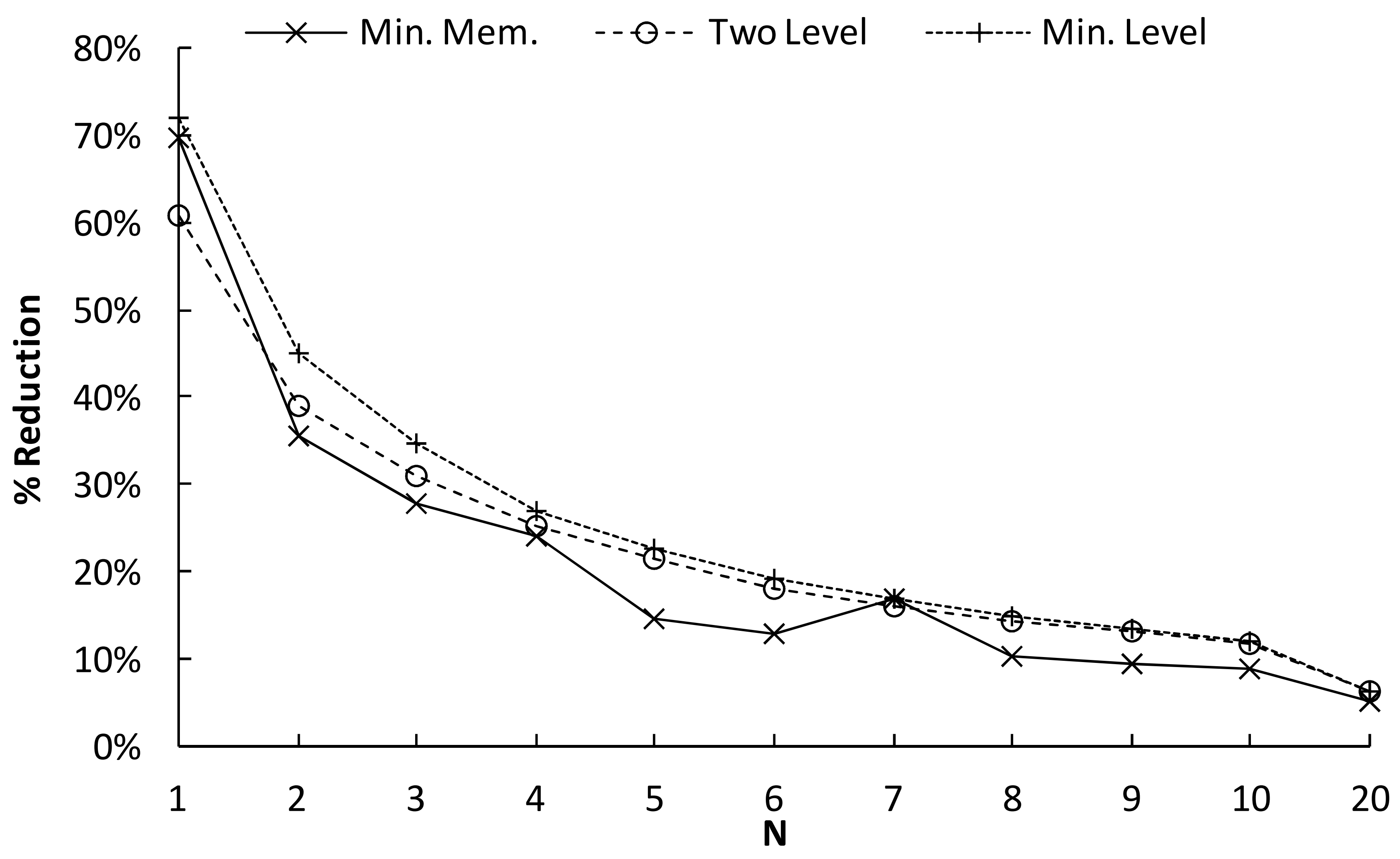




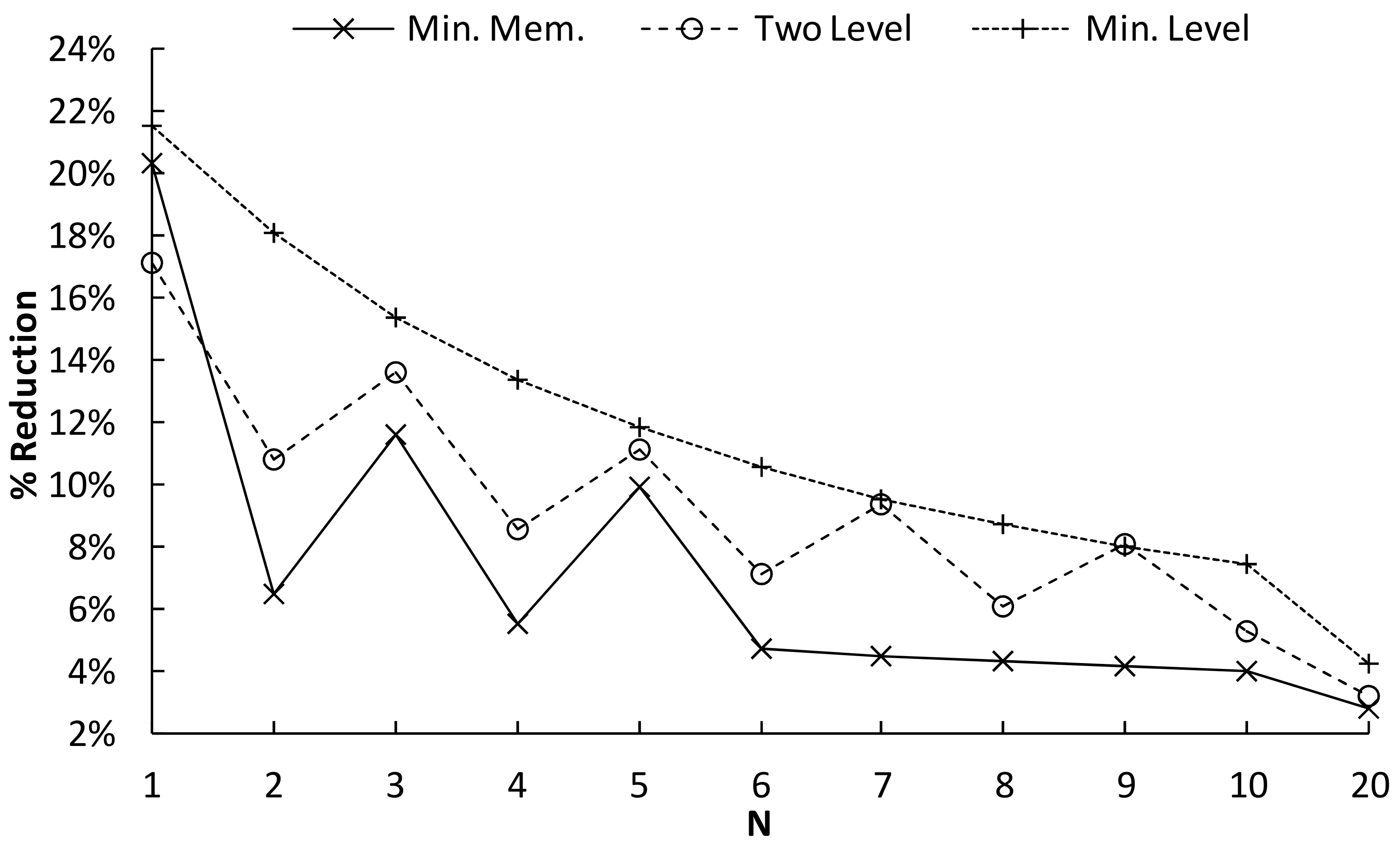
Local Routing Network Area Reduction k = 4



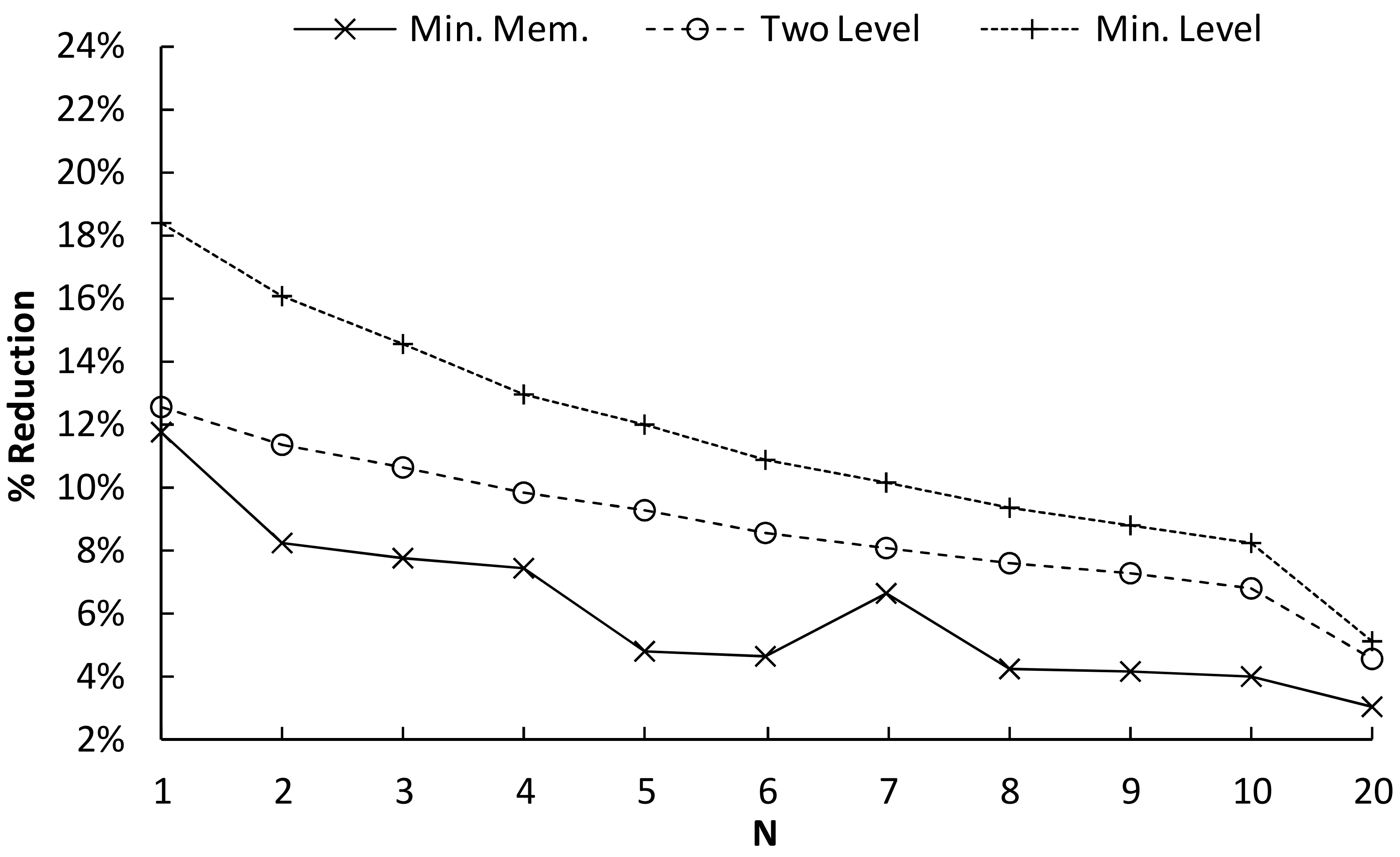
Local Routing Network Area Reduction k = 7



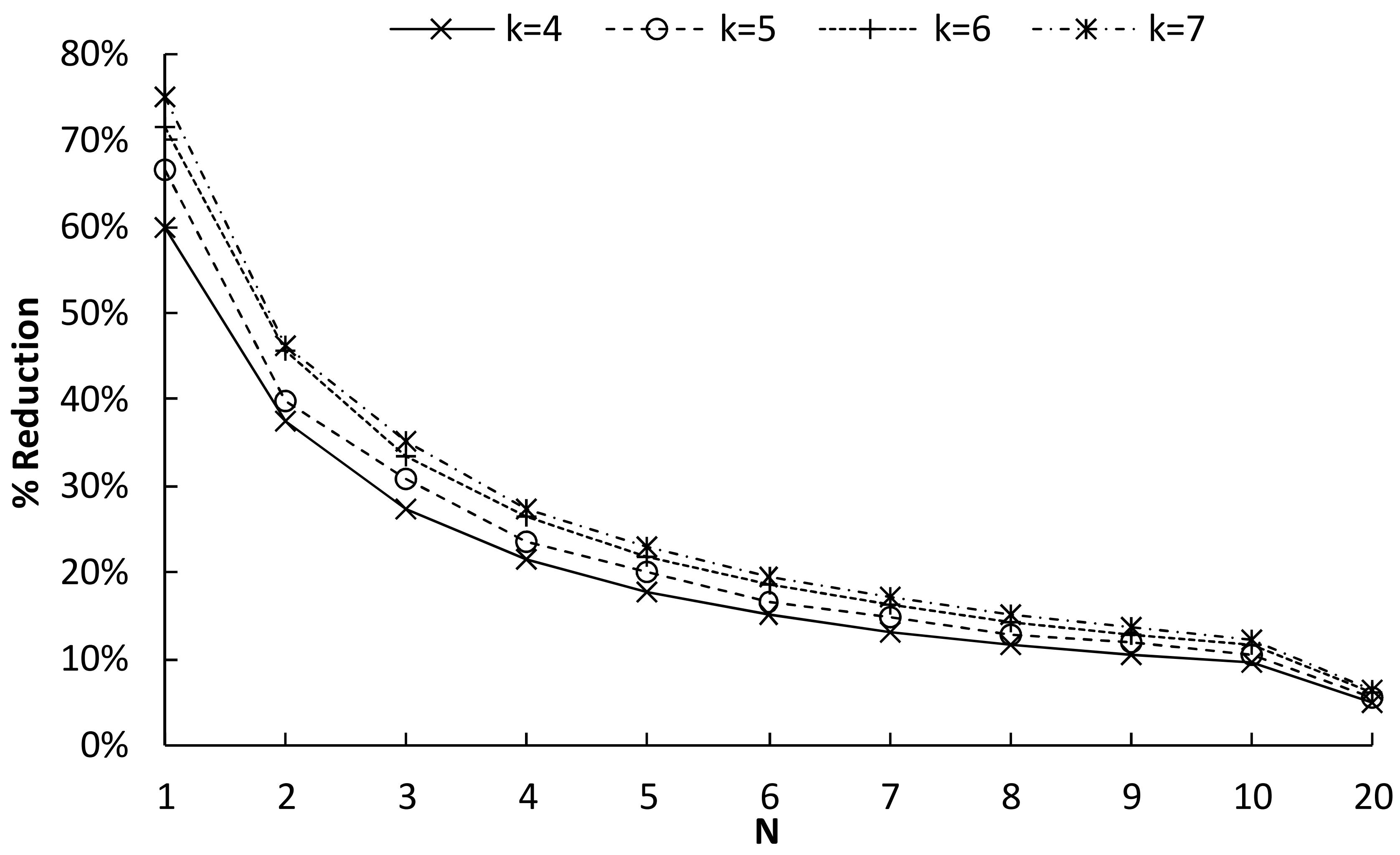
Logic Cluster Area Reduction k = 4



Logic Cluster Area Reduction k = 7



Fanout Reduction



Fanout Adjusted Logic Cluster Area Reduction

- N = 6

k	I	Min. Mem			Min. Level			Two Level		
		Trad.	New	% Reduc.	Trad.	New	% Reduc.	Trad.	New	% Reduc.
4	14	3376.4	3158.8	6.4%	5104.4	4526.8	11.3%	4048.4	3710.8	8.3%
5	17	5089.3	4729.3	7.1%	7849.3	6889.3	12.2%	6229.3	5569.3	10.6%
6	21	7609.1	7085.6	6.9%	11461.1	10037.6	12.4%	9337.1	8273.6	11.4%
7	24	11877.0	11139.9	6.2%	17211.0	15213.9	11.6%	14313.0	12945.9	9.6%

Conclusions

- Examined the relationship between logic equivalency and LUT reconfiguration.
- LUT reconfiguration can reduce mux size from (I+N):1 to (I+N-k+1):1.
- (I+N-k+1):1 is the minimum size required to retain logic equivalency (proved in paper TVLSI Jan 2010).

- Local Routing Network Area Reduction – 3.7%-72%
- Logic Cluster Area Reduction – 2.9%-25%
- Fanout Reduction – 5%-75%
- Fanout Adjust Logic Cluster Area Reduction (N=6) – 6.2% (k=7, Min. Mem) – 12.4% (k=6, Min. Level)

Questions?

Future Work 1: Based on [Lemieux01]

- Proposed a sparse local routing network
- N=6 (k=4-7) => over 50% reduction in local routing network area compare to the baseline (full connectivity)
- This Work => baseline can be improved by 8.5%-13%

- Didn't study smaller values of N in detail (e.g. N=4)
- This Work => N=4: baseline can be improved by 24% (k=6 and k=7)
- Is sparse local routing network still more area efficient than the improved baseline architecture at N=4?
- Overall, is the sparse architecture still more area efficient than the improved baseline architecture when all values of N are considered?

Future Work 2: Based on [Feng08]

- Actel Logic Cluster => 8 4-input LUTs, 32 logic cluster inputs, 8:1 mux per LUT input.
- Sacrifice logic equivalency and local feedbacks for larger logic cluster size
- Justification: 8:1 mux can only be used to construct VPR type logic clusters with 2 4-input LUTs only; 2 4-input LUTs are not efficient

- This work: 8:1 mux can be used to construct logic clusters with 3 4-input LUTs with full feedbacks or 4 4-input LUTs without feedbacks => VPR style clusters become competitive again
- Need further experimental studies

Impact on Previous Work [Lemieux01]

- N = 6 (k=4-7) => 50% reduction in local routing network area compare to full connectivity as baseline
- Our Work => baseline can be improved by 8.5%-13%

- Didn't study smaller values of N in detail (e.g. N=4)
- Our Work => N=4: baseline can be improved by 24% (k=6 and k=7)
- How much sparse local routing network can improve for N = 4 (probably much less than 50% as observed for N = 6)?
- 4 + 2 tightly connected LUTs as an alternative?

- Figure 4 needs update – N=2, k=7 => 35.8% reduction in local routing network area for baseline

Future Work: Impact on Previous Work [Lemieux01]

- Proposed a sparse local routing network
- N=6 (k=4-7) => over 50% reduction in local routing network area compare to the baseline (full connectivity)
- Our Work => baseline can be improved by 8.5%-13%

- Didn't study smaller values of N in detail (e.g. N=4)
- Our Work => N=4: baseline can be improved by 24% (k=6 and k=7)
- Is sparse local routing network still more area efficient than the improved baseline architecture at N=4?
- Overall, is the sparse architecture more area efficient than the baseline architecture when all values of N are considered?
- N=6 => 4 + 2 tightly connected LUTs as an alternative?