ice40 architecture

www.clifford.at/icestorm/

Pat Hanrahan

CS448H: Agile Hardware Design
Winter 2017

Deep Dive on ice40

Learn more about FPGA architecture

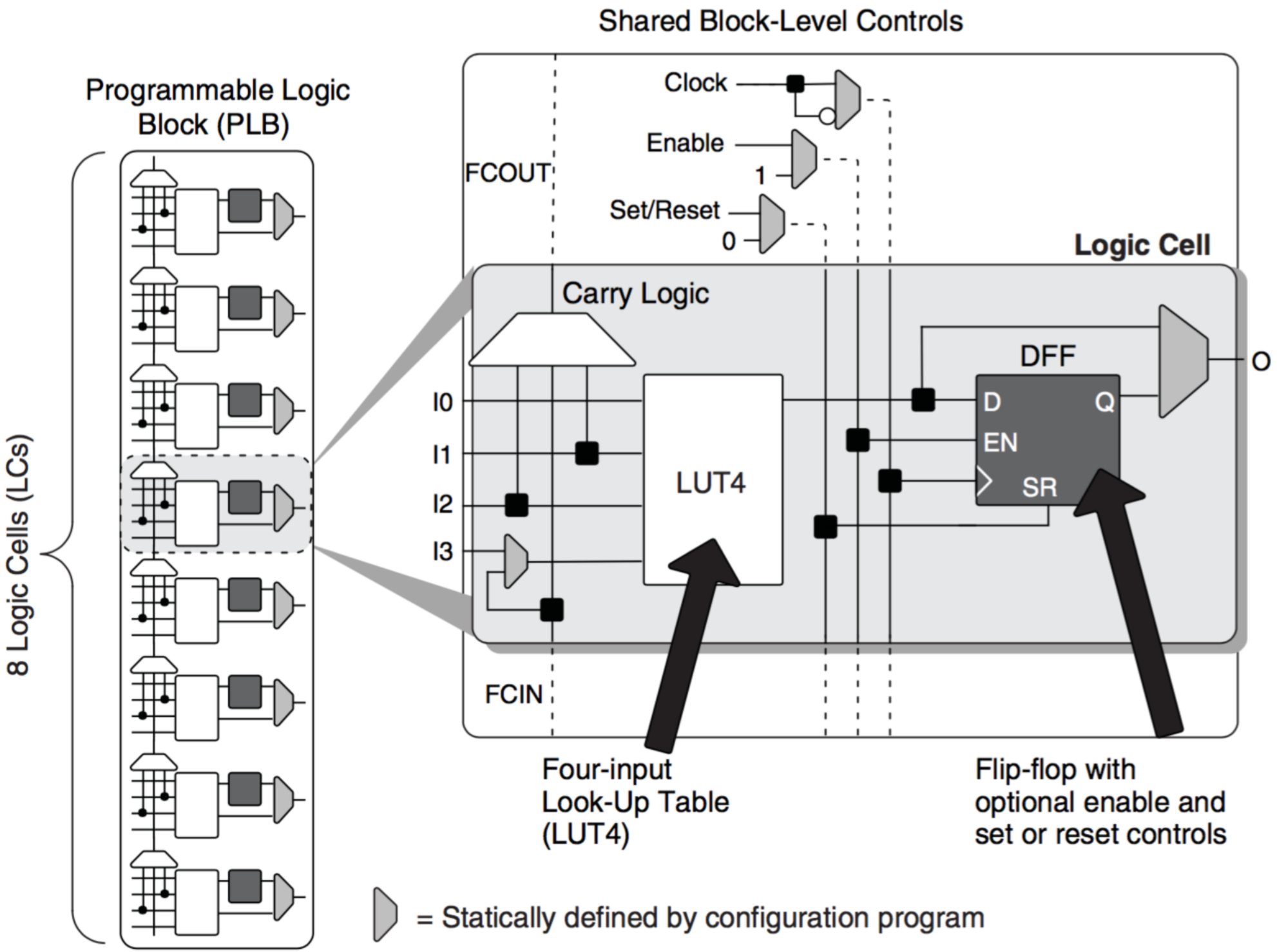
- **Interconnection network**

How to improve FPGAs?

	IO (1 17)	IO (2 17)	IO (3 17)	IO (4 17)	IO (5 17)	IO (6 17)	IO (7 17)	IO (8 17)	IO (9 17)	IO (10 17)	IO (11 17)	IO (12 17)	
IO (0 16)	LOGIC (1 16)	LOGIC (2 16)	RAMT (3 16)	LOGIC (4 16)	LOGIC (5 16)	LOGIC (6 16)	LOGIC (7 16)	LOGIC (8 16)	LOGIC (9 16)	RAMT (10 16)	LOGIC (11 16)	LOGIC (12 16)	IO (13 16)
IO (0 15)	LOGIC (1 15)	LOGIC (2 15)	RAMB (3 15)	LOGIC (4 15)	LOGIC (5 15)	LOGIC (6 15)	LOGIC (7 15)	LOGIC (8 15)	LOGIC (9 15)	RAMB (10 15)	LOGIC (11 15)	LOGIC (12 15)	IO (13 15)
IO (0 14)	LOGIC (1 14)	LOGIC (2 14)	RAMT (3 14)	LOGIC (4 14)	LOGIC (5 14)	LOGIC (6 14)	LOGIC (7 14)	LOGIC (8 14)	LOGIC (9 14)	RAMT (10 14)	LOGIC (11 14)	LOGIC (12 14)	IO (13 14)
IO (0 13)	LOGIC (1 13)	LOGIC (2 13)	RAMB (3 13)	LOGIC (4 13)	LOGIC (5 13)	LOGIC (6 13)	LOGIC (7 13)	LOGIC (8 13)	LOGIC (9 13)	RAMB (10 13)	LOGIC (11 13)	LOGIC (12 13)	IO (13 13)
IO (0 12)	LOGIC (1 12)	LOGIC (2 12)	RAMT (3 12)	LOGIC (4 12)	LOGIC (5 12)	LOGIC (6 12)	LOGIC (7 12)	LOGIC (8 12)	LOGIC (9 12)	RAMT (10 12)	LOGIC (11 12)	LOGIC (12 12)	IO (13 12)
IO (0 11)	LOGIC (1 11)	LOGIC (2 11)	RAMB (3 11)	LOGIC (4 11)	LOGIC (5 11)	LOGIC (6 11)	LOGIC (7 11)	LOGIC (8 11)	LOGIC (9 11)	RAMB (10 11)	LOGIC (11 11)	LOGIC (12 11)	IO (13 11)
IO (0 10)	LOGIC (1 10)	LOGIC (2 10)	RAMT (3 10)	LOGIC (4 10)	LOGIC (5 10)	LOGIC (6 10)	LOGIC (7 10)	LOGIC (8 10)	LOGIC (9 10)	RAMT (10 10)	LOGIC (11 10)	LOGIC (12 10)	IO (13 10)
IO (0 9)	LOGIC (1 9)	LOGIC (2 9)	RAMB (3 9)	LOGIC (4 9)	LOGIC (5 9)	LOGIC (6 9)	LOGIC (7 9)	LOGIC (8 9)	LOGIC (9 9)	RAMB (10 9)	LOGIC (11 9)	LOGIC (12 9)	IO (13 9)
IO (0 8)	LOGIC (1 8)	LOGIC (2 8)	RAMT (3 8)	LOGIC (4 8)	LOGIC (5 8)	LOGIC (6 8)	LOGIC (7 8)	LOGIC (8 8)	LOGIC (9 8)	RAMT (10 8)	LOGIC (11 8)	LOGIC (12 8)	IO (13 8)
IO (0 7)	LOGIC (1 7)	LOGIC (2 7)	RAMB (3 7)	LOGIC (4 7)	LOGIC (5 7)	LOGIC (6 7)	LOGIC (7 7)	LOGIC (8 7)	LOGIC (9 7)	RAMB (10 7)	LOGIC (11 7)	LOGIC (12 7)	IO (13 7)
IO (0 6)	LOGIC (1 6)	LOGIC (2 6)	RAMT (3 6)	LOGIC (4 6)	LOGIC (5 6)	LOGIC (6 6)	LOGIC (7 6)	LOGIC (8 6)	LOGIC (9 6)	RAMT (10 6)	LOGIC (11 6)	LOGIC (12 6)	IO (13 6)
IO (0 5)	LOGIC (1 5)	LOGIC (2 5)	RAMB (3 5)	LOGIC (4 5)	LOGIC (5 5)	LOGIC (6 5)	LOGIC (7 5)	LOGIC (8 5)	LOGIC (9 5)	RAMB (10 5)	LOGIC (11 5)	LOGIC (12 5)	IO (13 5)
IO (0 4)	LOGIC (1 4)	LOGIC (2 4)	RAMT (3 4)	LOGIC (4 4)	LOGIC (5 4)	LOGIC (6 4)	LOGIC (7 4)	LOGIC (8 4)	LOGIC (9 4)	RAMT (10 4)	LOGIC (11 4)	LOGIC (12 4)	IO (13 4)
IO (0 3)	LOGIC (1 3)	LOGIC (2 3)	RAMB (3 3)	LOGIC (4 3)	LOGIC (5 3)	LOGIC (6 3)	LOGIC (7 3)	LOGIC (8 3)	LOGIC (9 3)	RAMB (10 3)	LOGIC (11 3)	LOGIC (12 3)	IO (13 3)
IO (0 2)	LOGIC (1 2)	LOGIC (2 2)	RAMT (3 2)	LOGIC (4 2)	LOGIC (5 2)	LOGIC (6 2)	LOGIC (7 2)	LOGIC (8 2)	LOGIC (9 2)	RAMT (10 2)	LOGIC (11 2)	LOGIC (12 2)	IO (13 2)
IO (0 1)	LOGIC (1 1)	LOGIC (2 1)	RAMB (3 1)	LOGIC (4 1)	LOGIC (5 1)	LOGIC (6 1)	LOGIC (7 1)	LOGIC (8 1)	LOGIC (9 1)	RAMB (10 1)	LOGIC (11 1)	LOGIC (12 1)	IO (13 1)
	IO (1 0)	IO (2 0)	IO (3 0)	IO (4 0)	IO (5 0)	IO (6 0)	IO (7 0)	IO (8 0)	IO (9 0)	IO (10 0)	IO (11 0)	IO (12 0)	

```
% cd demo; make; cat and.txt
.logic_tile 1 11
000000000000001000000000000000000000000000000000
000000000000001010000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000010000000001000000000000
000000000000000000000000111000000000000000000000
000000000000000000000000000000000000000000000000
000000000000111100000000000000000000000000000000
000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000
```

```
% make explain
.logic_tile 1 11
LC_5 0001000000000000 0000
buffer local_g0_0 lutff_5/in_1
buffer local_g3_0 lutff_5/in_0
buffer neigh_op_lft_0 local_g0_0
buffer sp4_h_r_24 local_g3_0
```


Shared Block-Level Controls
Programmable Logic Block (PLB)
Clock
Enable
1
Set/Reset
0
FCOUT
Logic Cell
Carry Logic
DFF
O
I0
I1
I2
I3
LUT4
D
Q
EN
SR
FCIN
8 Logic Cells (LCs)
Four-input Look-Up Table (LUT4)
Flip-flop with optional enable and set or reset controls
= Statically defined by configuration program

```
// From  yosys/techlibs/ice40/cells_sim.v

module ICESTORM_LC (
    input I0, I1, I2, I3, CIN, CLK, CEN, SR,
    output LO, O, COUT
);
    parameter [15:0] LUT_INIT = 0;
    parameter [0:0] NEG_CLK      = 0;
    parameter [0:0] CARRY_ENABLE = 0;
    parameter [0:0] DFF_ENABLE   = 0;
    parameter [0:0] SET_NORESET  = 0;
    parameter [0:0] SET_ASYNC    = 0;
    parameter [0:0] ASYNC_SR     = 0;
    …
endmodule
```

```
// From  yosys/techlibs/ice40/cells_sim.v
module ICESTORM_LC (
    input I0, I1, I2, I3, CIN, CLK, CEN, SR,
    output LO, O, COUT
);
    wire COUT = CARRY_ENABLE ? (I1 && I2) || ((I1 || I2) && CIN) : 1'bx;
    wire [7:0] lut_s3 = I3 ? LUT_INIT[15:8] : LUT_INIT[7:0];
    wire [3:0] lut_s2 = I2 ?   lut_s3[ 7:4] :   lut_s3[3:0];
    wire [1:0] lut_s1 = I1 ?   lut_s2[ 3:2] :   lut_s2[1:0];
    wire       lut_o  = I0 ?   lut_s1[   1] :   lut_s1[  0];
    assign LO = lut_o;

    wire polarized_clk;
    assign polarized_clk = CLK ^ NEG_CLK;

    reg o_reg;
    always @(posedge polarized_clk)
        if (CEN)
            o_reg <= SR ? SET_NORESET : lut_o;

    reg o_reg_async;
    always @(posedge polarized_clk, posedge SR)
        if (SR)
            o_reg <= SET_NORESET;
        else if (CEN)
            o_reg <= lut_o;

    assign O = DFF_ENABLE ? ASYNC_SR ? o_reg_async : o_reg : lut_o;
endmodule
```

Interconnect

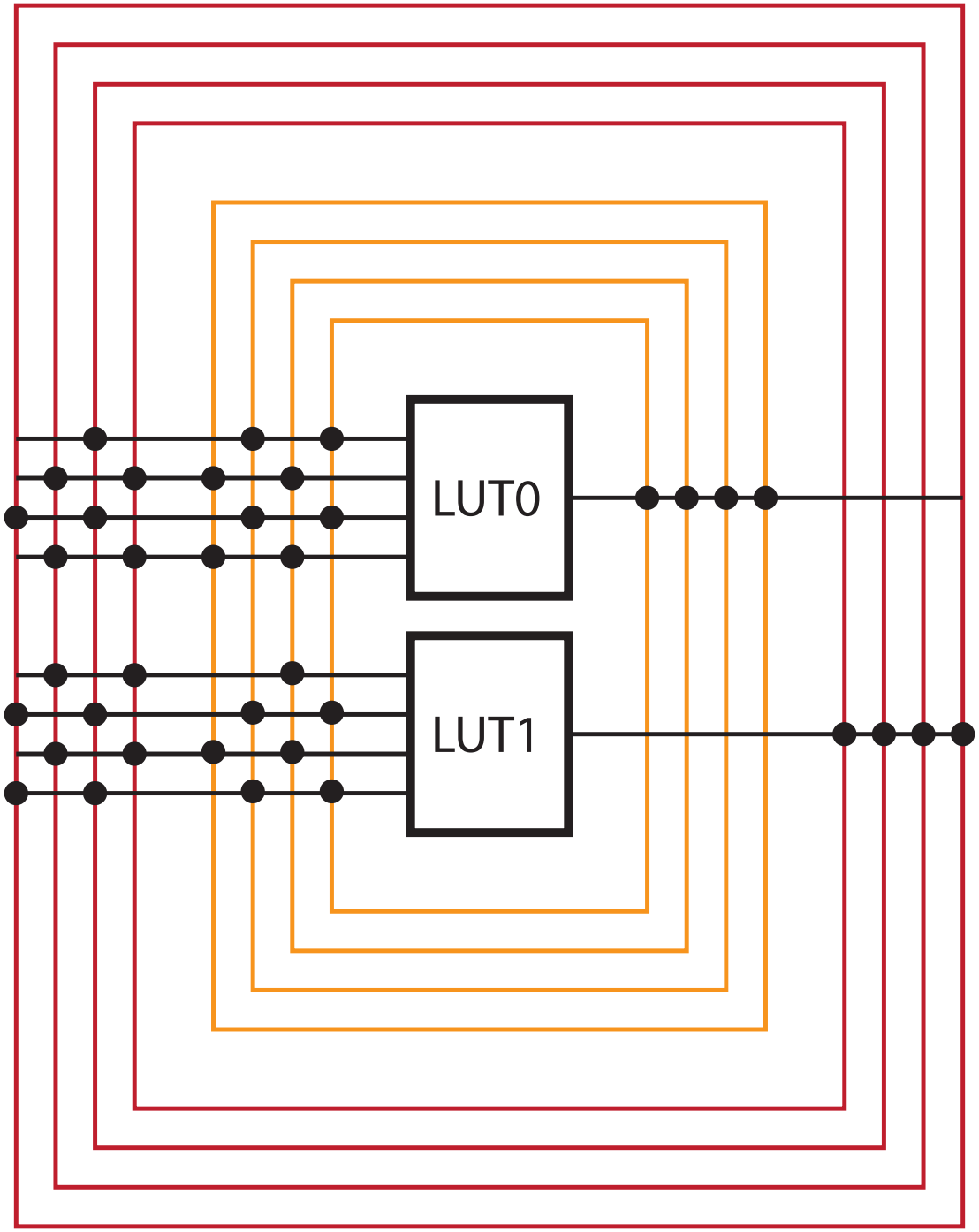
Local Tracks

There 32 local wires

They are organized into 4 groups of 8 wires

Each local track is drive by a single LUT output (LUT[i], g)

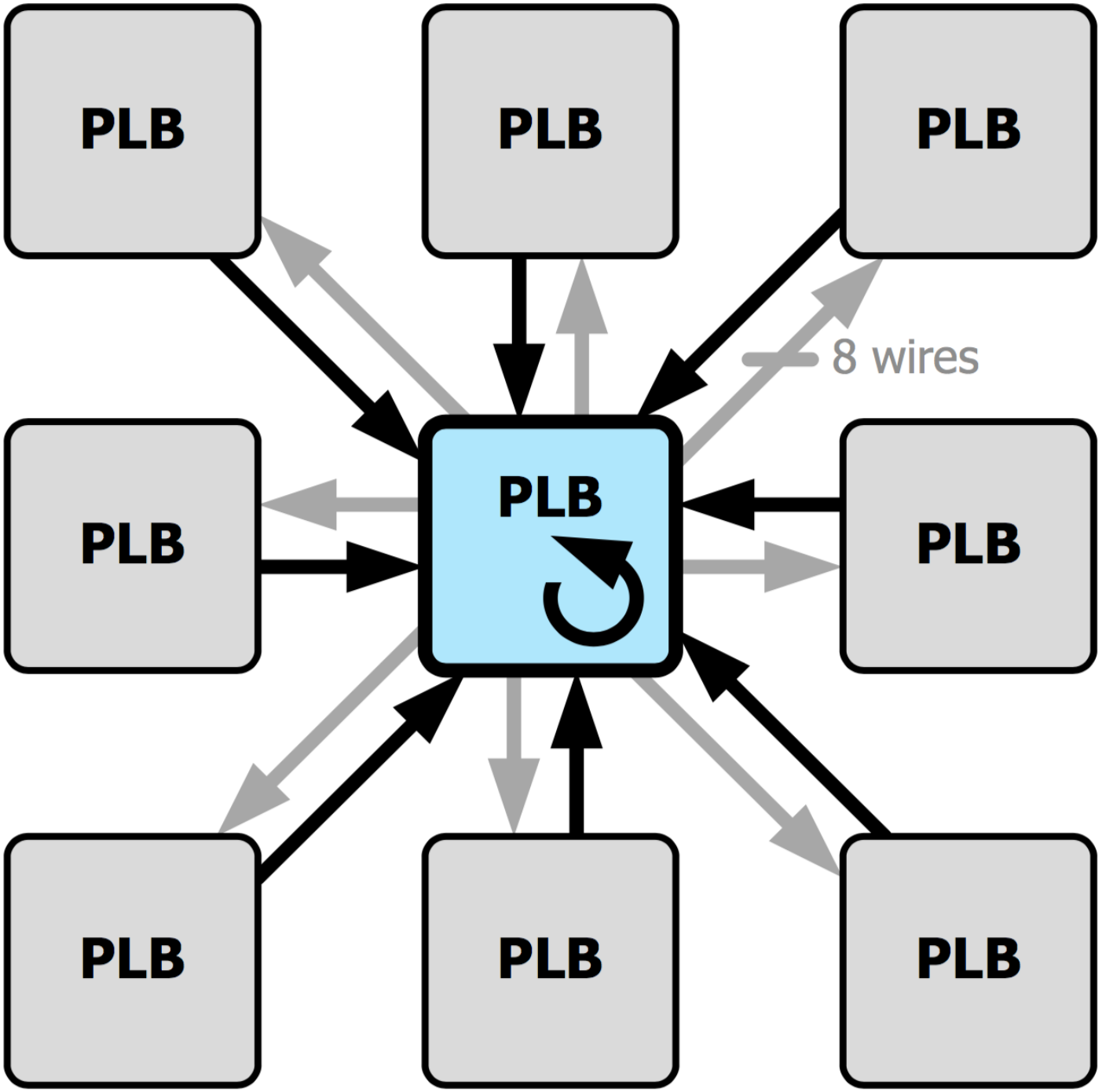
The local tracks are connected to the LUT inputs (more on this later)


LUT0
LUT1

```
l i 00001111222233334444555566667777
l g 01230123012301230123012301230123
0 0 x x  x xx x  x xx x  x xx x  x x
0 1  x xx x  x xx x  x xx x  x xx x
0 2 x x  x xx x  x xx x  x xx x  x x
0 3  x x  x  x xx x  x xx x  x xx x
1 0  x xx x  x xx x  x xx x  x xx x
1 1 x x  x xx x  x xx x  x xx x  x x
1 2  x xx x  x xx x  x xx x  x xx x
1 3   x  x xx x  x xx x  x xx x  x x
2 0 x x  x xx x  x xx x  x xx x  x x
2 1  x xx x  x xx x  x xx x  x xx x
2 2 x x  x xx x  x xx x  x xx x  x x
2 3  x x  x  x xx x  x xx x  x xx x
3 0  x xx x  x xx x  x xx x  x xx x
3 1 x x  x xx x  x xx x  x xx x  x x
3 2  x xx x  x xx x  x xx x  x xx x
3 3   x  x xx x  x xx x  x xx x  x x
4 0 x x  x xx x  x xx x  x xx x  x x
4 1  x xx x  x xx x  x xx x  x xx x
4 2 x x  x xx x  x xx x  x xx x  x x
4 3  x x  x  x xx x  x xx x  x xx x
5 0  x xx x  x xx x  x xx x  x xx x
5 1 x x  x xx x  x xx x  x xx x  x x
5 2  x xx x  x xx x  x xx x  x xx x
5 3   x  x xx x  x xx x  x xx x  x x
6 0 x x  x xx x  x xx x  x xx x  x x
6 1  x xx x  x xx x  x xx x  x xx x
6 2 x x  x xx x  x xx x  x xx x  x x
6 3  x x  x  x xx x  x xx x  x xx x
7 0  x xx x  x xx x  x xx x  x xx x
7 1 x x  x xx x  x xx x  x xx x  x x
7 2  x xx x  x xx x  x xx x  x xx x
7 3   x  x xx x  x xx x  x xx x  x x
```

Nearest Neighbors

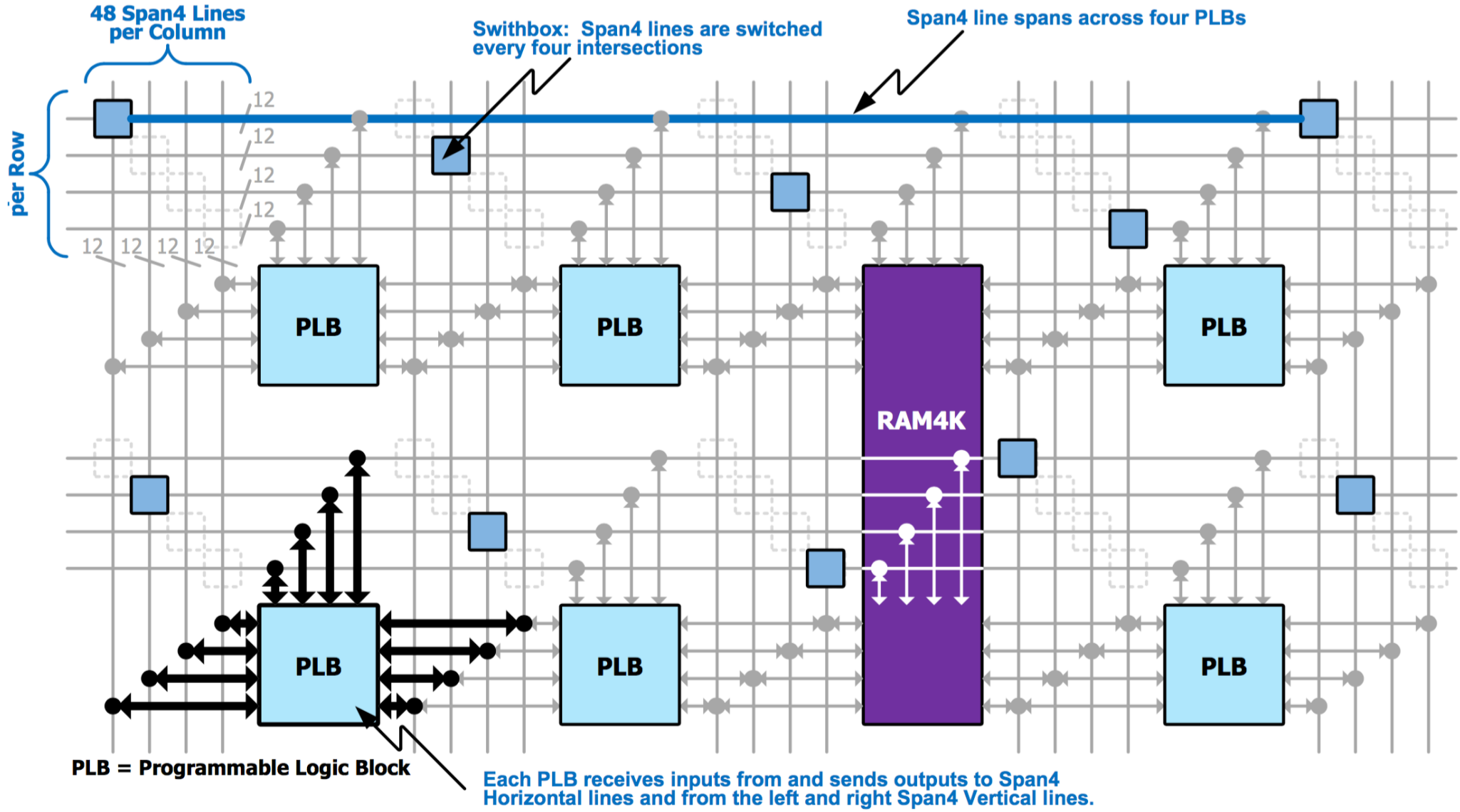
Figure 13: **PLB Connections to Eight Nearest Neighbors**

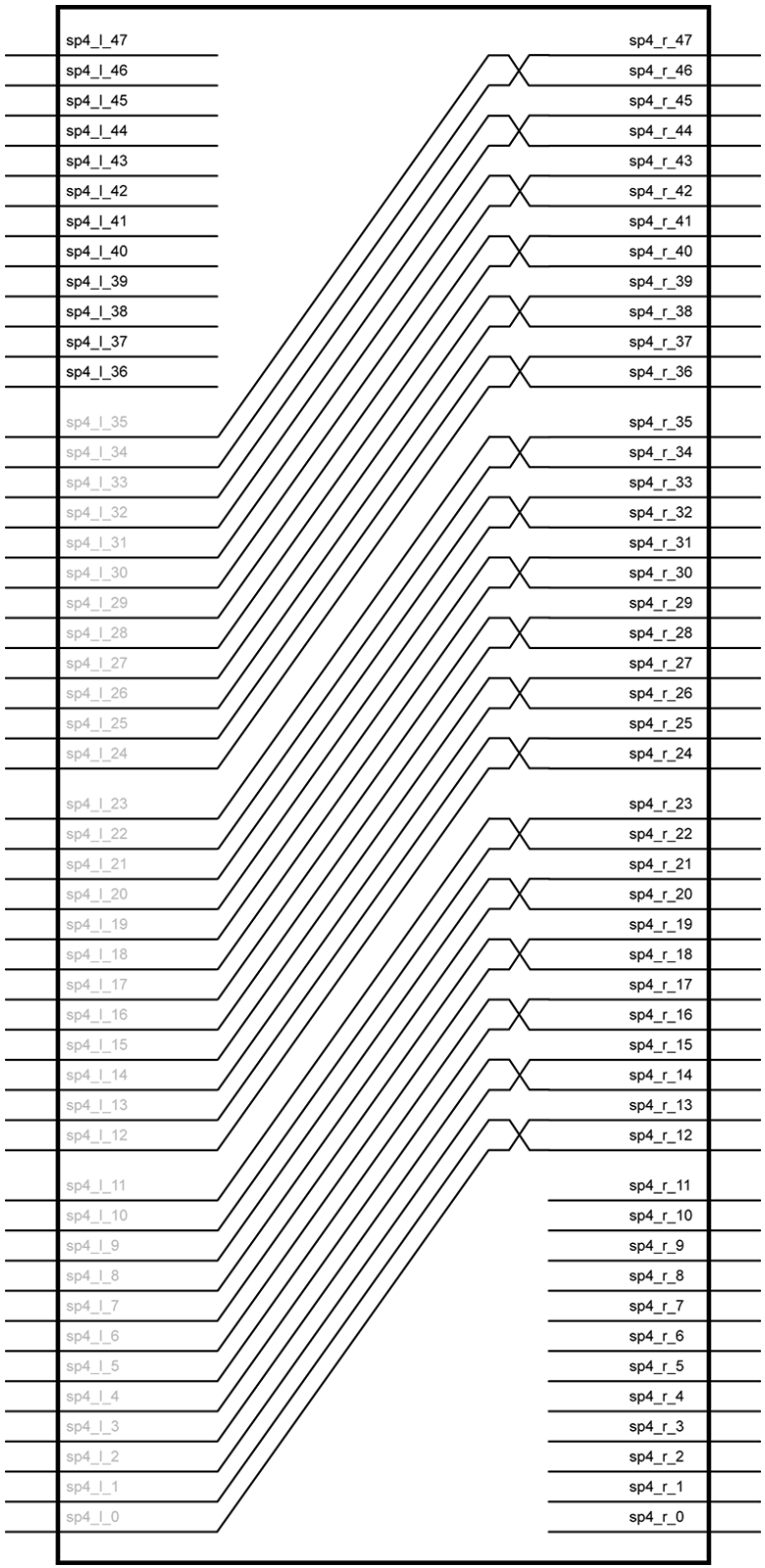


PLB = Programmable Logic Block

Spans

Figure 14: **Span4 Connection Resources**





Horizontal Span4

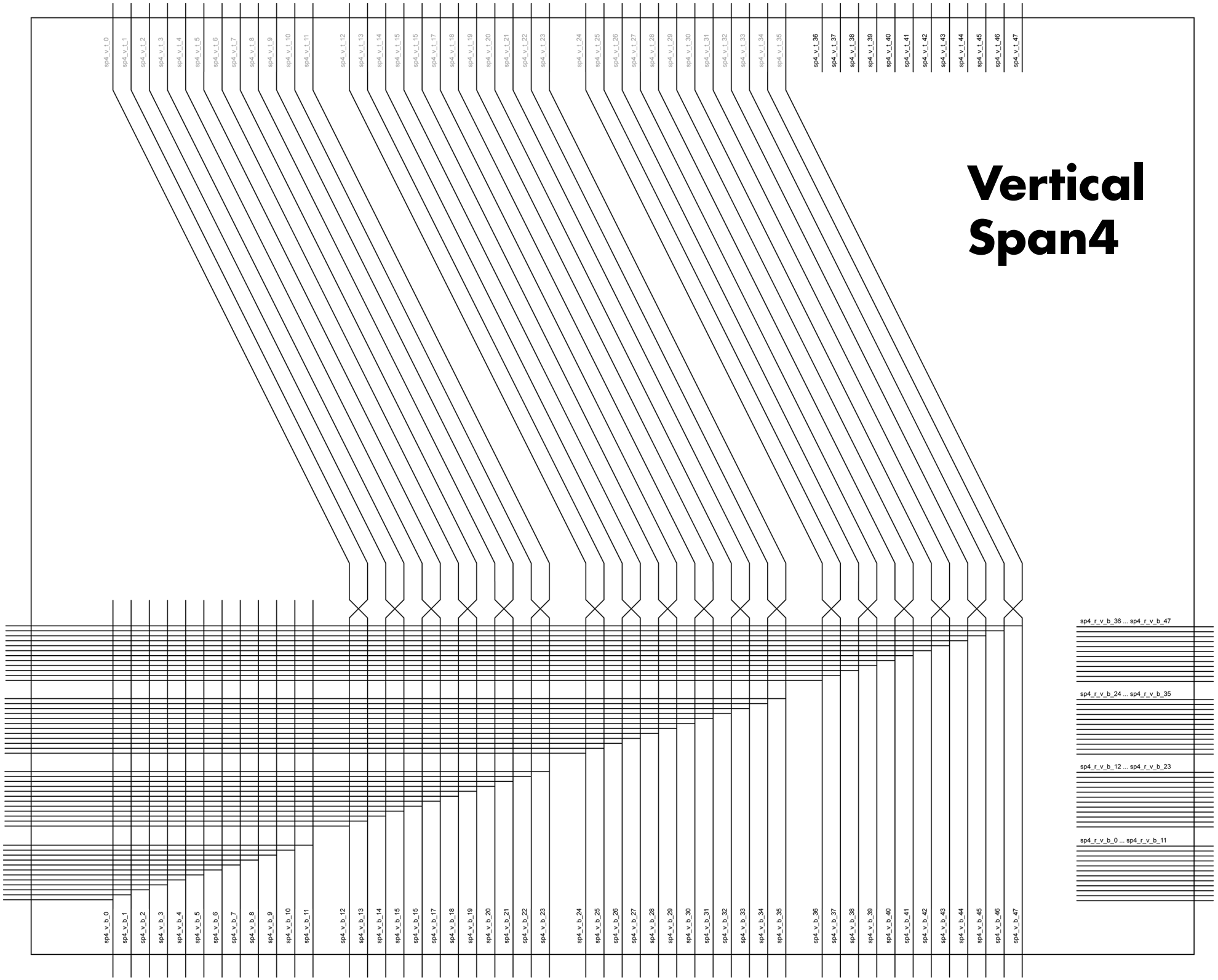
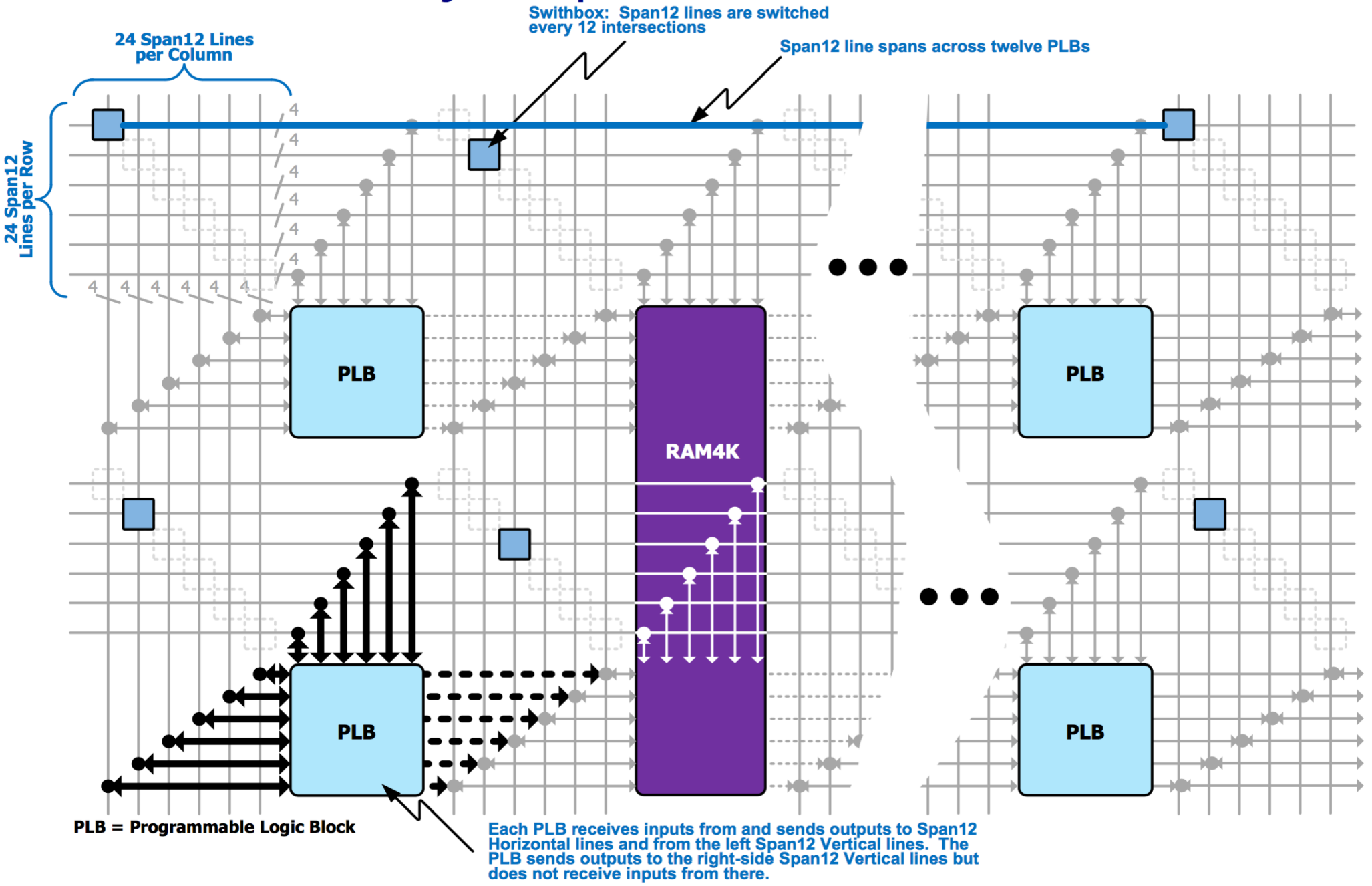

Vertical
Span4
sp4_v_t_0
sp4_v_t_1
sp4_v_t_2
sp4_v_t_3
sp4_v_t_4
sp4_v_t_5
sp4_v_t_6
sp4_v_t_7
sp4_v_t_8
sp4_v_t_9
sp4_v_t_10
sp4_v_t_11
sp4_v_t_12
sp4_v_t_13
sp4_v_t_14
sp4_v_t_15
sp4_v_t_16
sp4_v_t_17
sp4_v_t_18
sp4_v_t_19
sp4_v_t_20
sp4_v_t_21
sp4_v_t_22
sp4_v_t_23
sp4_v_t_24
sp4_v_t_25
sp4_v_t_26
sp4_v_t_27
sp4_v_t_28
sp4_v_t_29
sp4_v_t_30
sp4_v_t_31
sp4_v_t_32
sp4_v_t_33
sp4_v_t_34
sp4_v_t_35
sp4_v_t_36
sp4_v_t_37
sp4_v_t_38
sp4_v_t_39
sp4_v_t_40
sp4_v_t_41
sp4_v_t_42
sp4_v_t_43
sp4_v_t_44
sp4_v_t_45
sp4_v_t_46
sp4_v_t_47
sp4_v_b_0
sp4_v_b_1
sp4_v_b_2
sp4_v_b_3
sp4_v_b_4
sp4_v_b_5
sp4_v_b_6
sp4_v_b_7
sp4_v_b_8
sp4_v_b_9
sp4_v_b_10
sp4_v_b_11
sp4_v_b_12
sp4_v_b_13
sp4_v_b_14
sp4_v_b_15
sp4_v_b_15
sp4_v_b_17
sp4_v_b_18
sp4_v_b_19
sp4_v_b_20
sp4_v_b_21
sp4_v_b_22
sp4_v_b_23
sp4_v_b_24
sp4_v_b_25
sp4_v_b_26
sp4_v_b_27
sp4_v_b_28
sp4_v_b_29
sp4_v_b_30
sp4_v_b_31
sp4_v_b_32
sp4_v_b_33
sp4_v_b_34
sp4_v_b_35
sp4_v_b_36
sp4_v_b_37
sp4_v_b_38
sp4_v_b_39
sp4_v_b_40
sp4_v_b_41
sp4_v_b_42
sp4_v_b_43
sp4_v_b_44
sp4_v_b_45
sp4_v_b_46
sp4_v_b_47
sp4_r_v_b_36 ... sp4_r_v_b_47
sp4_r_v_b_24 ... sp4_r_v_b_35
sp4_r_v_b_12 ... sp4_r_v_b_23
sp4_r_v_b_0 ... sp4_r_v_b_11

Figure 16: **Span12 Connection Resources**



Interconnect Wires

span4 wires

- **4x12 sp4_(h) connected to the left**
- **4x12 sp4_v (v) connected above**
- **4x12 sp4_r_v (r) connected to the right**

span12 wires

- **12x2 sp12_h (H) connected to the left**
- **12x2 sp12_v (V) connected to the right**

global wires

- **8 global (G)**

Total: 192+8=200 wires

Connection Matrix

There is a 16:1 Mux connecting spans to the local tracks

Enable

g=0, lut=0

B0[14]	B1[14]	B1[15]	B1[16]	B1[17]	Function	Source-Net	Destination-Net
0	0	0	0	1	buffer	sp4_r_v_b_24	local_g0_0
0	0	0	1	1	buffer	sp12_h_r_8	local_g0_0
0	0	1	0	1	buffer	neigh_op_bot_0	local_g0_0
0	0	1	1	1	buffer	sp4_v_b_16	local_g0_0
0	1	0	0	1	buffer	sp4_r_v_b_35	local_g0_0
0	1	0	1	1	buffer	sp12_h_r_16	local_g0_0
0	1	1	0	1	buffer	neigh_op_top_0	local_g0_0
0	1	1	1	1	buffer	sp4_h_r_0	local_g0_0
1	0	0	0	1	buffer	lutff_0/out	local_g0_0
1	0	0	1	1	buffer	sp4_v_b_0	local_g0_0
1	0	1	0	1	buffer	neigh_op_lft_0	local_g0_0
1	0	1	1	1	buffer	sp4_h_r_8	local_g0_0
1	1	0	0	1	buffer	neigh_op_bnr_0	local_g0_0
1	1	0	1	1	buffer	sp4_v_b_8	local_g0_0
1	1	1	0	1	buffer	sp12_h_r_0	local_g0_0
1	1	1	1	1	buffer	sp4_h_r_16	local_g0_0

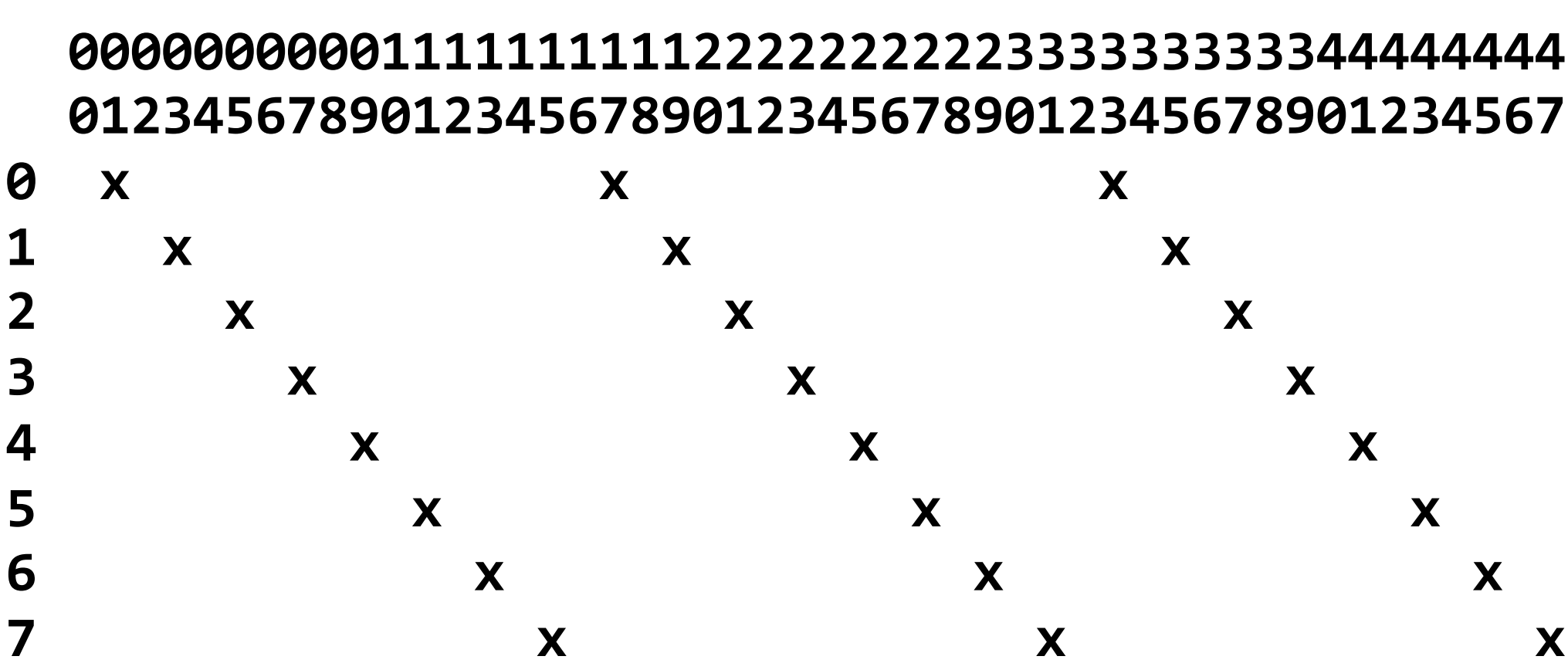
1 connection to LUT output
4 connections to nearest neighbors
11 connections to spans
The CM is different for each local track

LUT Outputs to Interconnect

```
sp4_h_r[i]=lut[j]
sp4_v_b[i]=lut[j]
sp4_r_v_b[i]=lut[j]

  0000000000111111111122222222223333333334444444
  0123456789012345678901234567890123456789012345
0 x                 x                   x
1   x                 x                   x
2     x                 x                   x
3       x                 x                   x
4         x                 x                   x
5           x                 x                   x
6             x                 x                   x
7               x                 x                   x
NB even wires, connect to right and bottom
Connections can be set independently
```

sp4_r_v_b[i]=lut[j]



NB odd wires, right and bottom
Connections can be set independently
Must be careful about driving wires twice!

Switch box

Figure 14: **Span4 Connection Resources**

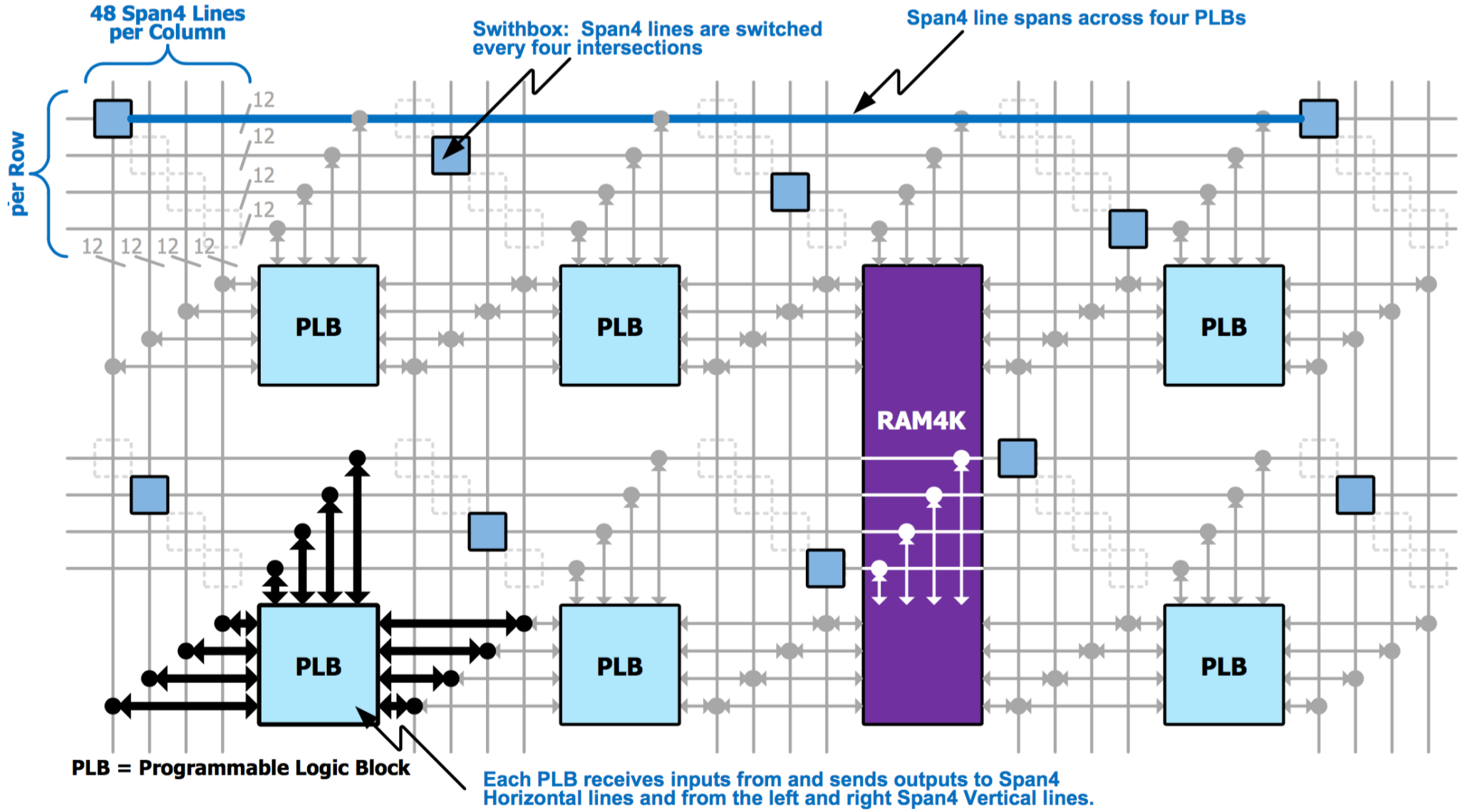


Figure 15: **Span4 Switchbox Connections**

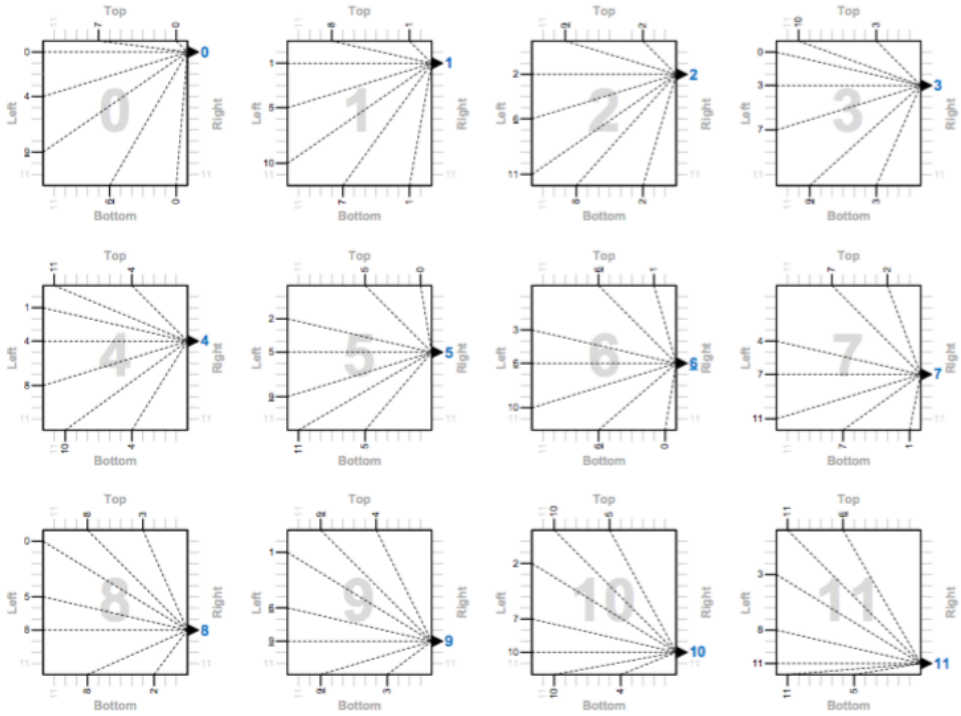


Table 11: **Span4 Switchbox Connections**

OUTPUT	←	INPUT CONNECTION						
Right	←	Top		Left			Bottom	
Top	←	Left		Bottom			Right	
Left	←	Bottom		Right			Top	
Bottom	←	Right		Top			Left	
0	←	7	0	9	0	4	0	6
1	←	8	1	10	1	5	1	7
2	←	9	2	11	2	6	2	8
3	←	10	3	0	3	7	3	9
4	←	11	4	1	4	8	4	10
5	←	0	5	2	5	9	5	11
6	←	1	6	3	6	10	6	0
7	←	2	7	4	7	11	7	1
8	←	3	8	5	8	0	8	2
9	←	4	9	6	9	1	9	3
10	←	5	10	7	10	2	10	4
11	←	6	11	8	11	3	11	5

span12

Figure 17: Span12 Switchbox Connections

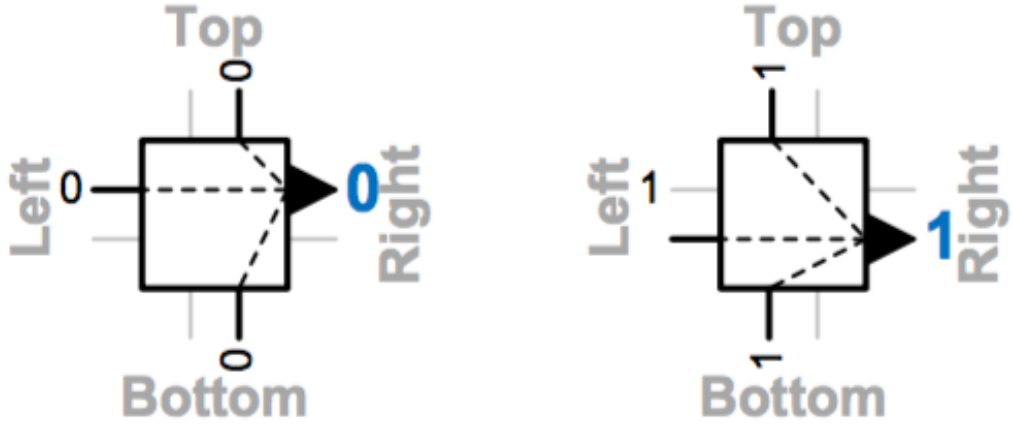


Table 12: **Span12 Switchbox Connections**

OUTPUT	←	INPUT CONNECTION		
Right	←	Top	Left	Bottom
Top	←	Left	Bottom	Right
Left	←	Bottom	Right	Top
Bottom	←	Right	Top	Left
0	←	0	0	0
1	←	1	1	1

	IO (1 17)	IO (2 17)	IO (3 17)	IO (4 17)	IO (5 17)	IO (6 17)	IO (7 17)	IO (8 17)	IO (9 17)	IO (10 17)	IO (11 17)	IO (12 17)	
IO (0 16)	LOGIC (1 16)	LOGIC (2 16)	RAMT (3 16)	LOGIC (4 16)	LOGIC (5 16)	LOGIC (6 16)	LOGIC (7 16)	LOGIC (8 16)	LOGIC (9 16)	RAMT (10 16)	LOGIC (11 16)	LOGIC (12 16)	IO (13 16)
IO (0 15)	LOGIC (1 15)	LOGIC (2 15)	RAMB (3 15)	LOGIC (4 15)	LOGIC (5 15)	LOGIC (6 15)	LOGIC (7 15)	LOGIC (8 15)	LOGIC (9 15)	RAMB (10 15)	LOGIC (11 15)	LOGIC (12 15)	IO (13 15)
IO (0 14)	LOGIC (1 14)	LOGIC (2 14)	RAMT (3 14)	LOGIC (4 14)	LOGIC (5 14)	LOGIC (6 14)	LOGIC (7 14)	LOGIC (8 14)	LOGIC (9 14)	RAMT (10 14)	LOGIC (11 14)	LOGIC (12 14)	IO (13 14)
IO (0 13)	LOGIC (1 13)	LOGIC (2 13)	RAMB (3 13)	LOGIC (4 13)	LOGIC (5 13)	LOGIC (6 13)	LOGIC (7 13)	LOGIC (8 13)	LOGIC (9 13)	RAMB (10 13)	LOGIC (11 13)	LOGIC (12 13)	IO (13 13)
IO (0 12)	LOGIC (1 12)	LOGIC (2 12)	RAMT (3 12)	LOGIC (4 12)	LOGIC (5 12)	LOGIC (6 12)	LOGIC (7 12)	LOGIC (8 12)	LOGIC (9 12)	RAMT (10 12)	LOGIC (11 12)	LOGIC (12 12)	IO (13 12)
IO (0 11)	LOGIC (1 11)	LOGIC (2 11)	RAMB (3 11)	LOGIC (4 11)	LOGIC (5 11)	LOGIC (6 11)	LOGIC (7 11)	LOGIC (8 11)	LOGIC (9 11)	RAMB (10 11)	LOGIC (11 11)	LOGIC (12 11)	IO (13 11)
IO (0 10)	LOGIC (1 10)	LOGIC (2 10)	RAMT (3 10)	LOGIC (4 10)	LOGIC (5 10)	LOGIC (6 10)	LOGIC (7 10)	LOGIC (8 10)	LOGIC (9 10)	RAMT (10 10)	LOGIC (11 10)	LOGIC (12 10)	IO (13 10)
IO (0 9)	LOGIC (1 9)	LOGIC (2 9)	RAMB (3 9)	LOGIC (4 9)	LOGIC (5 9)	LOGIC (6 9)	LOGIC (7 9)	LOGIC (8 9)	LOGIC (9 9)	RAMB (10 9)	LOGIC (11 9)	LOGIC (12 9)	IO (13 9)
IO (0 8)	LOGIC (1 8)	LOGIC (2 8)	RAMT (3 8)	LOGIC (4 8)	LOGIC (5 8)	LOGIC (6 8)	LOGIC (7 8)	LOGIC (8 8)	LOGIC (9 8)	RAMT (10 8)	LOGIC (11 8)	LOGIC (12 8)	IO (13 8)
IO (0 7)	LOGIC (1 7)	LOGIC (2 7)	RAMB (3 7)	LOGIC (4 7)	LOGIC (5 7)	LOGIC (6 7)	LOGIC (7 7)	LOGIC (8 7)	LOGIC (9 7)	RAMB (10 7)	LOGIC (11 7)	LOGIC (12 7)	IO (13 7)
IO (0 6)	LOGIC (1 6)	LOGIC (2 6)	RAMT (3 6)	LOGIC (4 6)	LOGIC (5 6)	LOGIC (6 6)	LOGIC (7 6)	LOGIC (8 6)	LOGIC (9 6)	RAMT (10 6)	LOGIC (11 6)	LOGIC (12 6)	IO (13 6)
IO (0 5)	LOGIC (1 5)	LOGIC (2 5)	RAMB (3 5)	LOGIC (4 5)	LOGIC (5 5)	LOGIC (6 5)	LOGIC (7 5)	LOGIC (8 5)	LOGIC (9 5)	RAMB (10 5)	LOGIC (11 5)	LOGIC (12 5)	IO (13 5)
IO (0 4)	LOGIC (1 4)	LOGIC (2 4)	RAMT (3 4)	LOGIC (4 4)	LOGIC (5 4)	LOGIC (6 4)	LOGIC (7 4)	LOGIC (8 4)	LOGIC (9 4)	RAMT (10 4)	LOGIC (11 4)	LOGIC (12 4)	IO (13 4)
IO (0 3)	LOGIC (1 3)	LOGIC (2 3)	RAMB (3 3)	LOGIC (4 3)	LOGIC (5 3)	LOGIC (6 3)	LOGIC (7 3)	LOGIC (8 3)	LOGIC (9 3)	RAMB (10 3)	LOGIC (11 3)	LOGIC (12 3)	IO (13 3)
IO (0 2)	LOGIC (1 2)	LOGIC (2 2)	RAMT (3 2)	LOGIC (4 2)	LOGIC (5 2)	LOGIC (6 2)	LOGIC (7 2)	LOGIC (8 2)	LOGIC (9 2)	RAMT (10 2)	LOGIC (11 2)	LOGIC (12 2)	IO (13 2)
IO (0 1)	LOGIC (1 1)	LOGIC (2 1)	RAMB (3 1)	LOGIC (4 1)	LOGIC (5 1)	LOGIC (6 1)	LOGIC (7 1)	LOGIC (8 1)	LOGIC (9 1)	RAMB (10 1)	LOGIC (11 1)	LOGIC (12 1)	IO (13 1)
	IO (1 0)	IO (2 0)	IO (3 0)	IO (4 0)	IO (5 0)	IO (6 0)	IO (7 0)	IO (8 0)	IO (9 0)	IO (10 0)	IO (11 0)	IO (12 0)	

demo/and.v

Interleaving

Why is this good?

- **Sparsity**

Details?

- **Cross-overs in spans**
- **Permutation to LUTs within a PLB**
- **Connection matrix interleave (spans to local)**
- **Switch box connections**

FPGA Interconnects

Problem

- Mapping circuits to a planar graph is difficult
- For example, laying out a binary tree is hard
- Rent's Law: $T = g^p$ $(0.5<p<0.8)$

Solution

- Provide lots of connectivity

Consequences

- Interconnect consumes most of the chip area and power
- Long wires increase long delays

Better FPGAs?

Power efficiency

- **ASIC = 10*FPGA = 10*GPU = 10*CPU**
- **What if ASIC = 2*FPGA?**

More efficient

- **Coarse-grained PLB (e.g. ALUs)**
- **Simpler, wider networks**
- **Pipeline networking**