Connecting RAMs with EFLX Array

EFLX® embedded FPGA cores

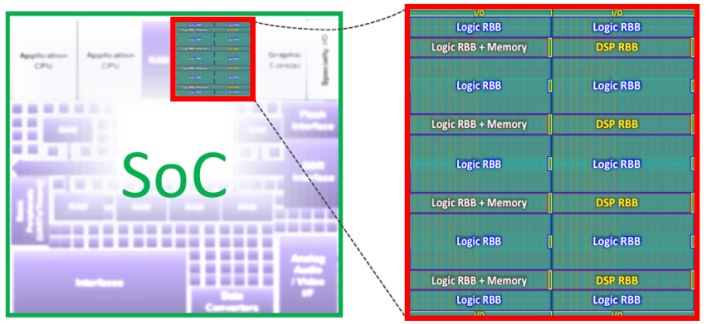
Shuying Fan

EFLX Embedded FPGA Overview

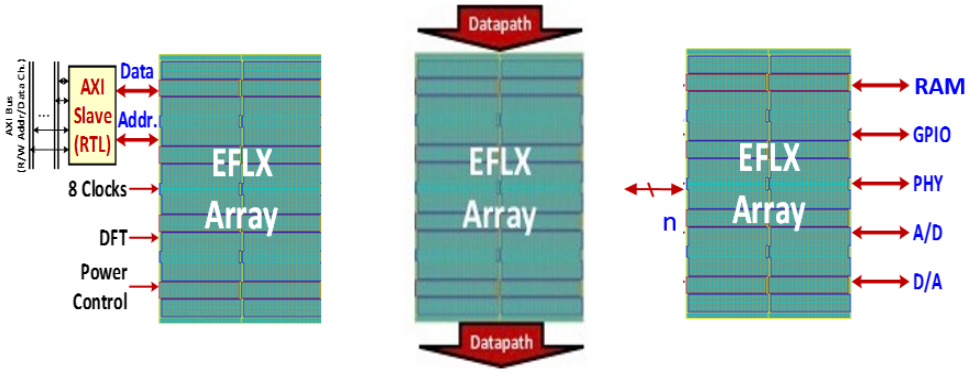
EFLX Embedded FPGA Overview

- ✓ Reconfigure SoC/MCU any time
- ✓ Integrate EFLX any where
  - Breakthrough technology: double density, high utilization, minimum metal
  - Multiple ways to integrate within a chip
- ✓ Scale to any size
  - From 150 to 200K LUT4s (Gen2)
  - Proprietary software for P&R and bitstream generation
- ✓ Port to any process
  - Portable to any process within 6 to 8 months
  - Standard cell based logic process

Note: Please refer to www.flex-logix.com to learn more about EFLX.



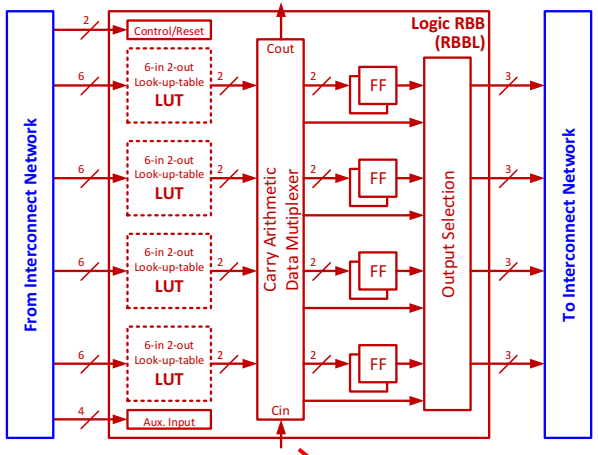
EFLX Embedded FPGA



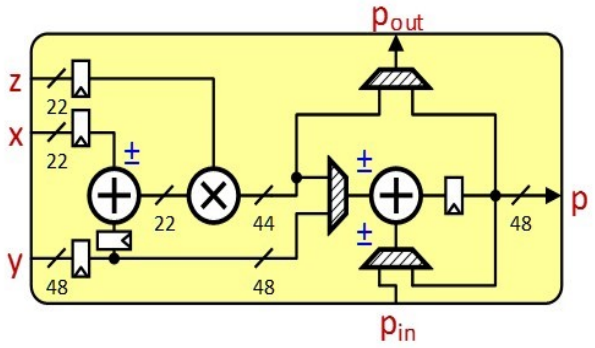
Multiple Ways to Integrate

EFLX IP Core Versions and Resource Types (1)

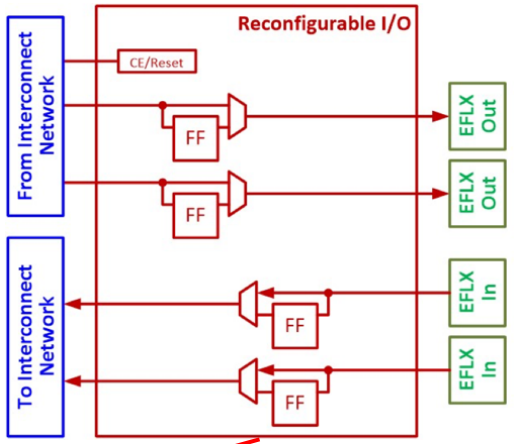
Logic

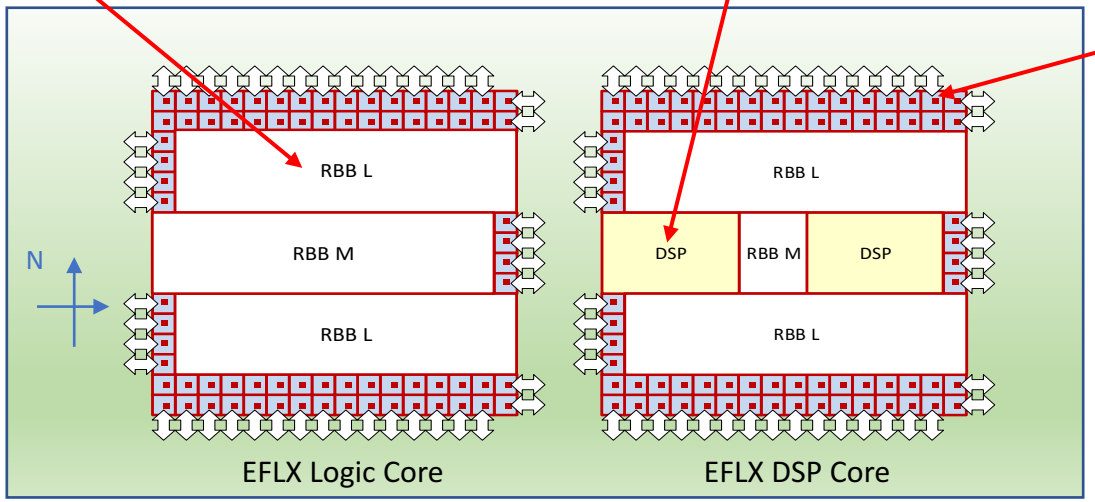


DSP MAC



IOB





EFLX IP Core Versions and Resource Types (2)

- ✓ 2 EFLX core versions
  - Logic Cores: RBBLs, RBBMs, IOBs
  - DSP Cores: RBBLs, RBBMs, DSP MACs, IOBs
- ✓ Same footprint, multiple cores can be tiled together to form a bigger array
- ✓ RBBMs are RBBLs that support distributed RAM or shift register functions
- ✓ IOBs are placed at the core peripheral
- ✓ More IOBs on north and south sides
- ✓ Fixed amount of resources within each EFLX core
- ✓ Adjustable amount of resources in an array

Flexible RAM Integration with EFLX

Implement RTL Design with RAMs using EFLX

- ✓ For RTL designs with no/small amount of distributed RAM resources
  - RBBMs will be used to map distributed RAMs
  - EFLX array in a feasible size can provide sufficient resource to map the design
- ✓ For RTL designs that use large amount of RAM resources (distributed or block RAMS)
  - RBBMs are insufficient to map all RAM resources in any feasible array size
  - Connect external RAMs with EFLX array to provide sufficient resource
  - EFLX Compiler will use external RAMs to map those RAM functions as if they are part of the EFLX cores

External RAMs Allow Maximum Flexibility in RAM Integration

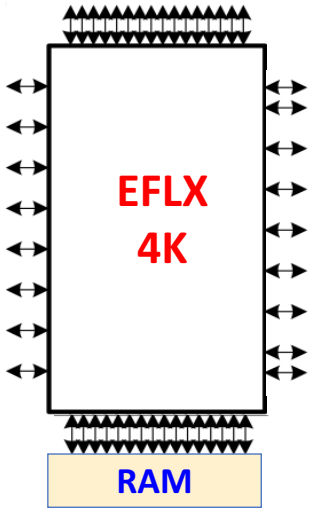
- ✓ External RAMs here refer to RAM resources external to EFLX array
- ✓ It can be any user defined on-chip RAM blocks *
  - Any type: single-port, 2-port, dual-port
  - Any width
  - Any size
  - ECC, parity, or neither
- ✓ Maximum flexibility
  - Use EFLX Compiler to customize external RAM usage and integration for maximum flexibility and best utilization
  - No physical design of the RAM involved

*EFLX Compiler currently only supports one size for each type of external RAM; multiple sizes will be supported in the future

2 Ways to Connect

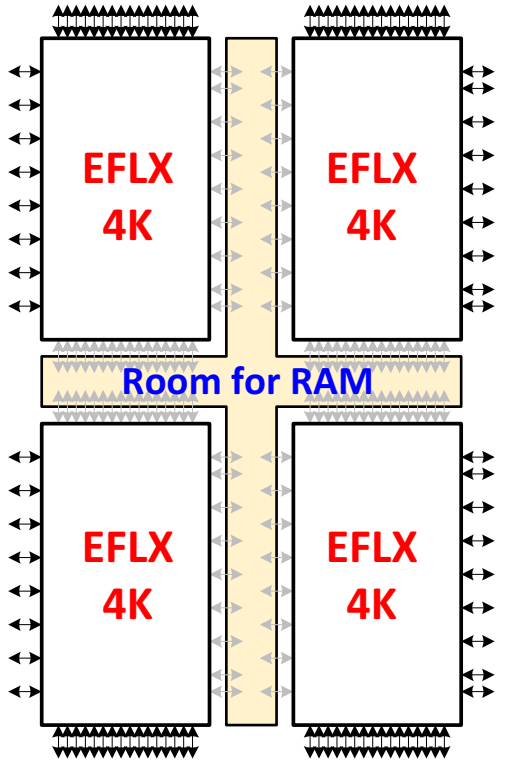
✓Connect RAM around EFLX array

- Use peripheral IOBs
- Longer routing path may result in worse timing
- Slightly smaller EFLX array size



✓Connect RAM in the EFLX array (Recommended)

- Use internal IOBs that are not accessible to other signals
- Shorter routing path may give better timing
- Slightly bigger EFLX array size

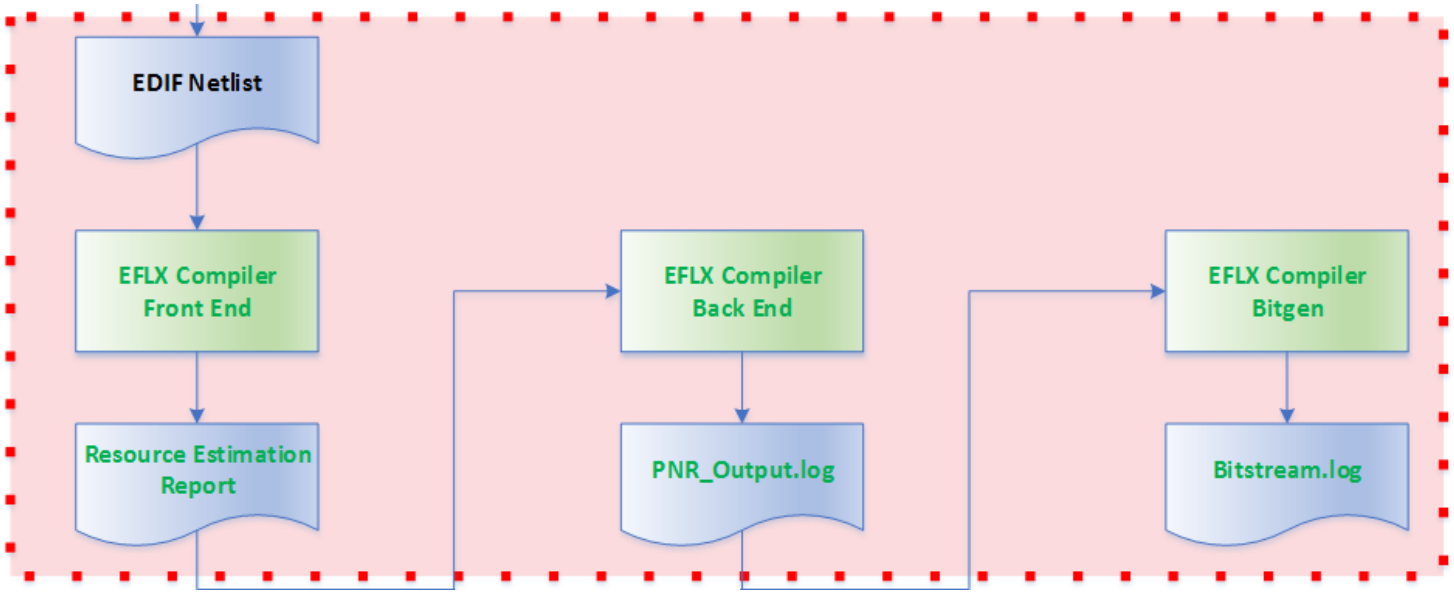


EFLX Compiler Flow for Customizable, Automatic RAM Connection

EFLX Compiler Flow for RAM Connection

✓ EFLX Compiler Flow

- Front-end EFLX Mapper: Logic Mapping and Resource Estimation, create EFLX floorplan and RAM Definition file
- Back-end EFLX P&R: Apply EFLX floorplan and RAM Definition file for detailed place-and-route and automatic RAM connection
- Bitgen



EFLX Compiler Logic Mapping

- ✓ LUTs, FFs, other logic functions --> RBBLs
- ✓ DSP functions --> DSP RBBs
- ✓ Distributed RAMs and Shift Registers (small amount) --> RBBMs
- ✓ Input and Output Ports --> IOBs
- ✓ Block RAMs or large amount of distributed RAMs --> External RAMs*

Resources within EFLX cores

Note:

1. EFLX Compiler can pack RAMs into external RAMs as long as the port is compatible:
    1) Single port RAM can pack into SP, 2P, or DP external RAMs
    2) Two-port RAM can pack into 2P or DP external RAMs
    3) Dual-port RAM can only pack into DP external RAMs
2. EFLX Compiler supports port width extension and address extension.

Identify Usage of All Resource Types

EFLX Compiler Resource Usage Report

Top level module name: dft_top
Resource usage:
32 DSP RBBs DSP MAC
13 BRAM RBBs External RAM
13 of which are 256x32b dual-port RAM
1124 Logic 6-LUT RBBs RBBL
336 of which require RAM and/or shift-register functions RBBM
3656 RBB LUTs utilized (up to 4 per RBB)
1273 of which are 6-input LUTs
784 of which are dual 5-input LUTs
4440 LUTs total
2774 RBB FFs utilized (up to 8 per RBB)
2560 of which are LUT-FF combination
510 of which are dual-FFs
145 of which are dual-LUT-dual-FF combination
3284 FFs total
66 I/O RBBs IOB
66 Input pins
65 Output pins
1 Clock domain
Clock 0: net 7869, clk_c, Fanout: 1001 RBBs

Based on DSP and Logic RBB requirement:
need 16 DSP and 31 Logic-Memory EFLX-100 Tiles.
or...
need 1 DSP and 2 Logic-Memory EFLX-4K Tiles.

Resource Type	Amount
DSP MAC	32
External RAM	Not Specified
RBBL	1124
RBBM	336
IOB	66

- ✓ The actual RAM type and size are defined by users with a ram definition file.
- ✓ A RAM Def file always comes with a floorplan, and they together must provide sufficient amount of resources of all types listed in the table above.

This can only provide sufficient resource inside the EFLX cores. The actual floorplan may need more tiles for external RAM connections.

RAM Definition File

- ✓ Users can manually make the connection or use a RAM Definition file (recommended) to automatically connect external RAMs to EFLX array
- ✓ A RAM Def file is composed of two parts: RAM definition and RAM instantiation
- ✓ RAM definition specifies
  - RAM name
  - RAM type, width, depth
  - RAM port names
  - EFLX IO locations of all RAM ports
- ✓ The same RAM can be instantiated multiple times in the RAM instantiation part with an offset coordinate
- ✓ Multiple external RAMs can be defined and instantiated in the RAM Def file

RAM Definition Example

```
RAM_NAME    = RAM_DP_N              # string only. Example: RAM_DP on N side of EFLX core
{
RAM_TYPE    = DP    # SP: single-port; 2P: two-port(1R1W); DP: dual-port
DATA_WIDTH  = 16 # 16-bit wide
ADDRESS_LEN = 9    # 512-word deep

D_IN_A                  = DA            # Name for Data In Port A
ADDR_A                  = AA            # Name for Address Port A
Q_OUT_A                 = QA            # Name for Data Out Port A
CLK_A                   = CLKA          # Name for Clock Port A
CLOCK_EN_A  = !CEBA                     # Name for Clock Enable Port A
WRITE_EN_A  = !WEBA                     # Name for Write Enable Port A

D_IN_B                  = DB            # Name for Data In Port B
ADDR_B                  = AB            # Name for Address Port B
Q_OUT_B                 = QB            # Name for Data Out Port B
CLK_B                   = CLKB          # Name for Clock Port B
CLOCK_EN_B  = !CEBB                     # Name for Clock Enable Port B
WRITE_EN_B  = !WEBB                     # Name for Write Enable Port B

AUX_TIE_ON  = !SLP                      # Signal to enable when RAM is ON (e.g. RAM_EN, or !SLEEP, or !SHUTDOWN)
AUX_TIE_ON  = !SD                       # Signal to enable when RAM is ON (e.g. RAM_EN, or !SLEEP, or !SHUTDOWN)

EFLX_CLK chip_tile_x=0, chip_tile_y=0, clk_side=N
  Output=0, pin=CLKA
  Output=1, pin=CLKB
EFLX_IOB chip_tile_x=0, chip_tile_y=0, chip_x=0, chip_y=0
  Output=0, pin=DA[13], output_delay=1000
  Output=1, pin=DA[12], output_delay=1000
   Input=0, pin=QA[13],  input_delay=1000
   Input=1, pin=QA[12],  input_delay=1000

< … NOT SHOWN TO SAVE SPACE …>

EFLX_IOB chip_tile_x=0, chip_tile_y=0, chip_x=14, chip_y=1
  Output=0, pin=SLP, output_delay=1000
  Output=1, pin=SD,  output_delay=1000
}
```

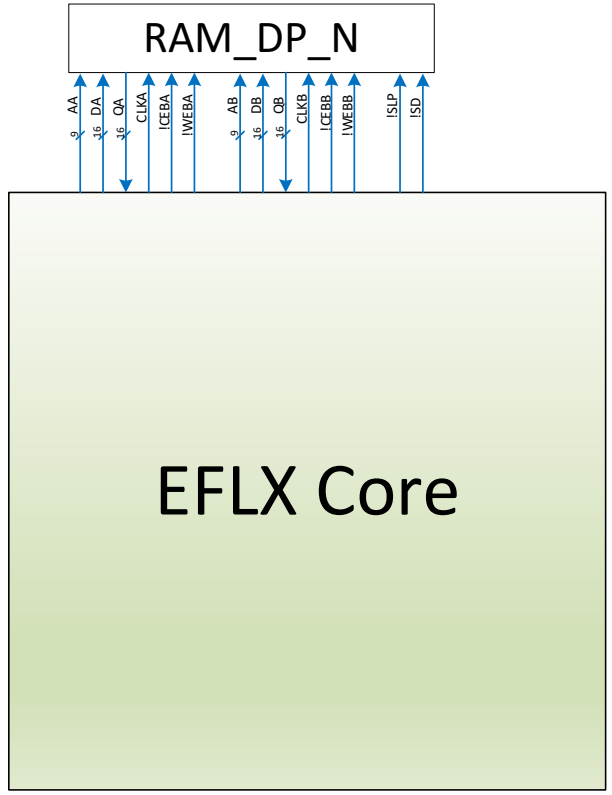
RAM name

RAM type, width, depth

RAM port names

RAM connections with EFLX IOs

RAM definition part defines the RAM and how it connects to a single EFLX core.



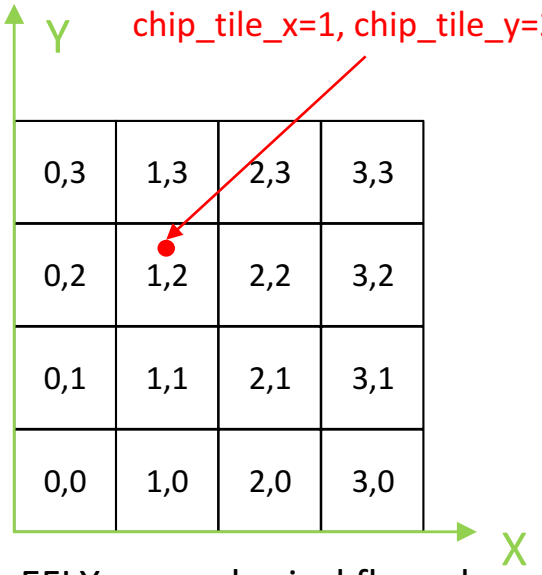
Define External RAMs in 4 Steps

- ✓ STEP 1. Specify RAM name.
- ✓ STEP 2. Specify RAM type and size.
  - SP: single port
  - 2P: two-port (1R1W)
  - DP: dual-port
- ✓ STEP 3. Specify RAM port names.
  - Active-low signals signified by '!'
  - Support ports to be tied off to constant values
- ✓ STEP 4. Specify RAM pin connections with a single EFLX core
  - EFLX_CLK: specify clock connections
  - EFLX_IOB: specify all other IO pin connections

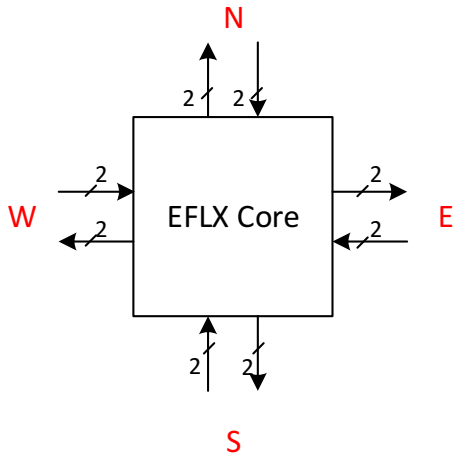
Define External RAM Clock Connection

Sample Code

```
EFLX_CLK chip_tile_x=0, chip_tile_y=0, clk_side=N
  Output=0, pin=CLKA
  Output=1, pin=CLKB
```



EFLX array physical floorplan with x, y coordinates.



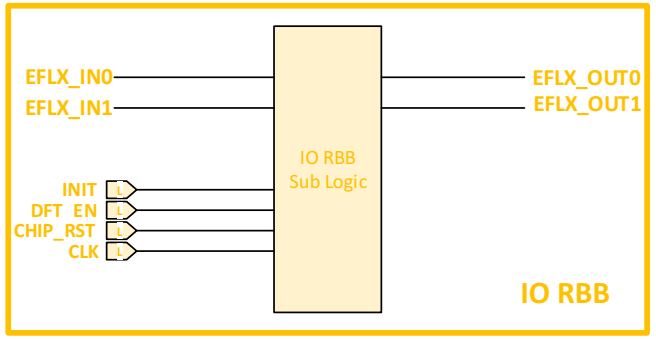
- ✓ EFLX_CLK: specify it is a clock connection
- ✓ chip_tile_x, chip_tile_y: specify EFLX core location within the array
- ✓ clock_side
  - Clocks can come in/out from all 4 directions of each EFLX core.
  - Use W, E, N, S to represent the 4 directions.
- ✓ Port direction and number
  - The port direction is in reference to EFLX array. For example, the clock input CLKA to external RAMs is connected to the output port of the specified EFLX core.
  - There are 2 clock inputs and 2 clock outputs at each side of an EFLX core.

Note: External RAMs may take significant amount of clock routing resources and block clock routing for the design. In this case, an IO Specification can be created to resolve this issue. Please refer to the “Guidelines for EFLX Clock Connection” session of the presentation for more details.

Define Other External RAM Connections

Sample Code

```
EFLX_IOB chip_tile_x=0, chip_tile_y=0, chip_x=0, chip_y=0
  Output=0, pin=DA[13], output_delay=1000
  Output=1, pin=DA[12], output_delay=1000
   Input=0, pin=QA[13],  input_delay=1000
   Input=1, pin=QA[12],  input_delay=1000
```

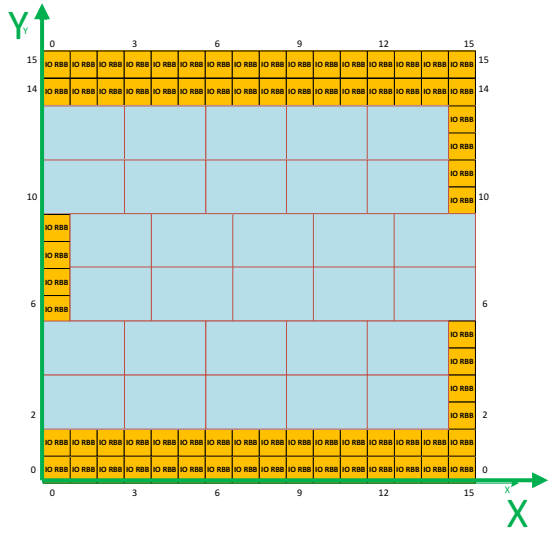


- ✓ EFLX_IOB: specify it is a connection with EFLX IOB
- ✓ chip_tile_x, chip_tile_y: specify EFLX core location within the array
- ✓ chip_x, chip_y: specify IOB location within the core
  - Each IOB has a coordinate (see next page for detail)
- ✓ Port direction and number
  - The port direction is in reference to EFLX array. For example, the data output QA[12] of external RAMs is connected to the input port of the specified EFLX core.
  - There are 2 input ports and 2 output ports of each IO RBB.

Recommendations:

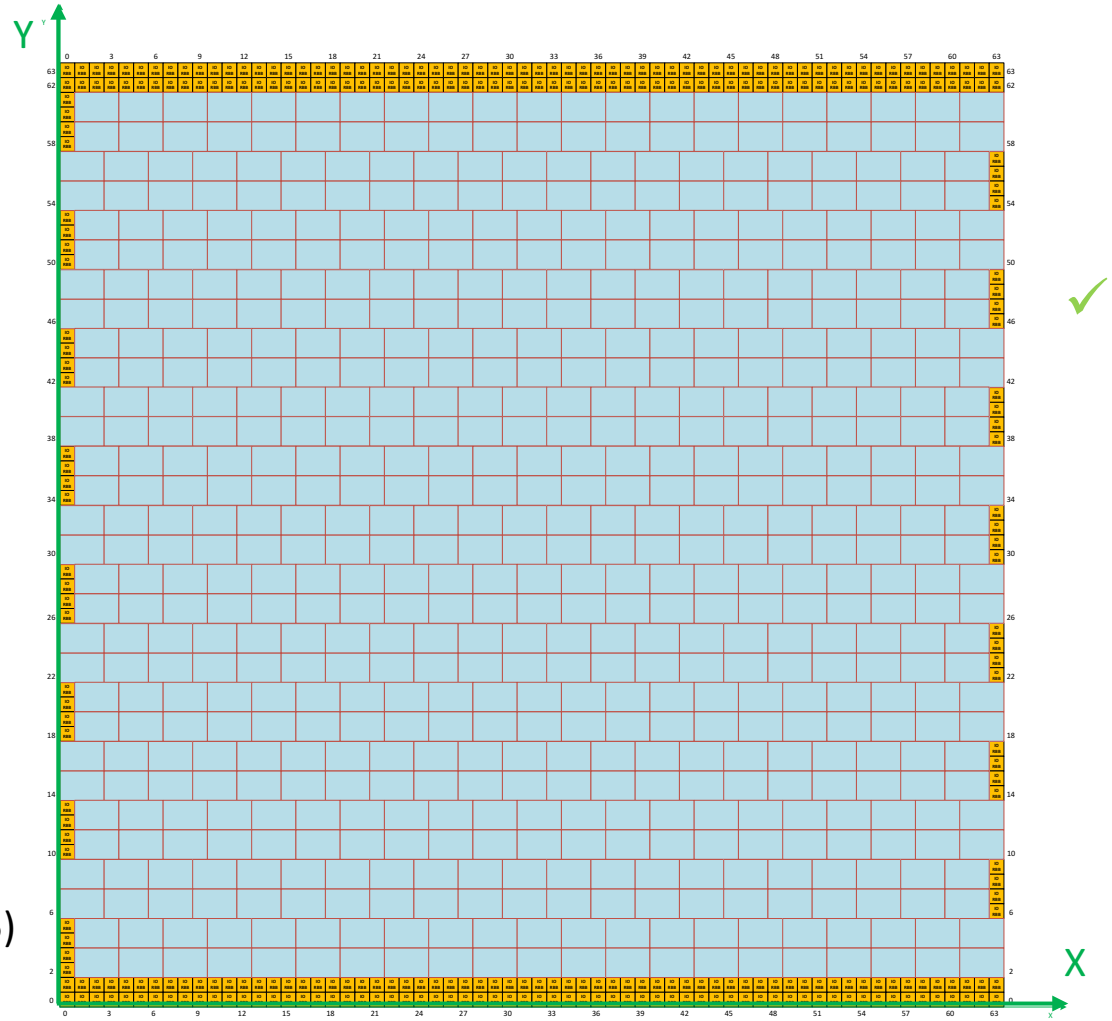
1. Connect external RAMs to the North or South side of the core because there are more IOBs on these two sides.
2. Put data output and input pins in the same IOBs
3. Use IOBs in order and count the total number of IOBs used by the specified RAM. It's easy to determine the valid offset coordinates during RAM instantiation.

EFLX IO RBB Coordinates



✓EFLX-100 core

- 76 IOBs in total
- Horizontal: X Axis, 0 to 15
- Vertical: Y Axis, 0 to 15
- N side: 32 IOBs, coordinate ranging from (0,14) to (15,15)
- S side: 32 IOBs, coordinate ranging from (0,0) to (15,1)



✓EFLX-4K core

- 316 IOBs in total
- Horizontal: X Axis, 0 to 63
- Vertical: Y Axis, 0 to 63
- N side: 128 IOBs, coordinate ranging from (0,62) to (63,63)
- S side: 128 IOBs, coordinate ranging from (0,0) to (63,1)

RAM Instantiation Example

EFLX_RAM	RAM_DP_N	chip_tile_x=0, chip_tile_y=0,	chip_x=0, chip_y=62
EFLX_RAM	RAM_DP_N	chip_tile_x=0, chip_tile_y=0,	chip_x=16, chip_y=62
EFLX_RAM	RAM_DP_N	chip_tile_x=0, chip_tile_y=1,	chip_x=32, chip_y=62
EFLX_RAM	RAM_DP_N	chip_tile_x=0, chip_tile_y=1,	chip_x=48, chip_y=62
……			
EFLX_RAM	RAM_SP_N	chip_tile_x=1, chip_tile_y=0,	chip_x=0, chip_y=62
EFLX_RAM	RAM_SP_N	chip_tile_x=1, chip_tile_y=0,	chip_x=16, chip_y=62
EFLX_RAM	RAM_SP_N	chip_tile_x=1, chip_tile_y=1,	chip_x=32, chip_y=62
EFLX_RAM	RAM_SP_N	chip_tile_x=1, chip_tile_y=1,	chip_x=48, chip_y=62

RAM Name

EFLX core location in an array

IOB coordinate on the specified EFLX core location

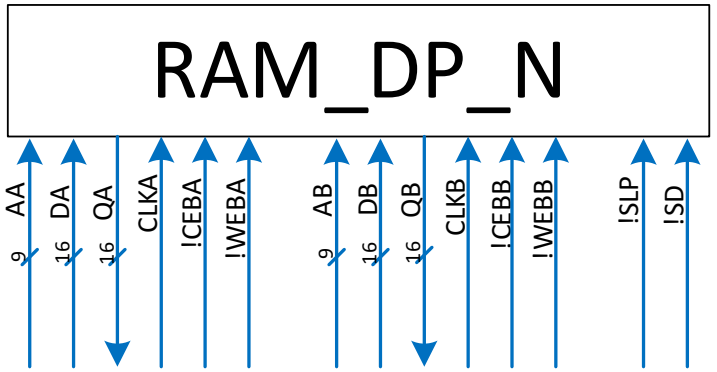
Instantiate External RAMs

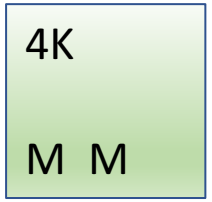
- ✓ EFLX Compiler allows users to define a RAM once and instantiate it multiple times with an offset coordinate to specify new IO connections within an EFLX array.
- ✓ Each instantiation has an offset coordinate
  - (chip_tile_x, chip_tile_y) – offset coordinate of EFLX core location in an array
  - (chip_x, chip_y) – offset coordinate of IOB location within the specified EFLX core
- ✓ The exact IO pin location of each instance is the sum of the IO location during RAM definitions, plus its offset coordinate.

Example and Guidelines

An Example

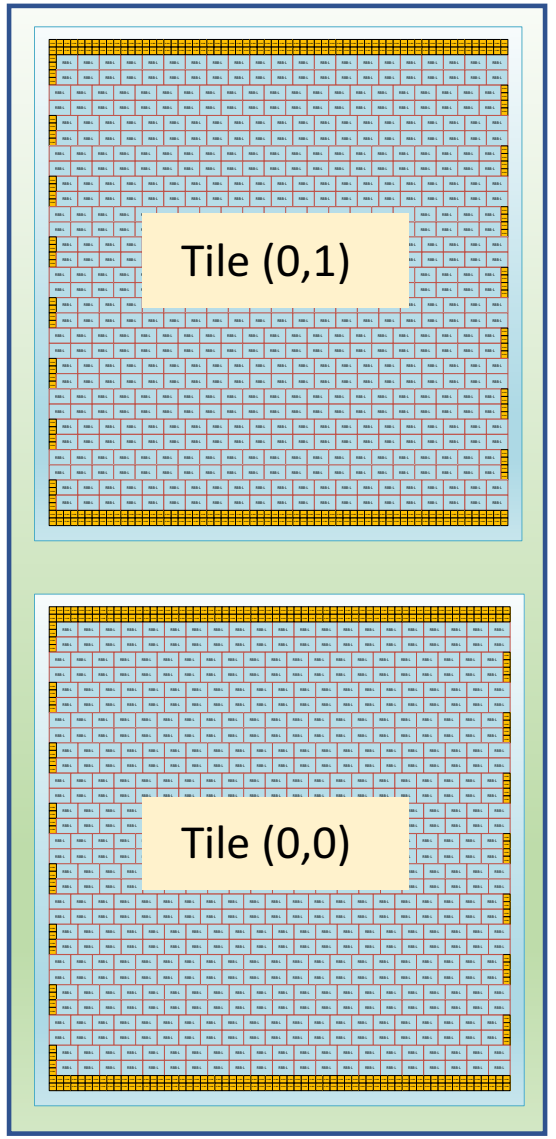
- ✓Same dual-port RAM showed in previous slide
- ✓Use a 1x2 EFLX-4K array
- ✓Assume using logic cores





Floorplan file (.fp)
for EFLX Compiler

90-degree clockwise rotation



Actual physical floorplan

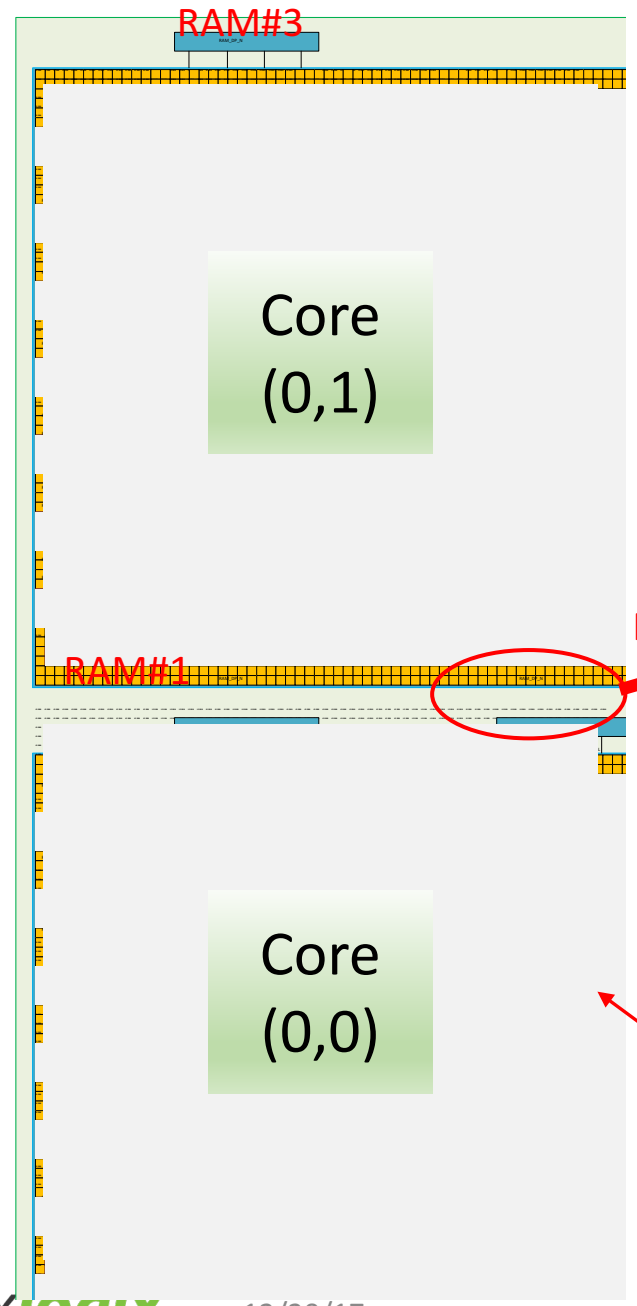
Note: Logic and DSP cores have exact same footprint.
The number of location of each IO RBB is the same.

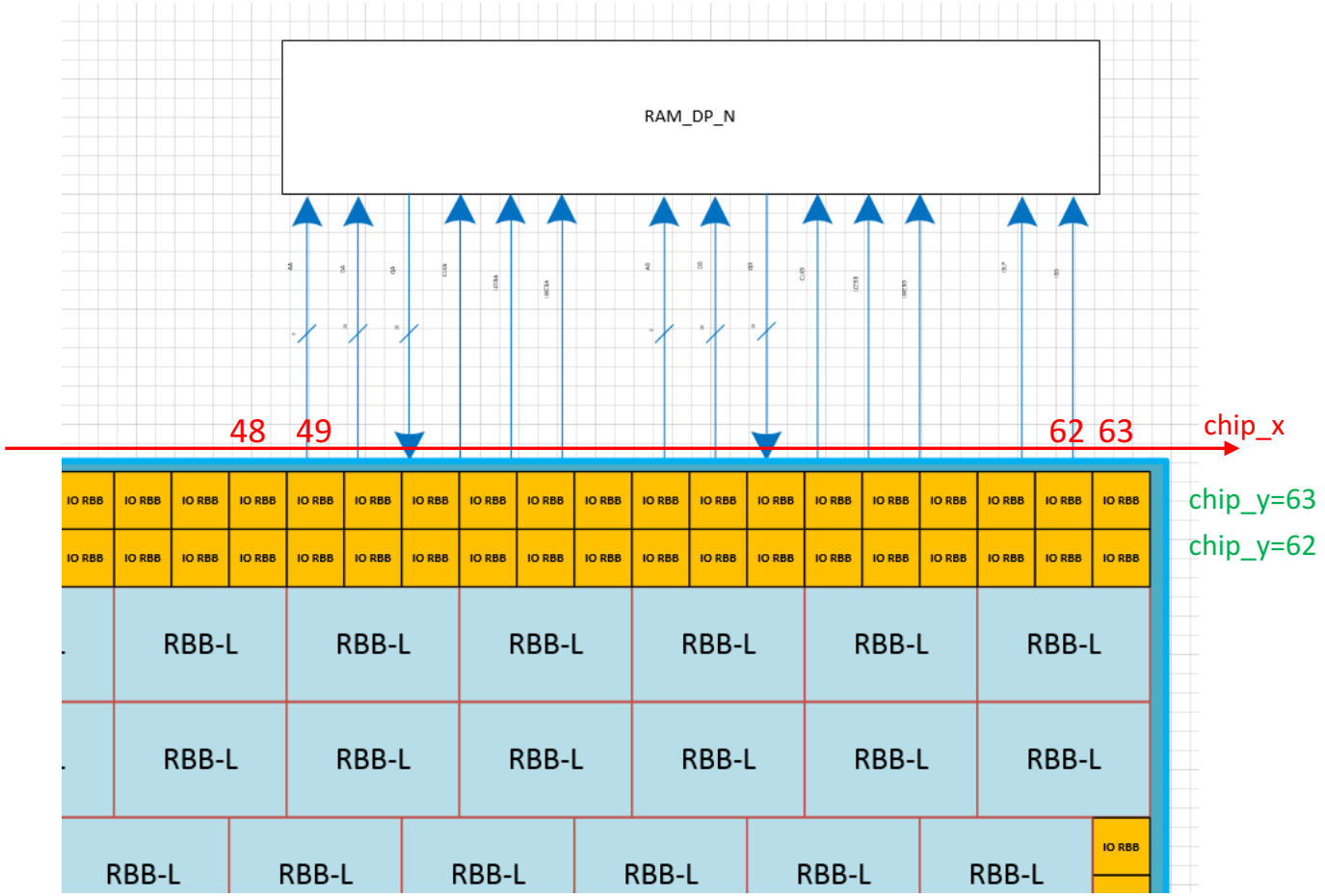
An Example – Cont.

```
RAM_NAME    = RAM_DP_N
{
< … NOT SHOWN TO SAVE SPACE …>
EFLX_IOB chip_tile_x=0, chip_tile_y=0, chip_x=0, chip_y=0
  Output=0, pin=DA[13], output_delay=1000
  Output=1, pin=DA[12], output_delay=1000
   Input=0, pin=QA[13],  input_delay=1000
   Input=1, pin=QA[12],  input_delay=1000
< … NOT SHOWN TO SAVE SPACE …>
EFLX_IOB chip_tile_x=0, chip_tile_y=0, chip_x=14, chip_y=1
  Output=0, pin=SLP, output_delay=1000
  Output=1, pin=SD,  output_delay=1000
}
EFLX_RAM RAM_DP_N chip_tile_x=0, chip_tile_y=0, chip_x=16, chip_y=62 External RAM#1
EFLX_RAM RAM_DP_N chip_tile_x=0, chip_tile_y=0, chip_x=48, chip_y=62 External RAM#2
EFLX_RAM RAM_DP_N chip_tile_x=0, chip_tile_y=1, chip_x=16, chip_y=62 External RAM#3
< … NOT SHOWN TO SAVE SPACE …>
```

- ✓ Calculate IOB location for each external RAM instantiations
- ✓ RAM#1: Connect to EFLX core (0,0) in the array and the IOB coordinates start at (16,62) to (28,63)
- ✓ RAM#2: Connect to EFLX core (0,0) in the array and the IOB coordinates start at (48,62) to (62,63)
- ✓ RAM#3: Connect to EFLX core (0,1) in the array and the IOB coordinates start at (16,62) to (28,63)

An Example – Cont.





2x1 EFLX-2.5K Array

An Example – Cont.

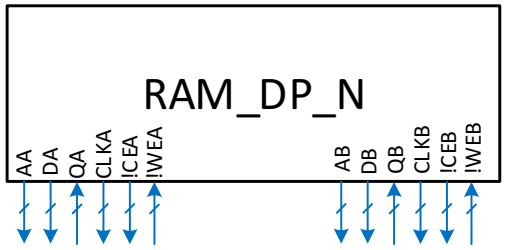
- ✓ Run EFLX Compiler with the –ram_def option

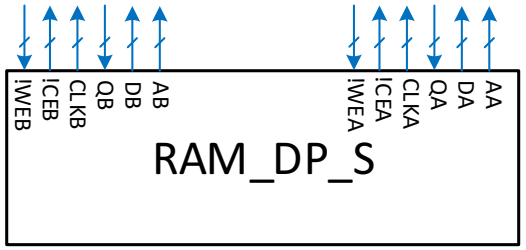
  ./EFLX_COMPILER –edif <FILE_NAME.edf> –fp <FLOORPLAN_NAME.fp> -ram_def <RAM_DEF.ram_def>

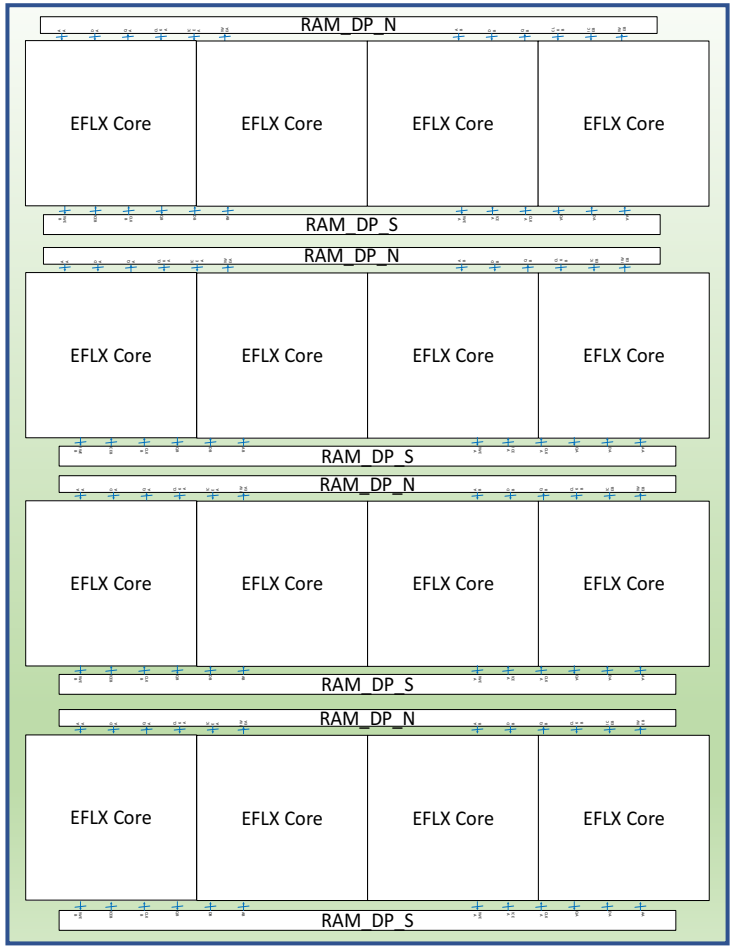
- ✓ Specified external RAMs will be shown as a part of the EFLX resource from the current floorplan during the compiler run. The placement tool will automatically choose the external RAM based on timing and location, just like the rest of the internal RBBs.
- ✓ A file *PNR_EXT_RAM_VERILOG.v* should be created after the EFLX Compiler place-and-route is successfully done.
- ✓ This file automatically connects the EFLX IOs to the ports of the external RAMs based on the RAM def file. It is useful for functional simulation and debugging of external RAM connections with gate-level EFLX Verilog model.

Guidelines for EFLX Clock Connections

- ✓When embedding RAMs within the EFLX array, a large amount of clock routing resources are taken in order to connect with the RAMs. For example:
  - The design uses an EFLX-4K 4x4 array with all logic cores.
  - 112 dual-port, 18-bit wide, 4K-deep RAMs are connected to both north and south sides of each EFLX core.
  - The number of cores is driven by RAM connections.

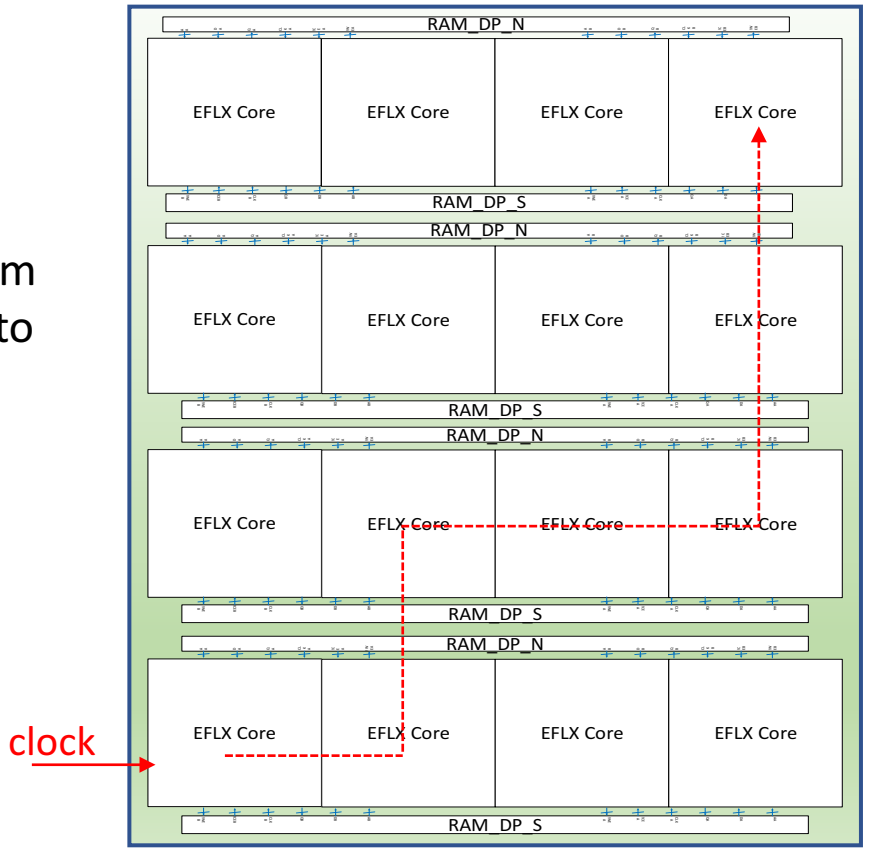






Guidelines for EFLX Clock Connections – Cont.

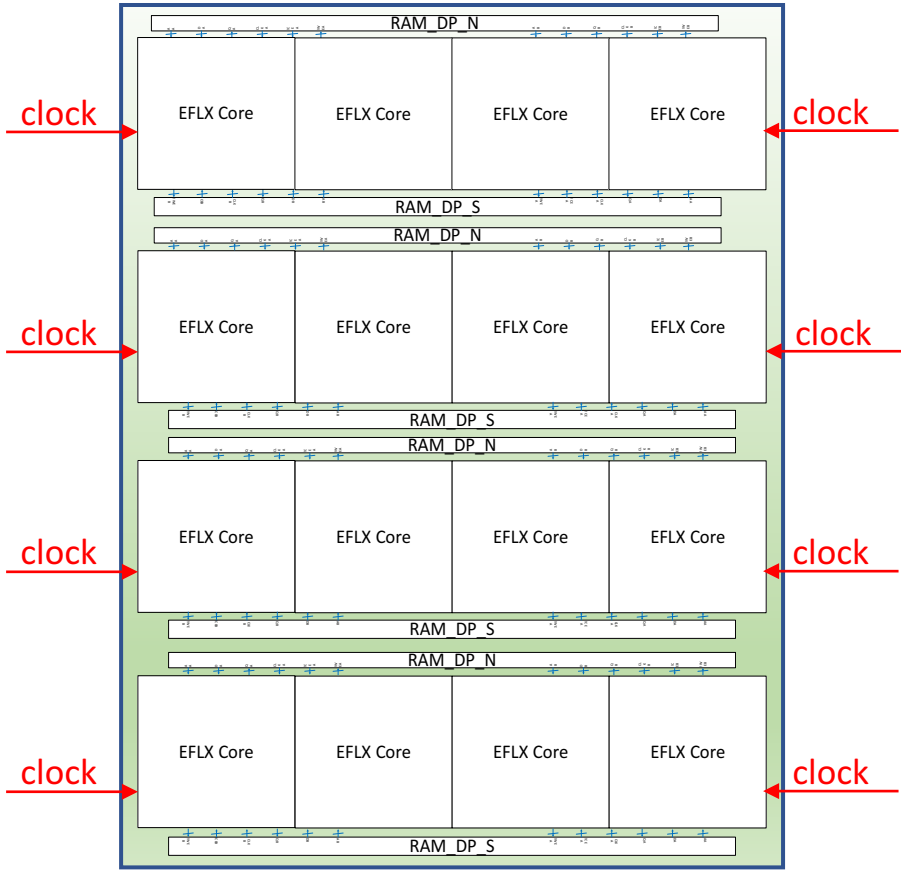
- ✓ The P&R is likely to fail due to insufficient clock routing resource. Because:
  - By default, the EFLX Compiler will connect the clock signal from a single clock pin at the lower left corner of the array and try to route and balance the clock tree.
  - The compiler won't be able to route the clock vertically when the array is big (in this case a 4x4 array), and when the North and South sides of EFLX cores are connected with external RAMs.



Guidelines for EFLX Clock Connections – Cont.

Solution:

- ✓ Create an IO Specification (.io file) to connect the clock signal from both West and East sides of the array to help with the clock routing and balancing.



Guidelines for Clock Connection – Cont.

Save file *PNR_IO.log* in the EFLX Compiler run directory as *<FILE_NAME>.io* and make the follow changes.

✓ Search for default clock signal connection

```
EFLX_CLK chip_tile_x=0, chip_tile_y=0, clk_side=W
   Input=0, pin=clk_c
```

✓ Connect the clock signal from both West and East sides of the EFLX array

```
EFLX_CLK chip_tile_x=0, chip_tile_y=0, clk_side=W
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=0, chip_tile_y=1, clk_side=W
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=0, chip_tile_y=2, clk_side=W
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=0, chip_tile_y=3, clk_side=W
   Input=0, pin=clk_c
```

```
EFLX_CLK chip_tile_x=3, chip_tile_y=0, clk_side=E
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=3, chip_tile_y=1, clk_side=E
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=3, chip_tile_y=2, clk_side=E
   Input=0, pin=clk_c
EFLX_CLK chip_tile_x=3, chip_tile_y=3, clk_side=E
   Input=0, pin=clk_c
```

✓ Rerun EFLX Compiler with the following command

```
./EFLX_COMPILER –edif <FILE_NAME.edf> –fp <FLOORPLAN_NAME.fp> -ram_def <RAM_DEF.ram_def> -io
<FILE_NAME>.io
```

Note: PNR_IO.log is a file automatically generated by the EFLX Compiler to show all the input and output ports connection with the EFLX array.