



Bitfiltrator

A general approach for reverse-engineering
Xilinx bitstream formats

Sahand Kashani, Mahyar Emami, James Larus



Reverse-engineering bitstreams

- FPGA bitstream
 - Binary configuration of FPGA
 - Loaded at startup
 - Vivado is only tool that can generate bitstreams for modern US/US+ FPGAs
- Bitstream generation is slow
 - Load P&R design checkpoint
 - Run DRC checks
 - Emit bitstream
 - 30 min for large designs
- Alternative: **Modify** an **existing** bitstream
 - Enables new types of applications
 - Configure overlays without runtime circuitry
 - Simulate faults in self-healing systems
 - Move simulator data to bitstream and continue execution in HW
 - Etc.
- Bitstream format undocumented
 - Requires reverse-engineering

Reverse-engineering bitstreams

- Discover device & architecture parameters (constants)
 - Goal is to locate configuration bits for named cells
- Prior work has done this
 - Focus on application of bitstream manipulation
 - Does not explain reverse-engineering **process**!
 - Targets only a specific device
 - Device modeling assumptions do not hold for other devices

→ Bitfiltrator

- **Automated** bitstream parameter extraction tool for Xilinx US/US+ FPGAs
- Basis for bitstream-manipulation tools / open-source FPGA toolchains

Outline

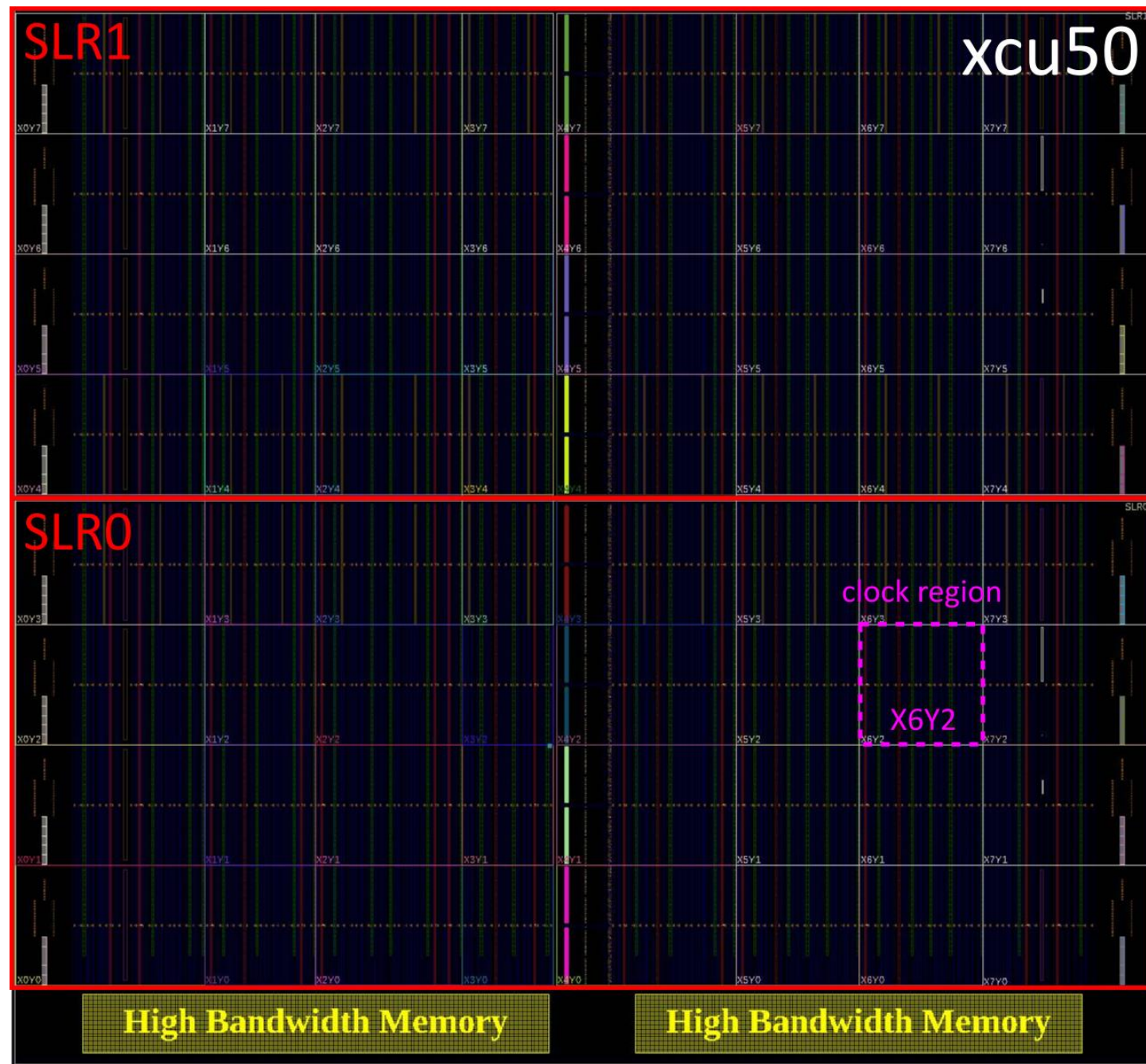
- Device structure
- Bitstream organization
- Finding device parameters
 - Frame address boundaries
 - CLB/BRAM major rows/columns
- Finding architecture parameters
 - CLB/BRAM minor columns
 - CLB/BRAM frame offsets
- Evaluation

More details in paper

- DSP major columns
- Device modeling pitfalls to avoid
 - SLR ordering in bitstream
 - Hidden clock regions
 - Empty frames after major rows
 - Etc.

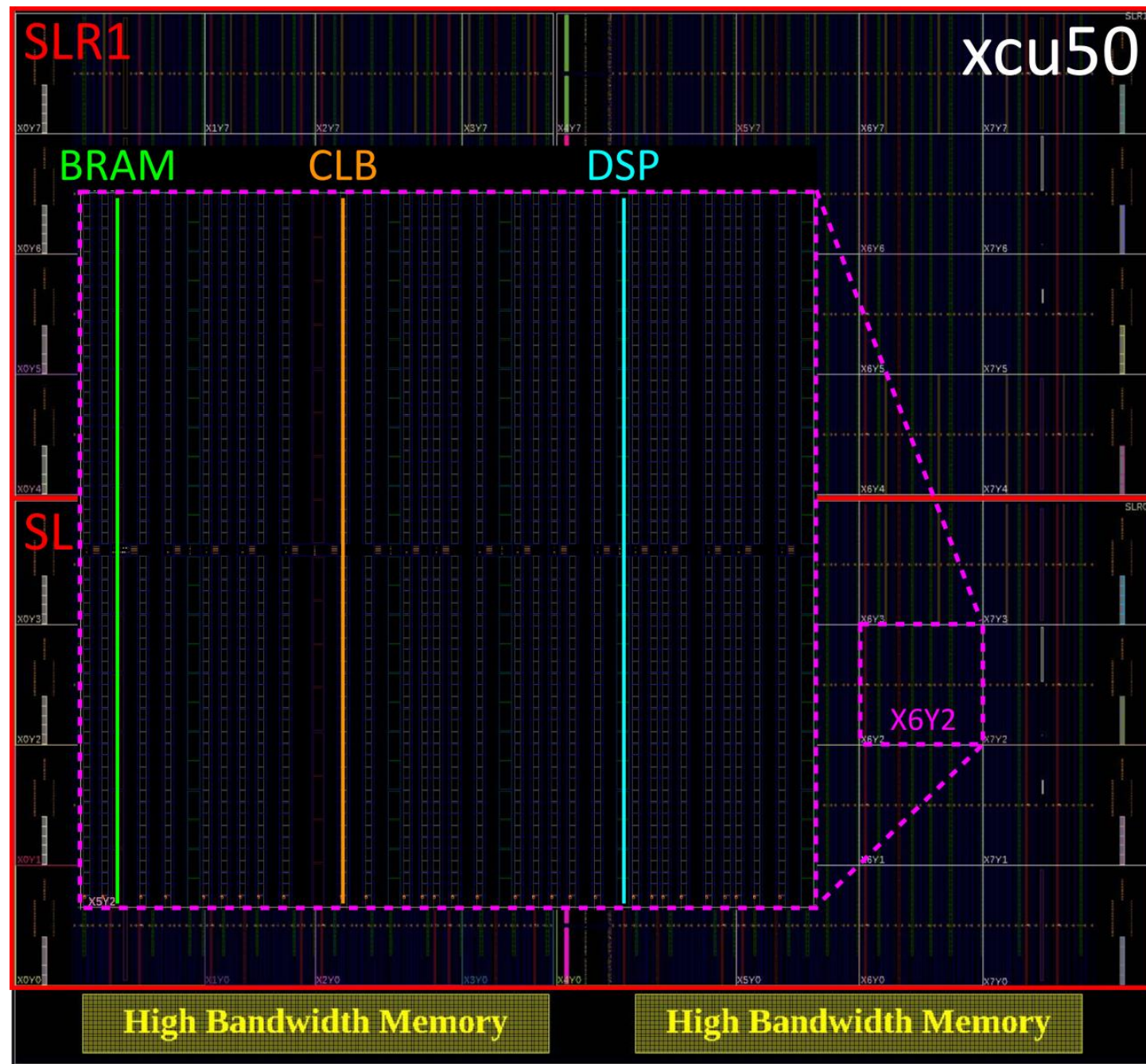
Device structure

- Multiple Super Logic Regions (SLR)
 - Grid of clock regions (e.g. X6Y2)



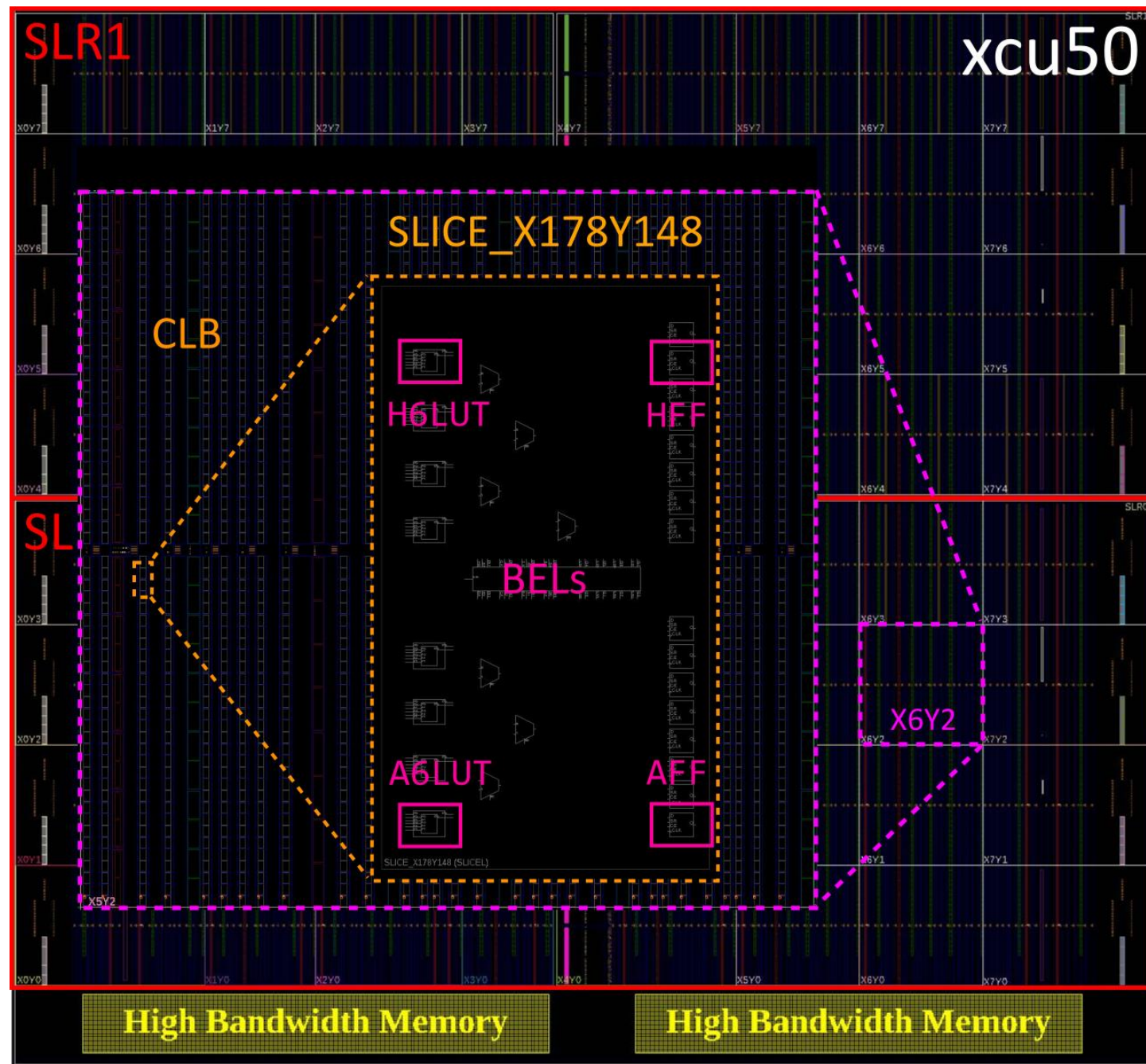
Device structure

- Multiple Super Logic Regions (SLR)
 - Grid of clock regions (e.g. X6Y2)
- Clock regions
 - Composed of homogeneous resource columns (CLB, BRAM, DSP, etc.)



Device structure

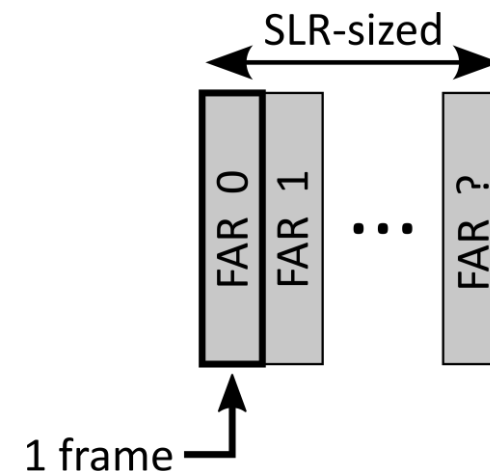
- Multiple Super Logic Regions (SLR)
 - Grid of clock regions (e.g. X6Y2)
- Clock regions
 - Composed of homogeneous resource columns (CLB, BRAM, DSP, etc.)
 - Basic elements (BELs) have properties that affect bitstream initial configuration
 - CLB 6-LUT : `INIT[63:0]`
 - CLB Flip-Flop (FF) : `INIT[0:0]`
 - BRAM content : `INIT[16383:0]`
- Goal
 - Locate exact position in bitstream of `INIT` properties for CLBs and BRAMs



Bitstream organization

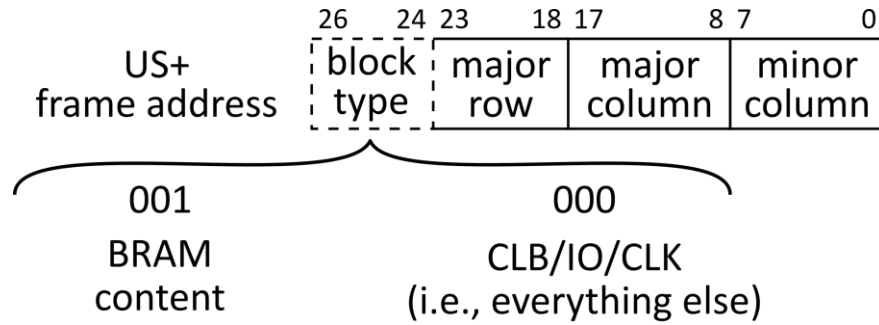
- Sequence of commands for FPGA configuration processor
- Each SLR configured independently
- SLR configuration
 - Sequence of **frames**
 - Frame: Smallest configurable unit in Xilinx FPGAs
 - One element (CLB, BRAM, etc.) horizontally [1]
 - One clock region vertically [1]
 - Every frame has an SLR-local address
- Bitstream configuration commands (single SLR, high-level)
 - Write 0 to Frame Address Register (FAR)
 - Write **SLR-sized** command with **all frames** to Frame Data Input Register (FDRI)

→ Large flat 1D array

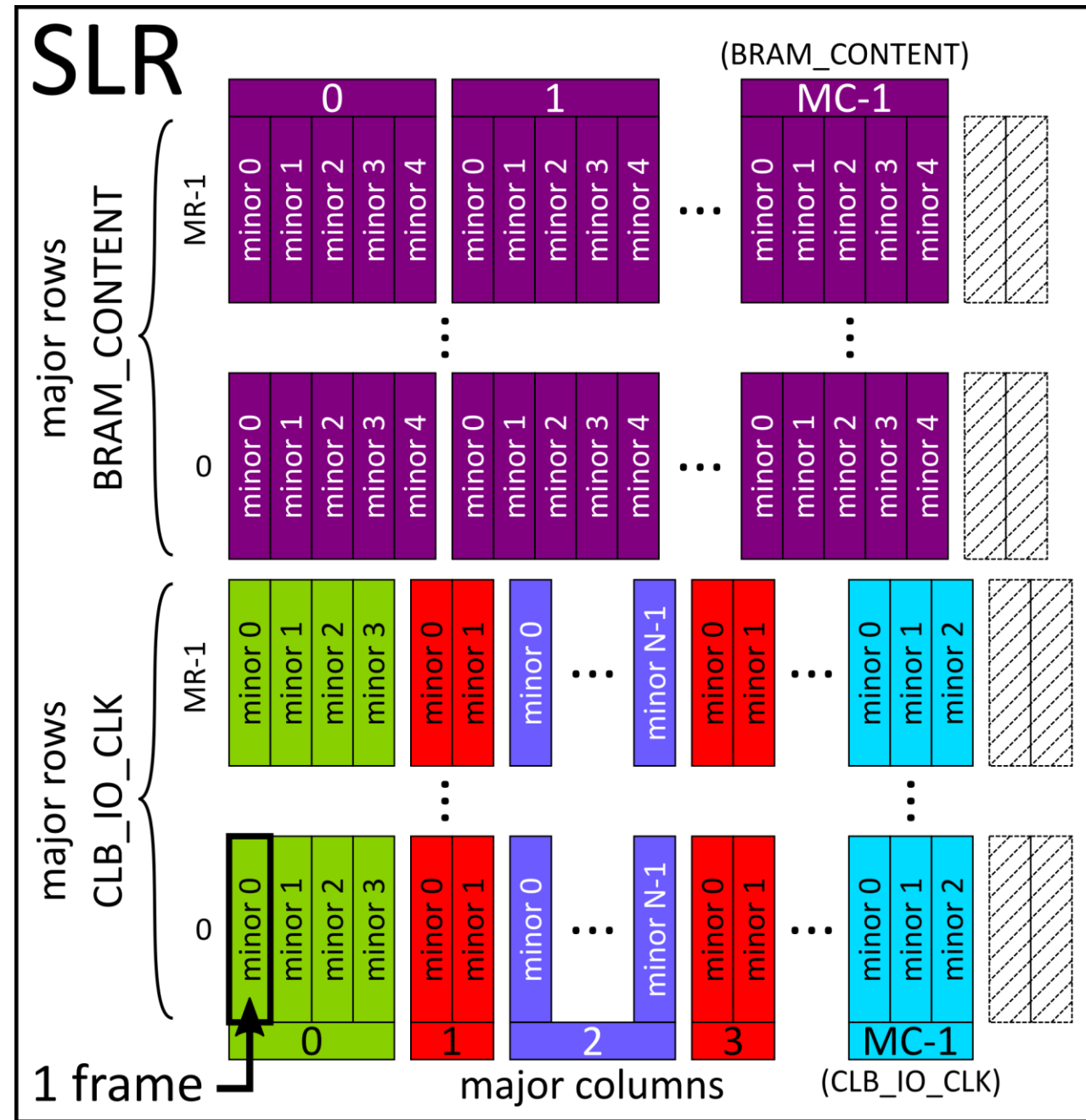


[1] UG570: UltraScale Architecture Configuration User Guide, 2022

Frame ordering



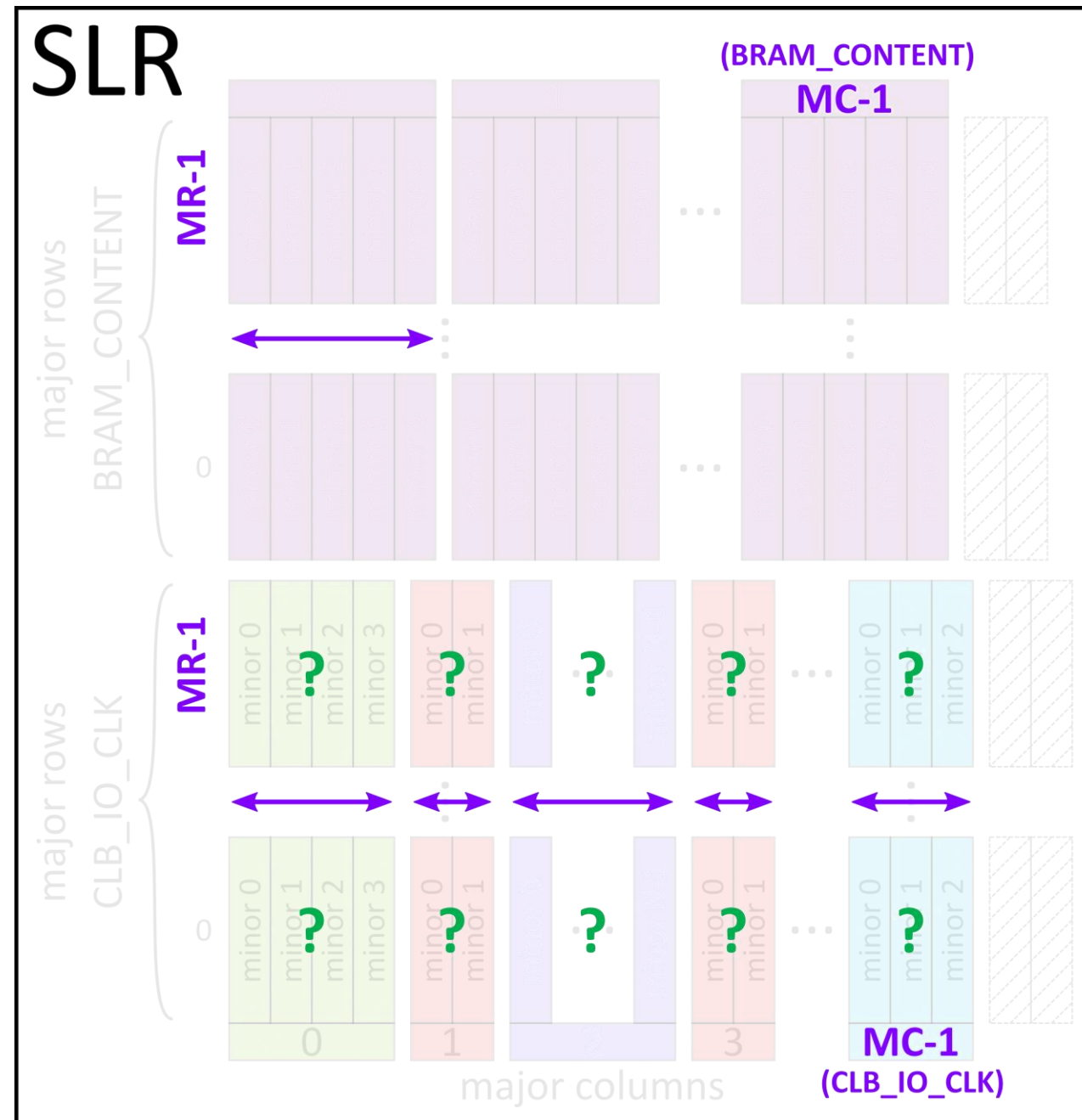
- Every frame spans one clock region vertically
 - ➔ Major row must be Y-offset of clock region in which a BEL is located
- Columns contain homogeneous resources
 - ➔ Any two major columns that contain the same resource must have the same number of minor columns
- BRAM content block type (001) > CLB/IO/CLK (000)
 - ➔ Frames containing BRAM initial values come after all other frames
- Number of major columns for BRAM content must match number of BRAM columns in a row.



Device parameters

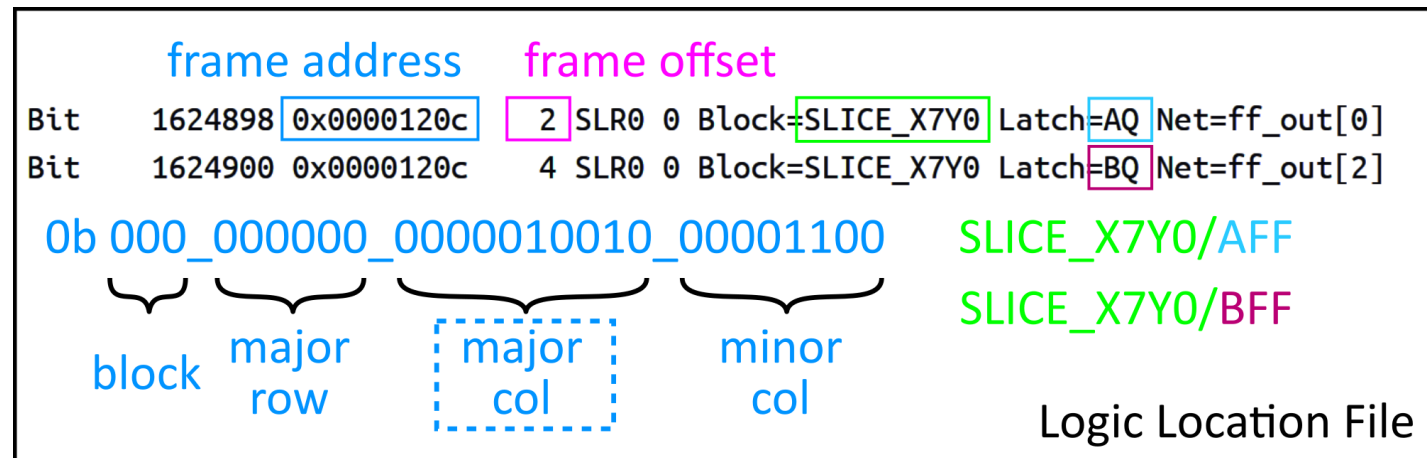
block type	major row	major column	minor column
------------	-----------	--------------	--------------

- Enumerate valid frames
 - Number of major rows
 - Number of major columns in each major row
 - Number of minors in each major column
- Placement experiments
 - Major column numbers of CLBs/BRAMs

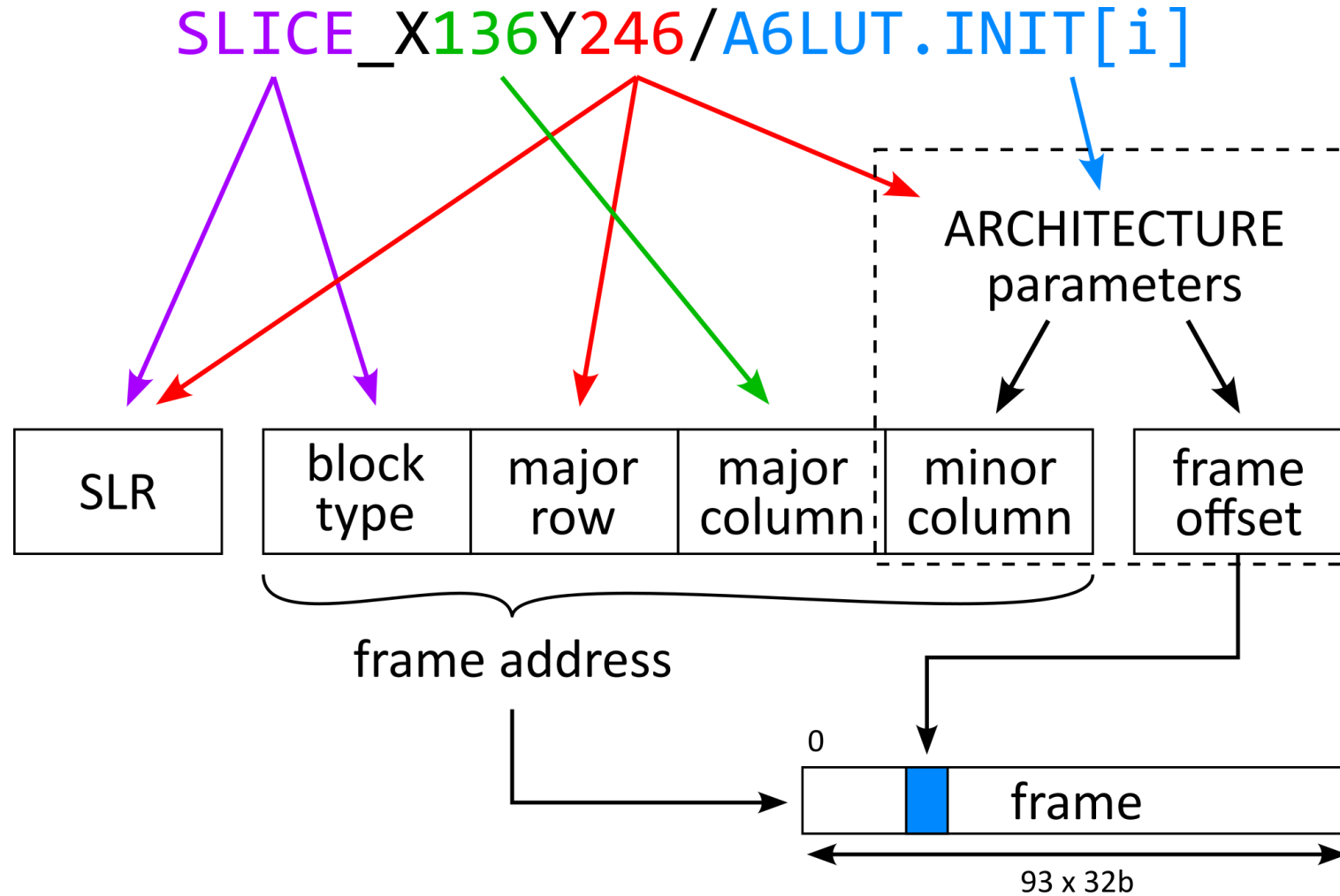


Identifying CLB/BRAM major column numbers

- Vivado can provide frame addresses and offsets of **user-state** bits
 - BRAMs, FFs, LUTRAMs (not standard LUTs)
 - Available in **logic location file** when generating bitstream
- For each clock region
 - Place one FF in every CLB column
 - Place one BRAM in every BRAM column
 - Generate bitstream and logic location file
 - **Parse logic location file** to extract major columns of CLBs/BRAMs



Architecture parameters



Identifying CLB/BRAM minors and frame offsets

- BRAMs, Flip-Flops, LUTRAMs (user-state bits)
 - Populate one **full column** with BRAMs, Flip-Flops, or LUTRAMs
 - Generate bitstream and logic location file
 - **Parse logic location file** to extract minor and frame offsets
- Standard LUTs (not user-state bits)
 - Hardware is regular → Frame encoding of any two LUTs should be identical
 - Populate all 8 LUTs in **one CLB**
 - Try all $8 \times 64 = 512$ **one-hot INIT** configuration bits
 - Generate bitstream
 - **Compare bitstreams against empty bitstream**
 - Minors and frame offsets that differ are LUT **INIT** bits
 - **Translate offsets** to other CLBs by locating anchor bit

Evaluation

- Implementation
 - **Python**: Bitstream parsing and analysis
 - **Tcl**: Replicating and configuring BELs in Vivado
- Dataset
 - **All** US/US+ FPGAs available in free WebPack version of Vivado 2021.1 (34 devices)

Architecture	Vivado Name	Count
UltraScale	Kintex UltraScale	2 / 12
	Virtex UltraScale	0 / 7
UltraScale+	Kintex UltraScale+	3 / 10
	Virtex UltraScale+	8 / 31
	Zynq UltraScale+	21 / 38
	Zynq UltraScale+ RFSOC	0 / 16

- Validation
 - Create design that populates all LUTs, FFs, and BRAMs in a device
 - Initialize every resource with a **random INIT** value
 - Generate & parse bitstream
 - Use **Bitfiltrator** to extract device and architecture parameters
 - Reconstruct all **INIT** values → **Match**
- Reverse-engineered parameters extracted by **Bitfiltrator** are correct
- Frame ordering scheme is correct

Summary

- **Bitfiltrator**: **Automated** bitstream parameter extraction tool for Xilinx US/US+ FPGAs

- Basis for bitstream-manipulation tools or open source FPGA toolchains
- Bitstream reverse-engineering **process** explained in detail
- Evaluated on 34 US/US+ devices
- Open source
github.com/epfl-vlsc/bitfiltrator



- Bitstream parameters extracted
 - SLR ordering in bitstream
 - Major rows (CLBs, BRAMs, DSPs)
 - Major columns (CLBs, BRAMs, DSPs)
 - Num minors in each major column
 - Minor columns & frame offsets (**INIT** bits)
 - LUTs, LUTRAMs, Flip-Flops, BRAMs
 - Hidden clock regions
 - “Holes” in resource columns
 - Unexpected offsets in bitstream
 - Etc.

Backup slides

Example device database (xcu50)

```
"SLR0": {  
  "rowMajors": {  
    "0": {  
      "num_minors_per_bram_content_colMajor": [256,256,256,256,256,256,256,256,256,256,256,256,256],  
      "num_minors_per_std_colMajor"           : [ 8, 4, 76, 16, 16, 76, 8, 16, 76, 16, 16, 76, 16, 16, 76, ... ],  
      "clb_colMajors"                         : { "0": 3, "1": 4, "2": 7, "3": 9, "4": 10, "5": 12, "6": 13, "7": 15, ... },  
      "bram_content_colMajors"                : { "0": 0, "1": 1, "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, ... },  
      "bram_content_parity_colMajors"         : { "0": 0, "1": 1, "2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, ... },  
      "bram_reg_colMajors"                    : { "0": 19, "1": 32, "2": 65, "3": 106, "4": 115, "5": 169, "6": 175, "7": 213, ... },  
      "dsp_colMajors"                         : { "0": 6, "1": 37, "2": 55, "3": 61, "4": 70, "5": 84, "6": 102, "7": 111, ... },  
    },  
  },  
  ...  
}
```

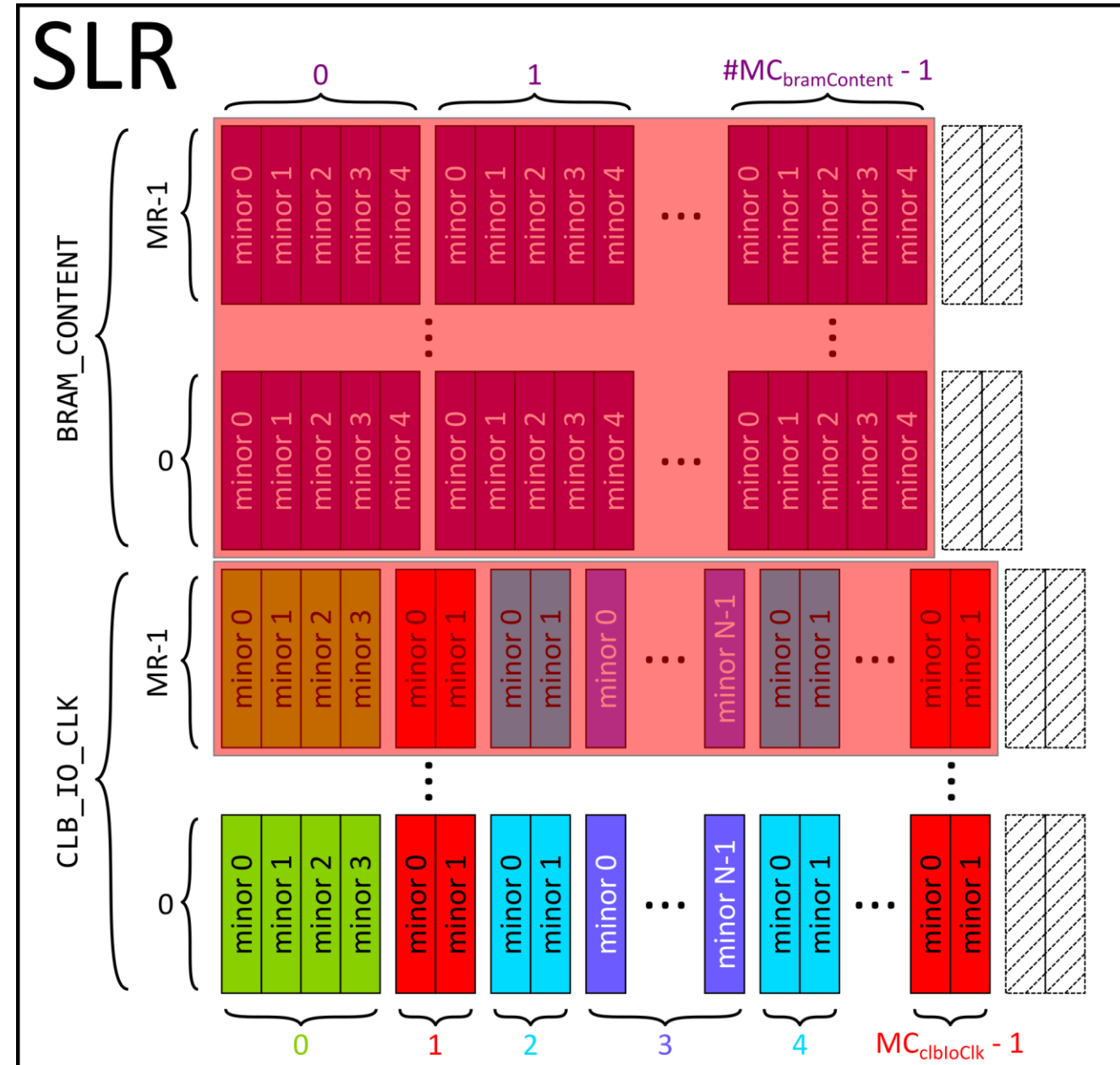
DSP48E2_X1Y0-17 → major column 37 (CLB_IO_CLK)

SLICE_X2Y0-59 → major column 7 (CLB_IO_CLK)

RAMB18_X3Y0-23 → major column 3 (BRAM_CONTENT)

Locating a BEL

- E.g. **SLICE_X178Y12**
 - Block type
 - **SLICE** → CLB/IO/CLK
 - Major row
 - **Y12** → 60 CLBs per row → row 0
 - Major column
 - **X178**
 - How to locate in row 0?
- Do not know the number of minors in each major column
- Do not know which major columns are CLBs/DSPs/BRAMs



Finding major/minor column boundaries

Problem

- Frames **bulk**-loaded in bitstream with base frame address 0
 - Cfg processor auto-increments frame address **implicitly** (no external feedback)
 - Frame address field boundaries unknown
- Cannot associate frames with their address when parsing bitstream

Solution

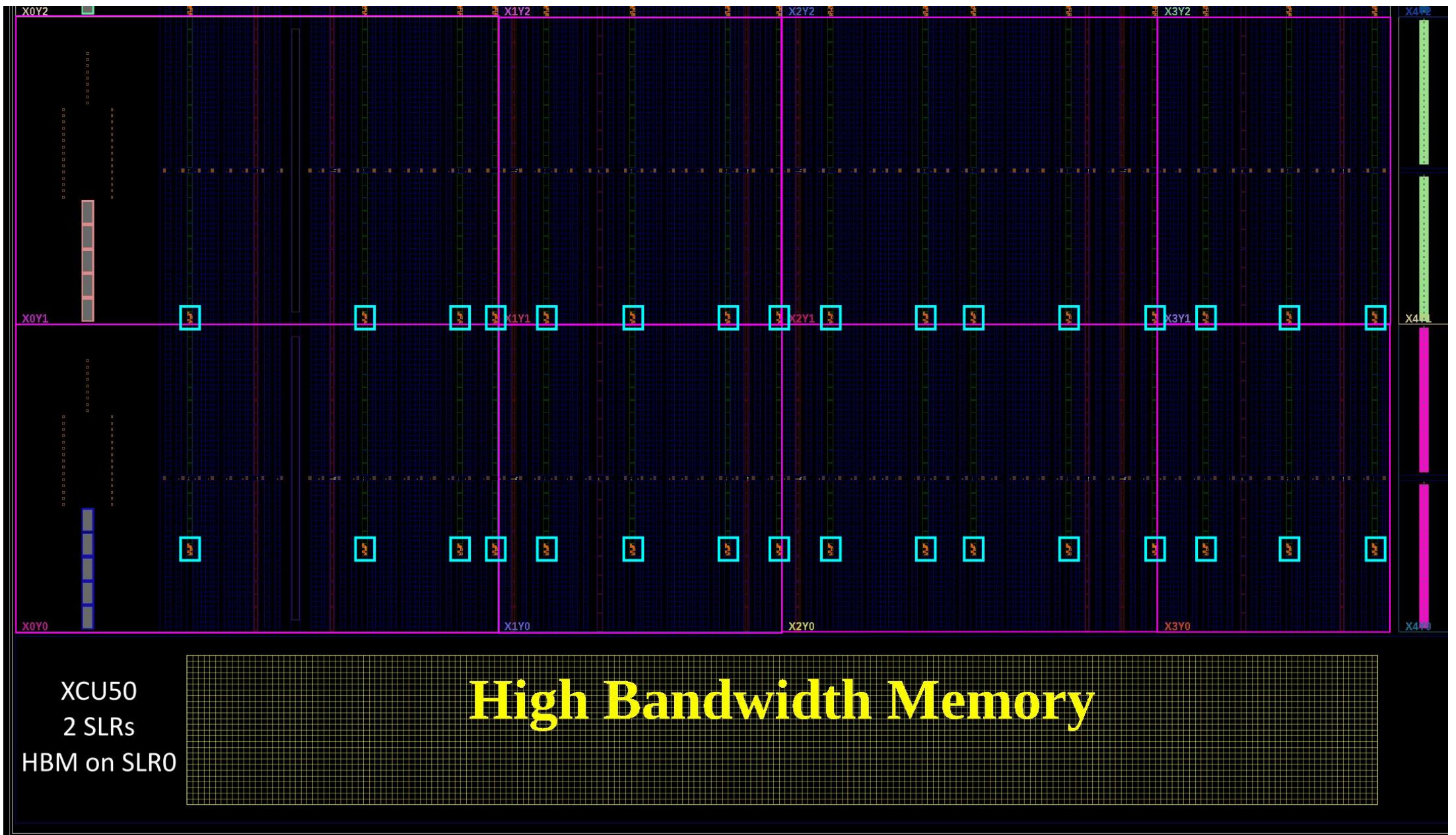
- Generate bitstream such that frames are **individually**-loaded
 - Bitstream with per-frame CRCs
 - Bitstream **explicitly** increments frame address register with a command after each frame
 - Extract all addresses written to the frame address register in bitstream
- Reconstruct frame address boundaries

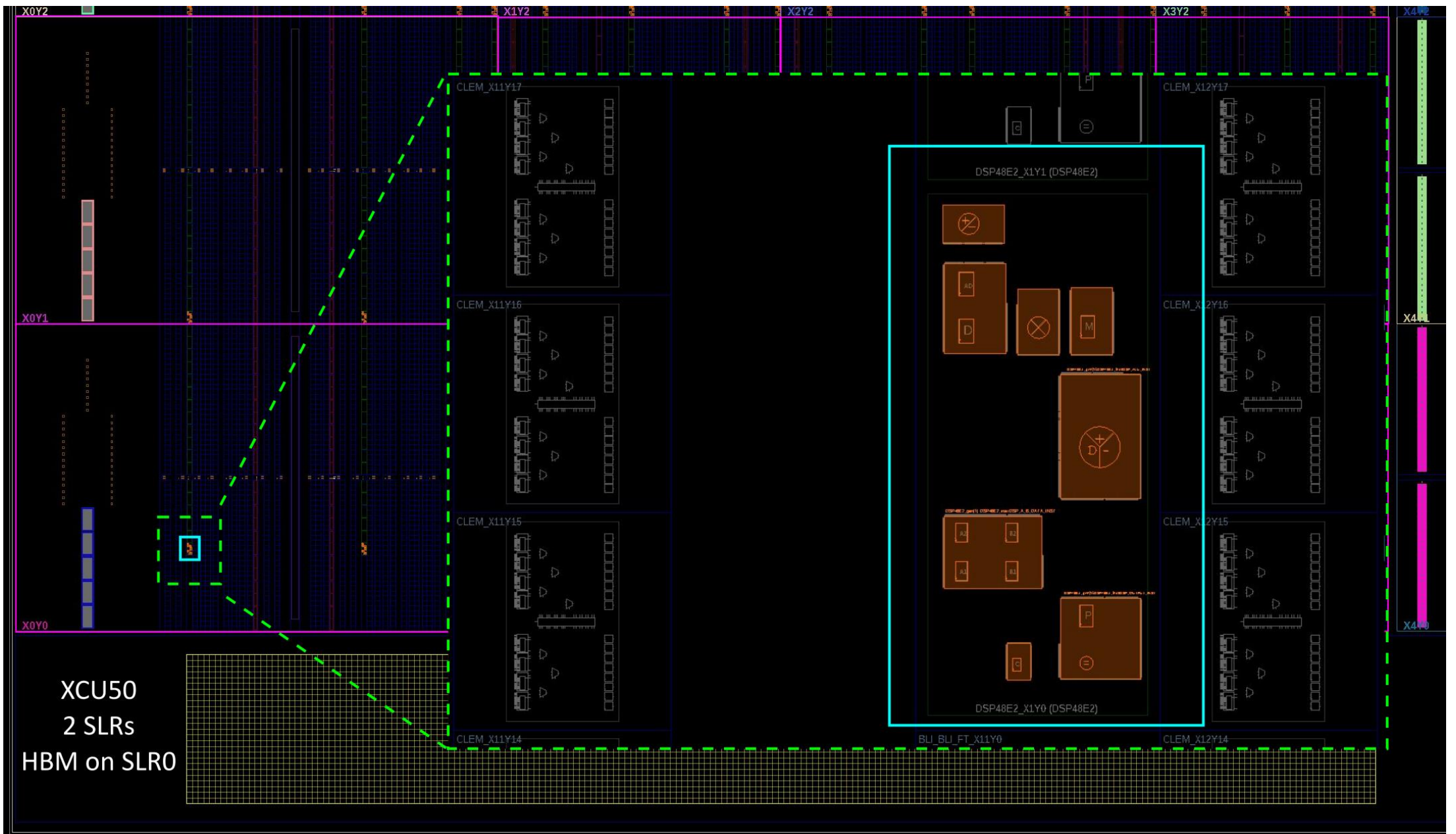
Finding number of minors in each major column

- Generate a bitstream with per-frame CRCs
 - Ensures every frame is loaded individually
 - Address of every frame explicitly written to frame address register in bitstream
- Parse bitstream
 - Extract all writes to the frame address register
 - Compute number of minors in each major column

```
"SLR0": {  
  "rowMajors": {  
    "0": {  
      "num_minors_per_bram_content_colMajor": [256,256,256,256,256,256,256,256,256,256,256,256,256],  
      "num_minors_per_std_colMajor"           : [ 8, 4, 76, 16, 16, 76, 8, 16, 76, 16, 16, 76, 16, 16, 76, ... ],  
    },  
    ...  
  }  
}
```

CLB/IO/CLK major column 0 → 8 minor columns

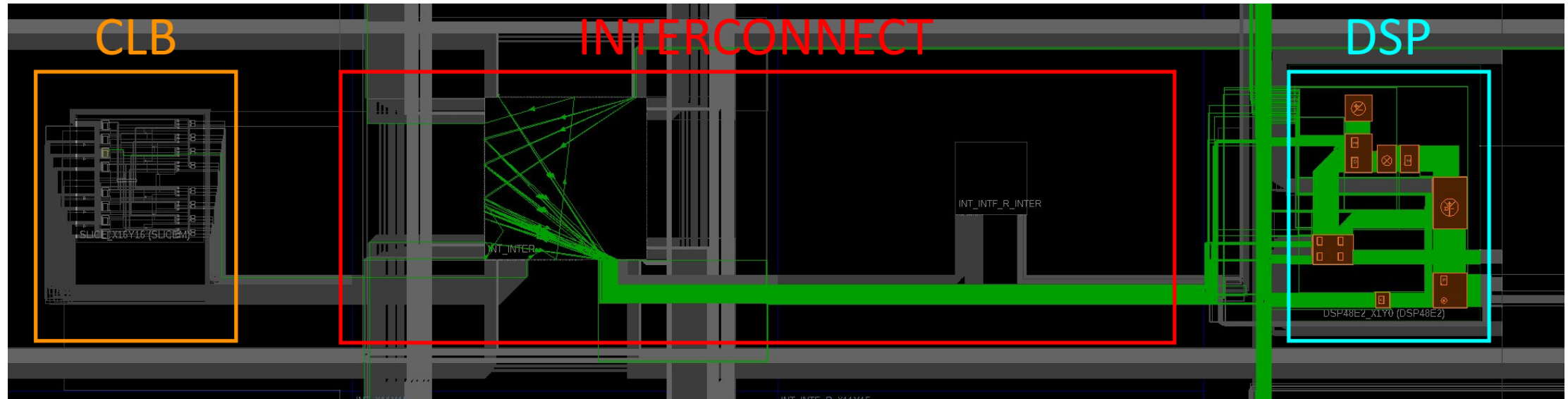




Identifying DSP major column numbers

- DSPs do not contain user-state bits
 - Need another mechanism to locate DSP major columns in the bitstream
- Idea
 - Place one DSP in every DSP column
 - Generate bitstream
 - Compare bitstream against empty bitstream
 - Major column of frames that differ should be DSP columns

Identifying DSP major column numbers



- DSP inputs cannot be floating
 - Driven by constants (from CLBs)
 - Routed by interconnect
 - Compare bitstream against empty one
 - Non-DSP frames also differ!
 - Filter out CLB columns (now known)
 - Filter out interconnect columns
 - Intuition: Interconnect columns need more minors to configure than DSP columns
- Easy to locate and filter out