XILINX
ALL PROGRAMMABLE™

Automatic time sharing for area reduction

Henri Fraisse

Contributors

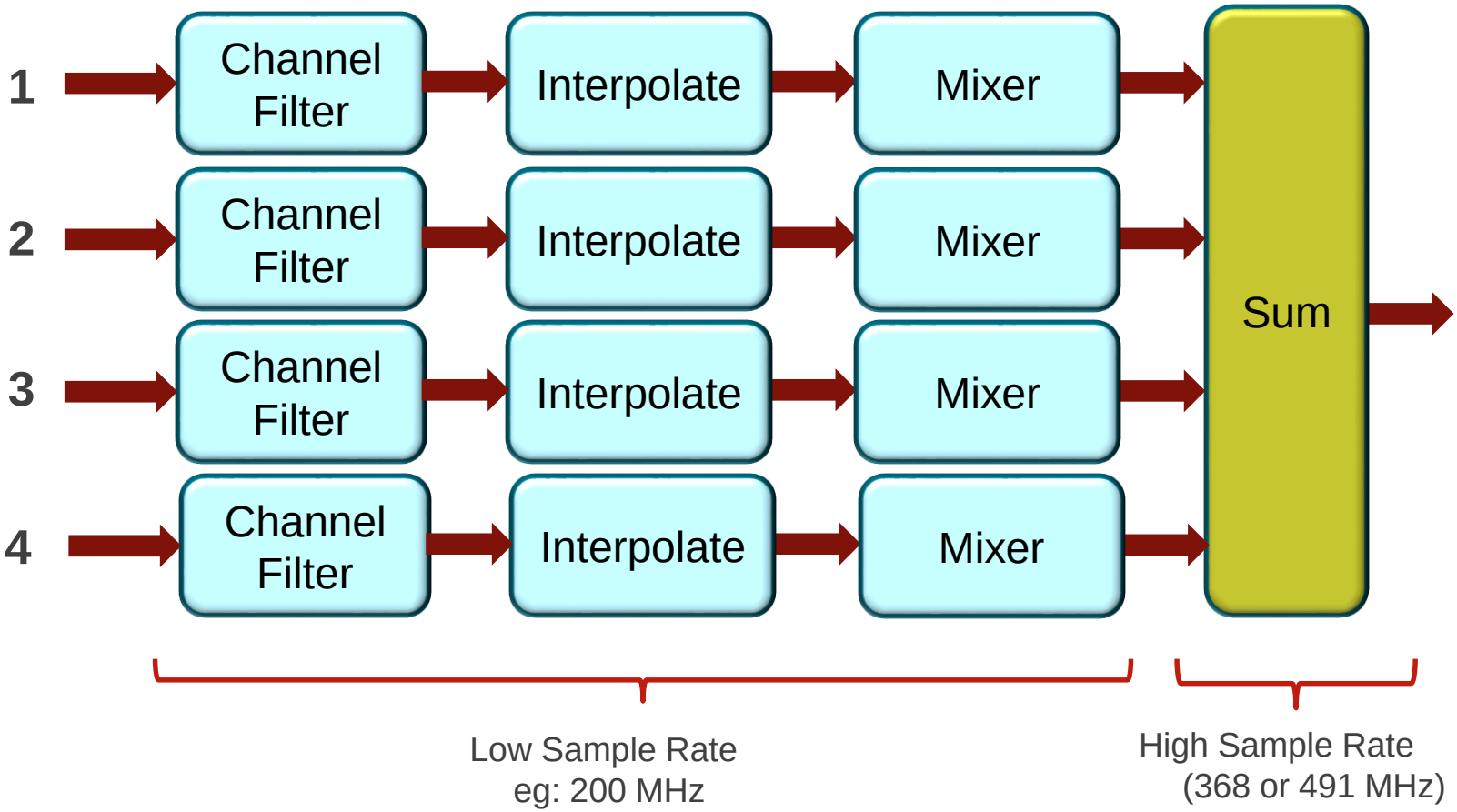
- Architecture group: Ilya Ganusov, Henri Fraisse
- Synthesis group: Ashish Sirasao, Elliott Delaye, Bing Tian
- CTO Labs: Alireza Kaviani

XILINX
ALL PROGRAMMABLE™

Agenda

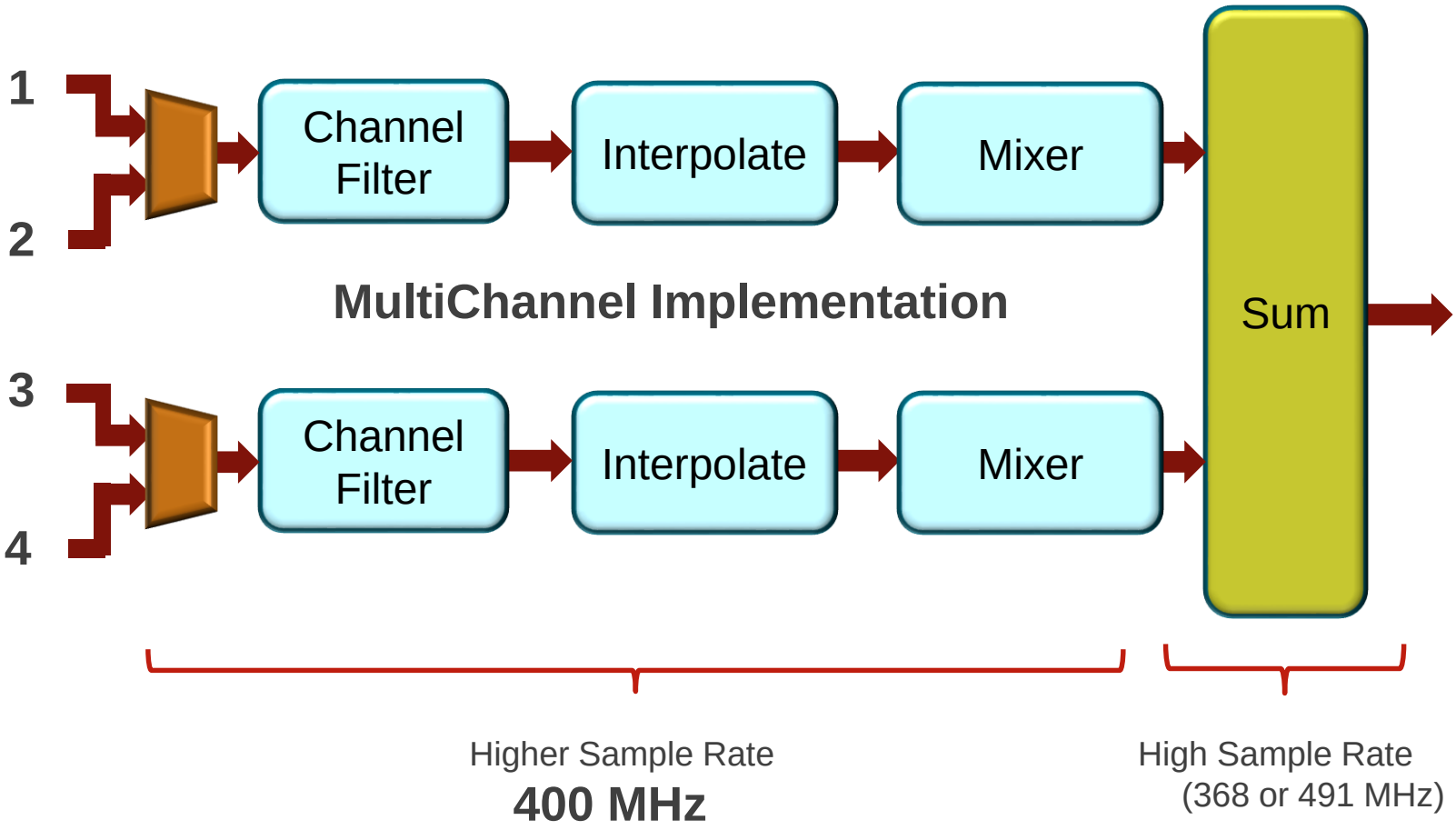
- Introduction
- Design transformations
- Flow changes
- Experimental data
- Conclusion

Making the best use of the FPGA



… but DSP48E can run at 500MHz

Making the best use of the FPGA



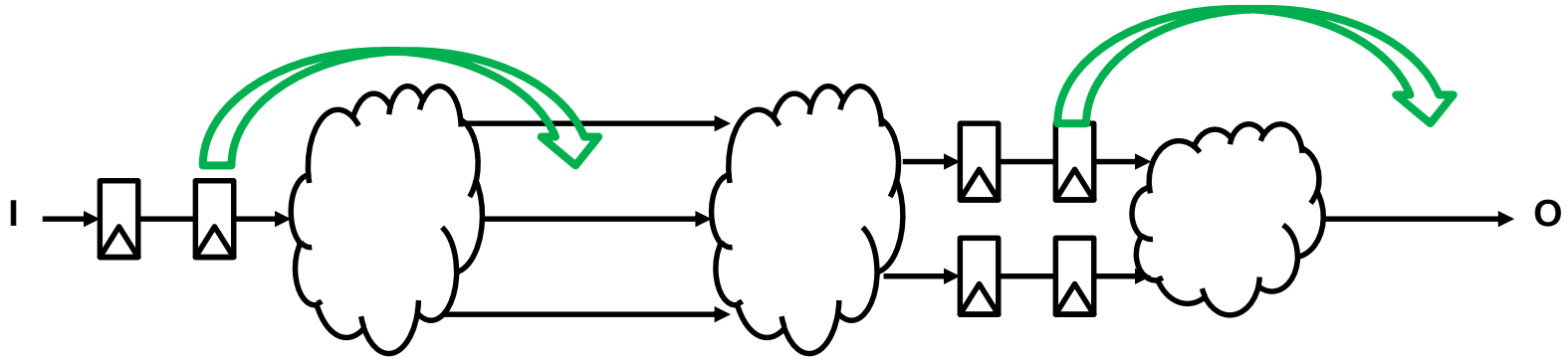
Requires a lot of manual design work

Solution: Automatic optimization in Vivado

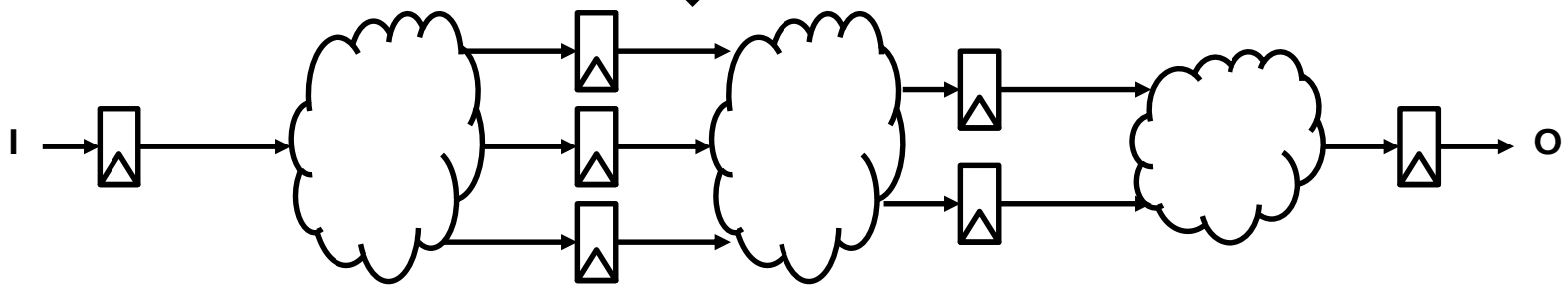
Basic principles

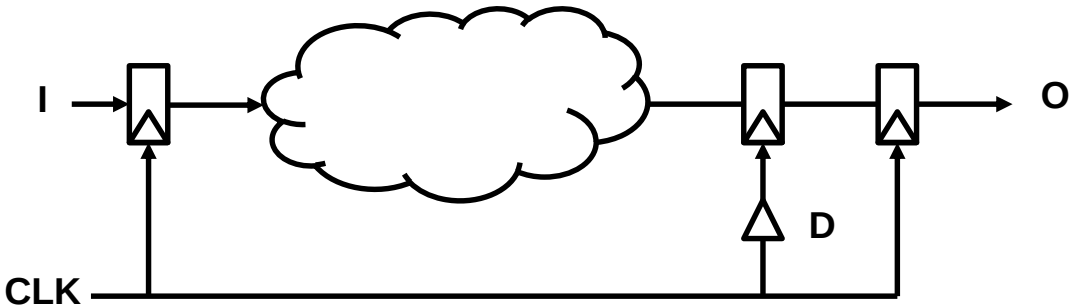
- Share the combinational part (LUT, CARRY) of duplicated instances
  - Can dramatically improve LUT, CARRY and DSP utilization
  - Do not change significantly FF or BRAM utilization
- Perform multiple operations during one clock cycle
  - Need to operate at a higher clock frequency (typically 2X)
  - Looks very hard to achieve the same QOR but:
    - Each path contains twice as many register levels
    - Retiming and time borrowing should balance delay between register levels

Timing optimizations



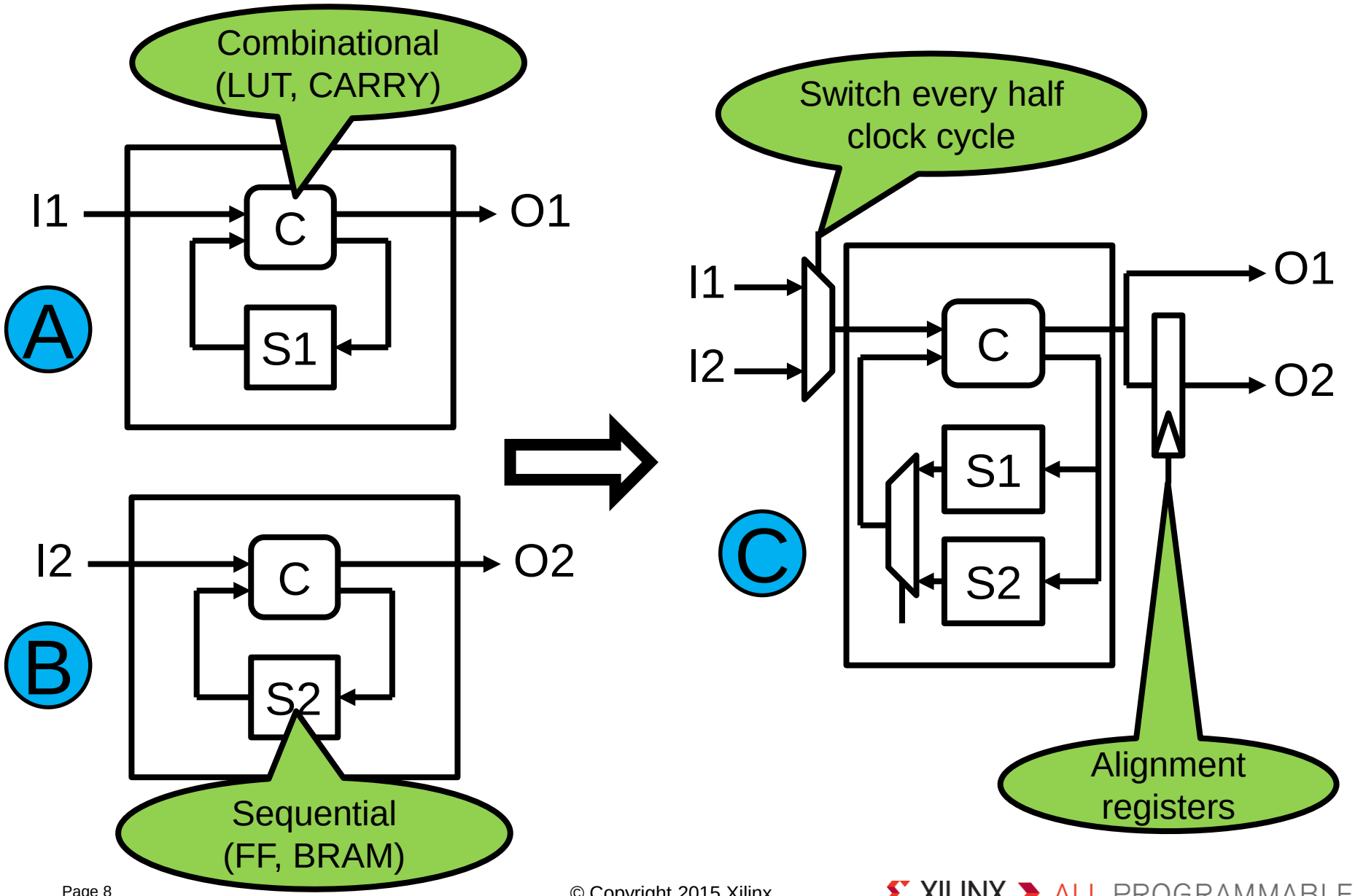
Retiming: spread registers evenly





Time borrowing: delay the arrival time of clock signals on some registers

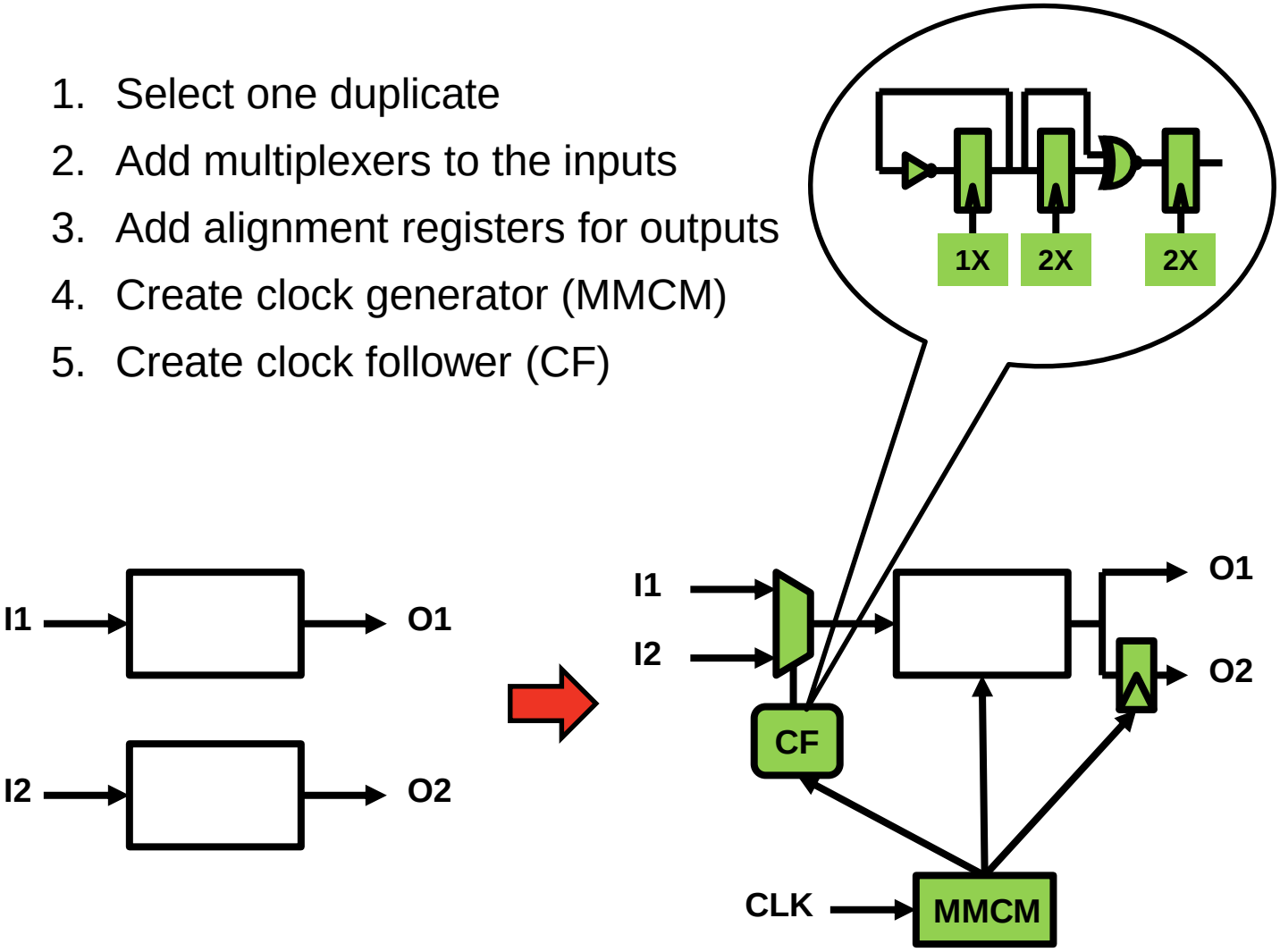
Overview of design transformations



Design transformations

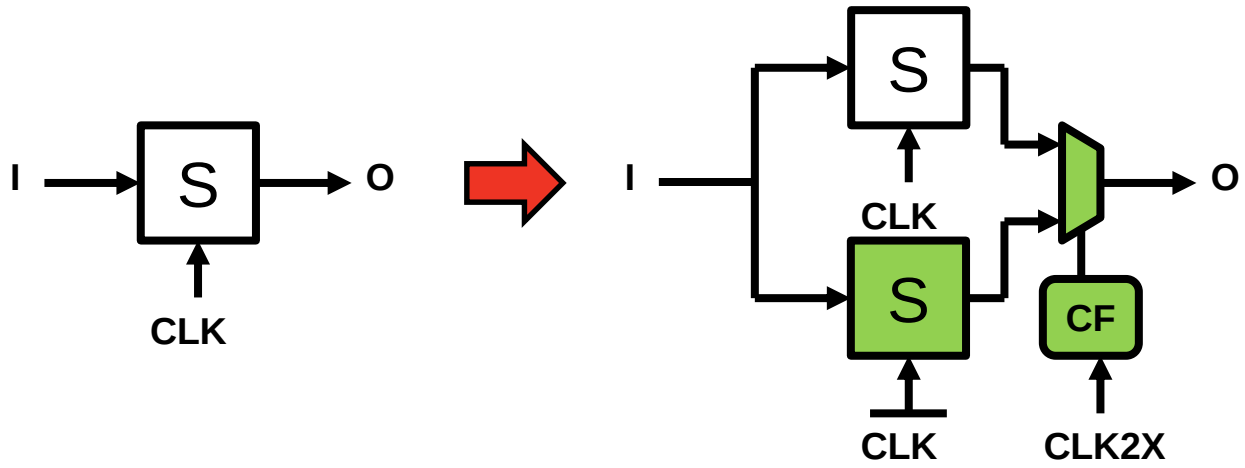
External module transformations

1. Select one duplicate
2. Add multiplexers to the inputs
3. Add alignment registers for outputs
4. Create clock generator (MMCM)
5. Create clock follower (CF)

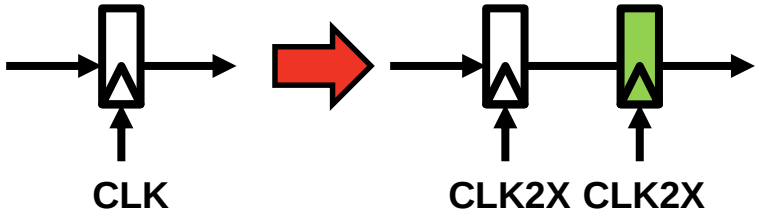


Internal module transformations

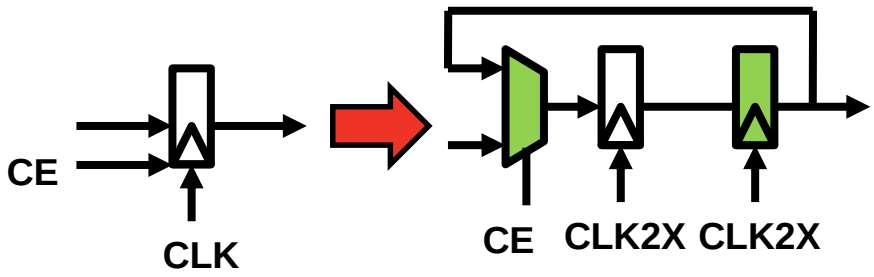
For LUTRAM, BLOCKRAM, SRL, or any basic sequential module:
duplicate, add multiplexer on output, invert one clock



Better solution for register (no clock enable)



register with clock enable

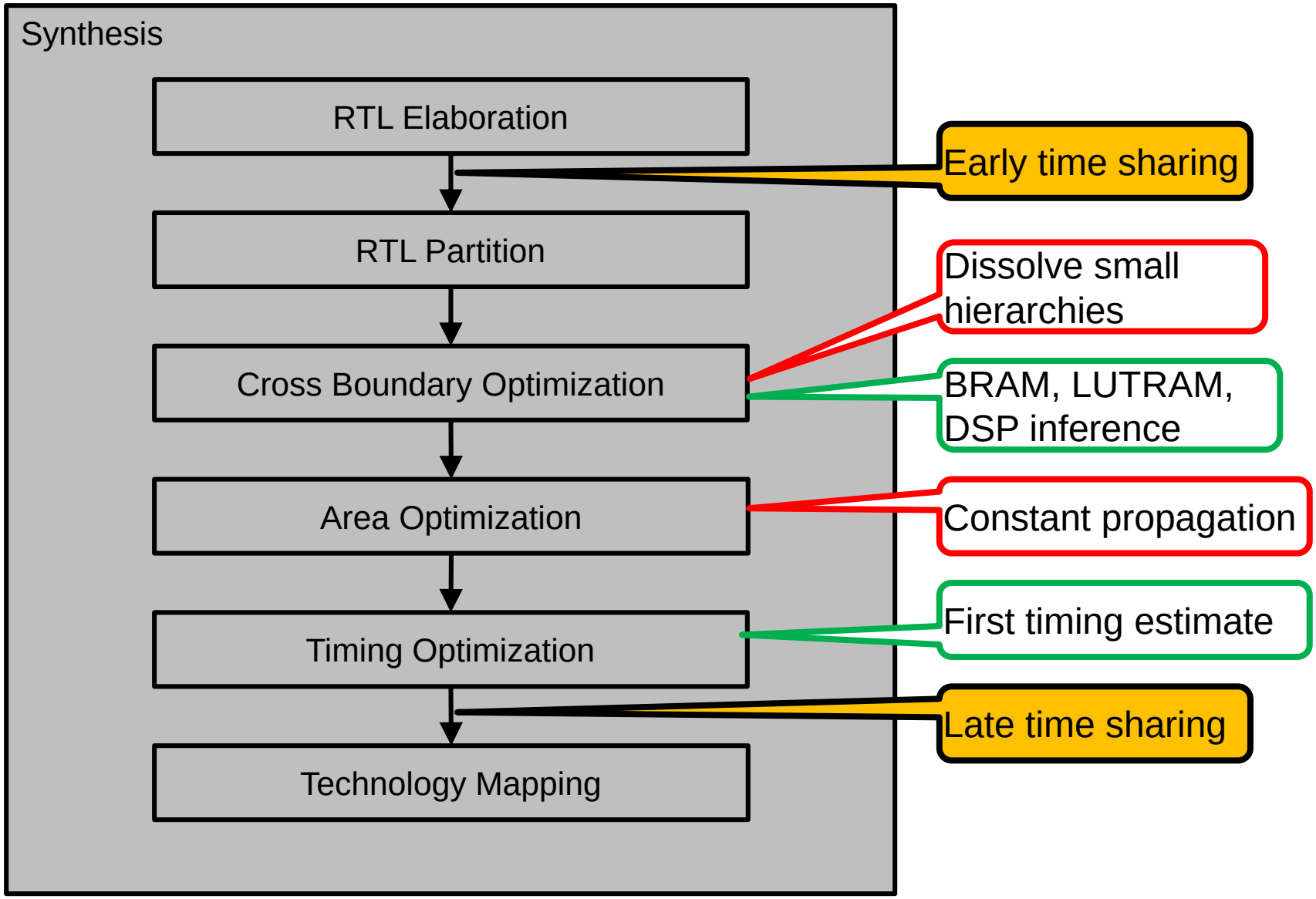


Related academic work

- Resource sharing is a classic HLS optimization
  - Only applied for operations scheduled on separate cycles
  - Only applied for single functional units (ex: DSP)
- A closer related academic work is Multi-pumping
  - Typically applied as HLS optimization
  - Only applied for single functional units (ex: DSP)
- C-slow retiming (Berkeley)
  - Replace each register with a sequence of N registers (typically 2) and apply retiming
  - Alter the behavior of the design
  - Improves throughput but adds latency
  - Do not improve area (rather increase area by adding registers)

Flow changes

Flow options for time sharing



How to take advantage of both options?

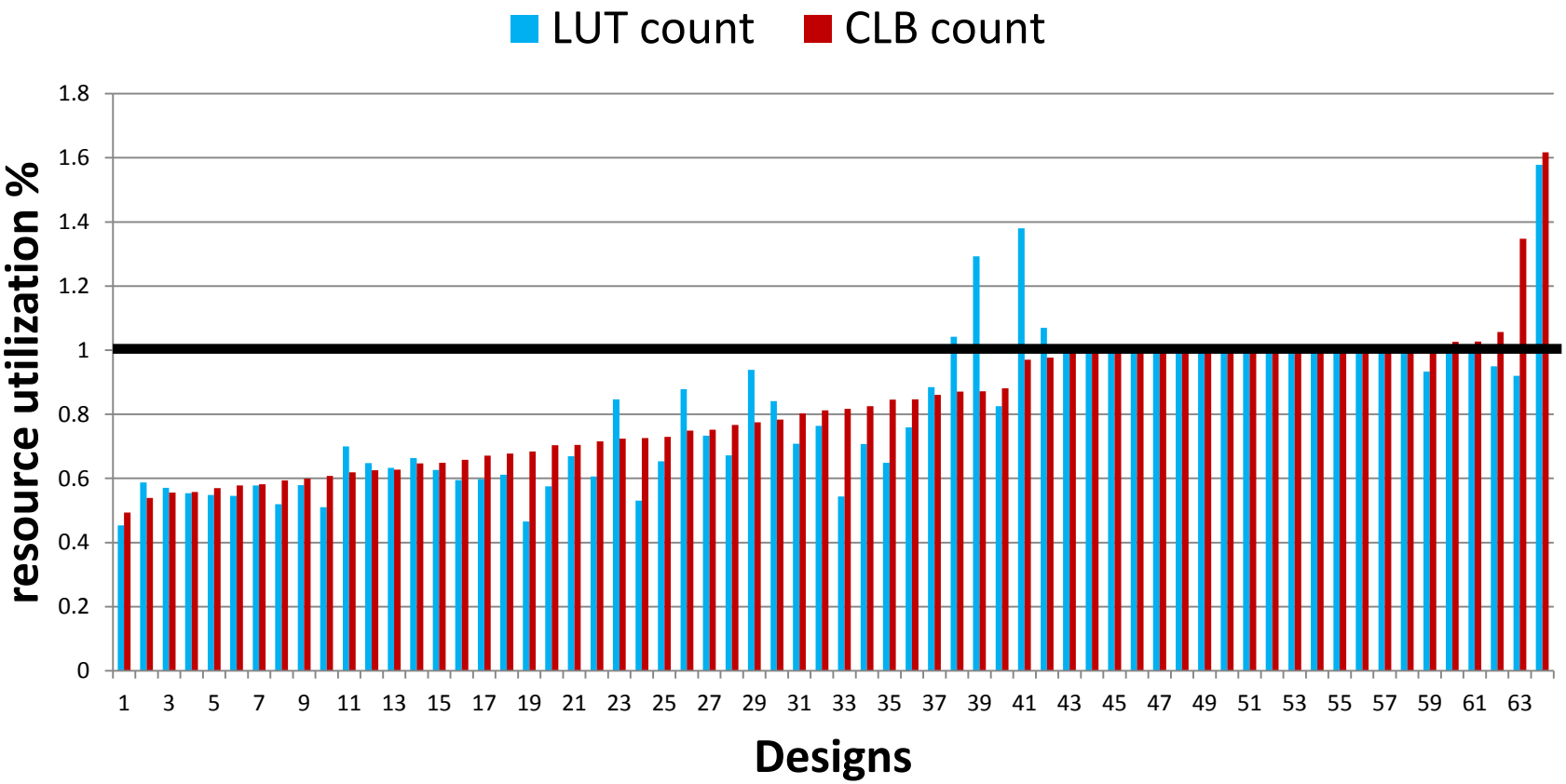
- Step 1: Annotate the candidates for time sharing early
  - Prevent optimizations crossing the boundaries of annotated instances
- Step 2: Apply time sharing later on annotated instances
  - Filter timing critical instances
- Step 3: Re-apply cross boundary optimizations
  - Focus specifically on annotated instances

Experimental results

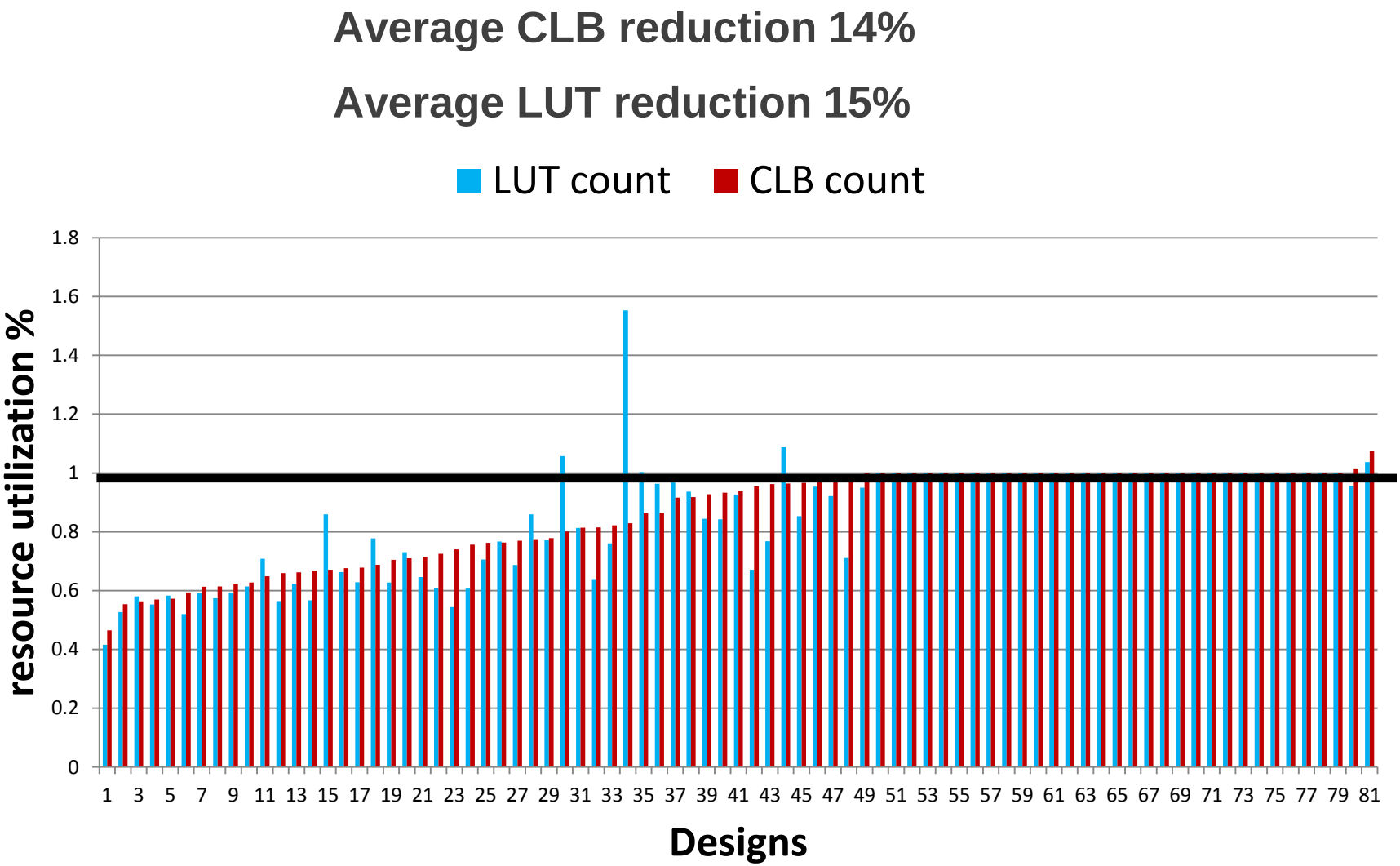
Results for the early time sharing flow

Average CLB reduction 17%

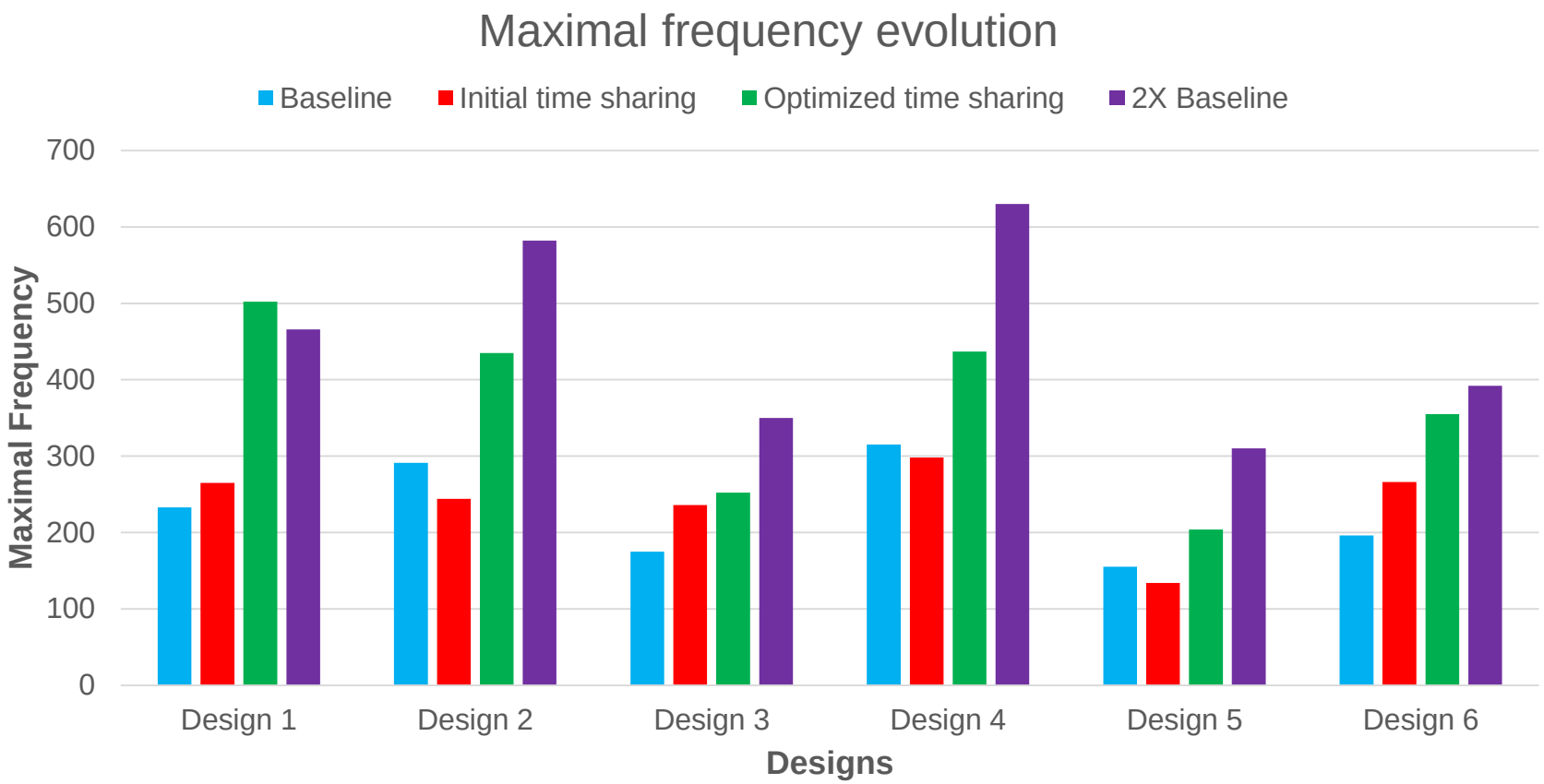
Average LUT reduction 19%



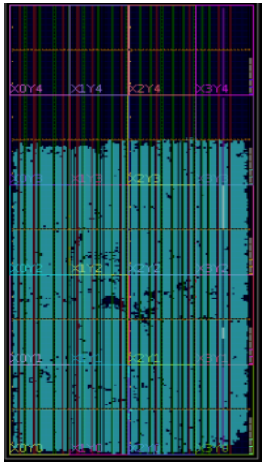
Results for the late time sharing flow



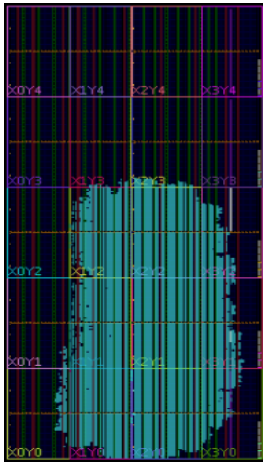
Impact on Maximal Frequency



Design 1

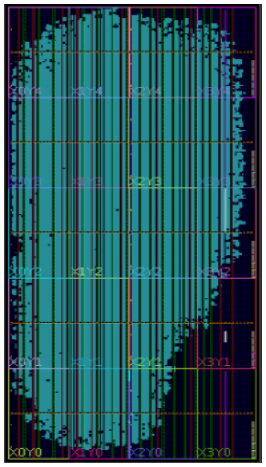


Baseline

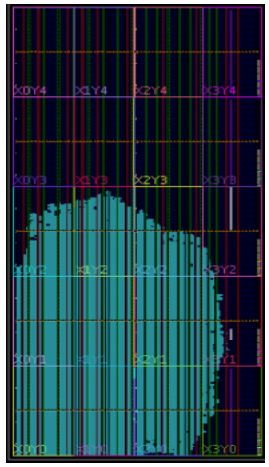


Time sharing

Design 2

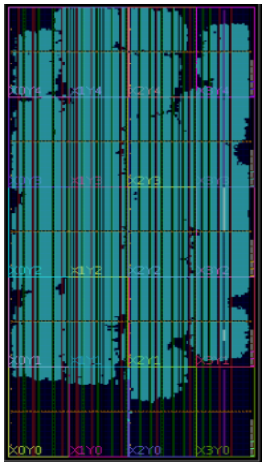


Baseline

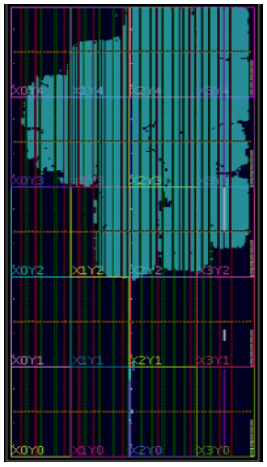


Time sharing

Design 3

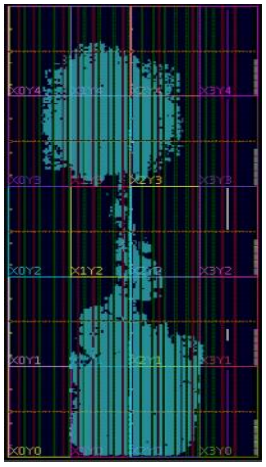


Baseline

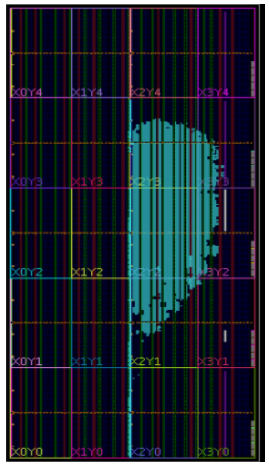


Time sharing

Design 4



Baseline



Time sharing

Conclusion and future works

Concluding Remarks

- Implemented automatic time sharing in Vivado
- Demonstrated the potential for area reduction on a representative set of designs
- Investigated the QOR impact on few selected designs

Thank you!