The Stratix™ 10 Highly Pipelined FPGA Architecture

David Lewis, Gordon Chiu, Jeffrey Chromczak, David Galloway, Ben Gamsa, Valavan Manohararajah, Ian Milton, Tim Vanderhoek, John Van Dyken

Altera (now part of Intel Corp.)

Pipelining in the Routing: Previous Work & Assumptions

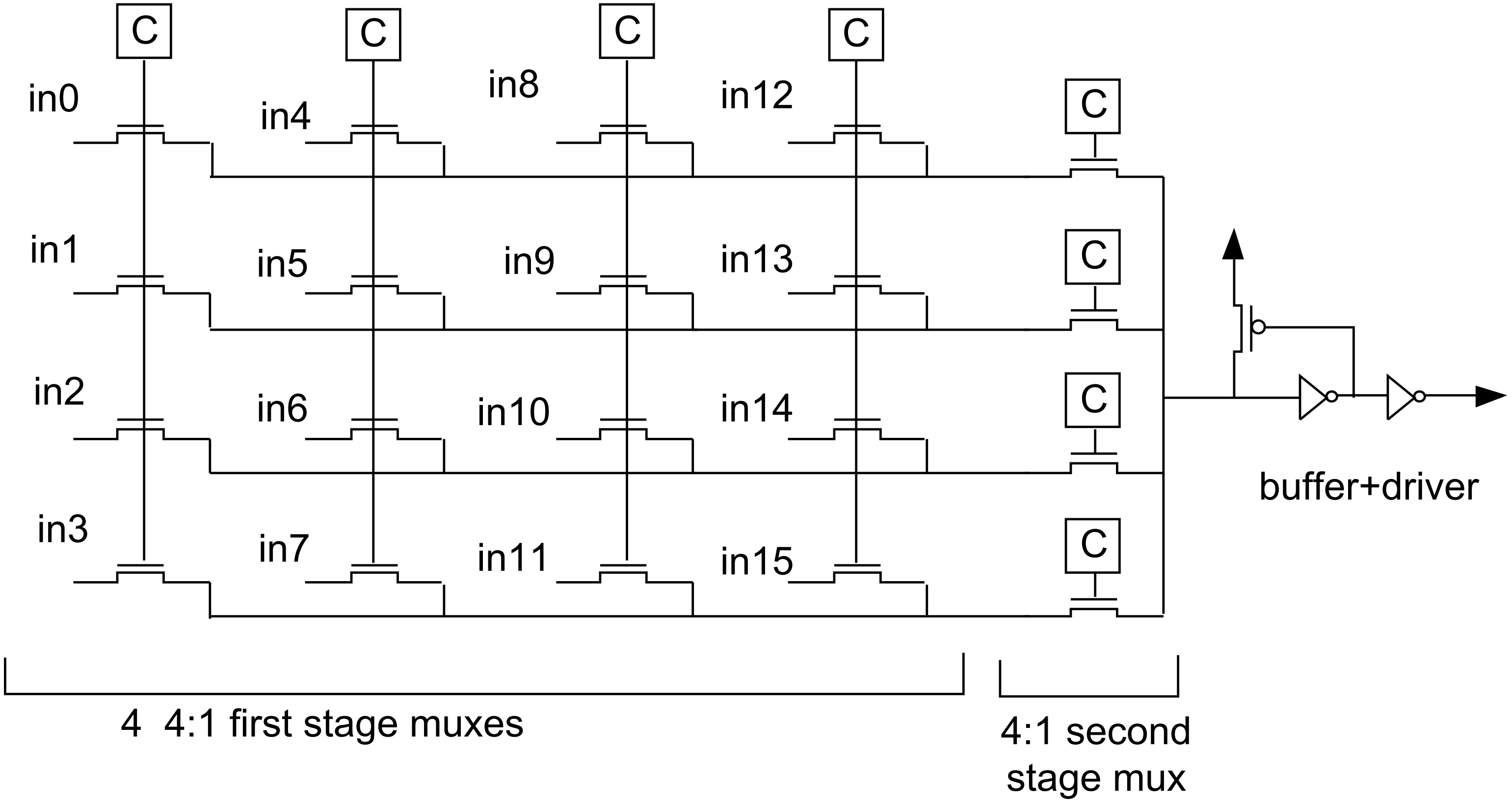
- An old idea:
  - Singh 2001; Eguro 2008
  - Older fine grained studies used pass transistor architectures
  - Reported costly and mediocre performance for registered routing;
  - Focused on retiming

- Other architectures targeted synchronous datapath designs
  - With higher level models

- Assumptions going forward:
  1. Designs are becoming more pipelined
  2. Can auto pipeline to add latency
  3. Designers are more willing to redesign to gain speed

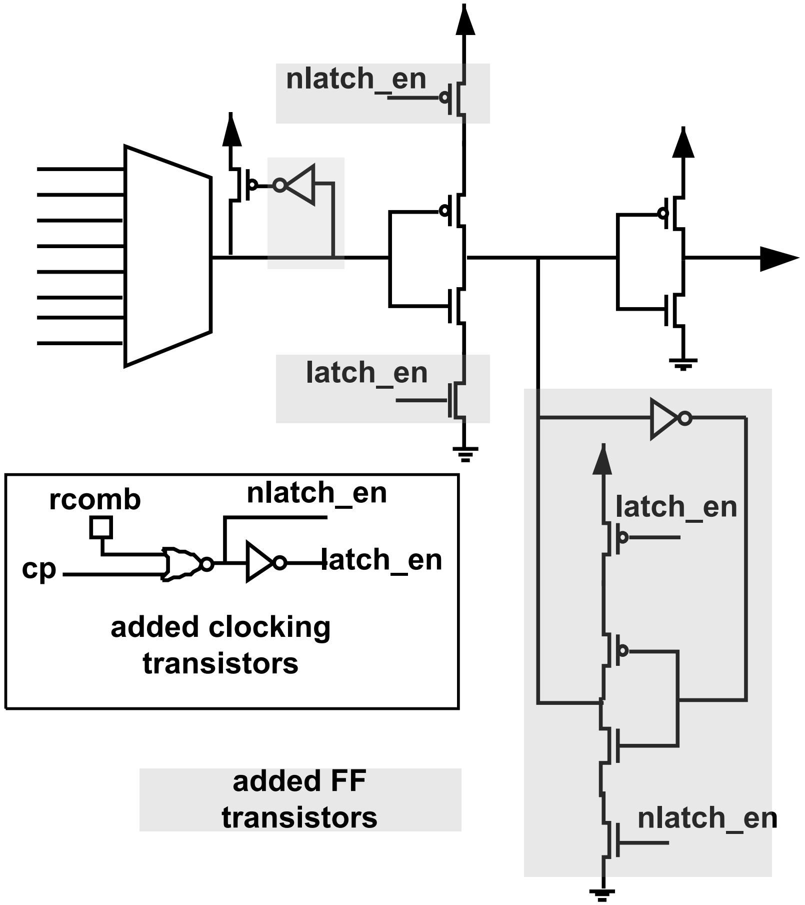
Typical Routing Multiplexers

- Have two stages
- Are followed by 1 or 2 buffers to drive the wire or LE input



Key Idea: Add Pipelining to Routing Mux

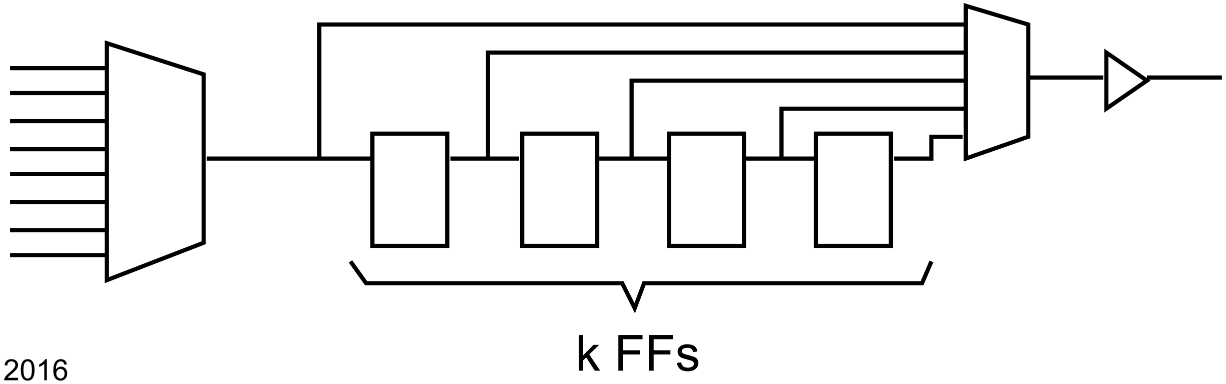
- Creates a pipelined direct drive routing fabric
- Uses internal pulse latch to buffer
  - 8 minimum width and 2 larger transistors
- Minimal area cost and delay
- Latch alone is <5% soft logic area for 100% routing drivers
- Clocking is key to cost



How Many Flip-Flops (k) Should Go After Mux?

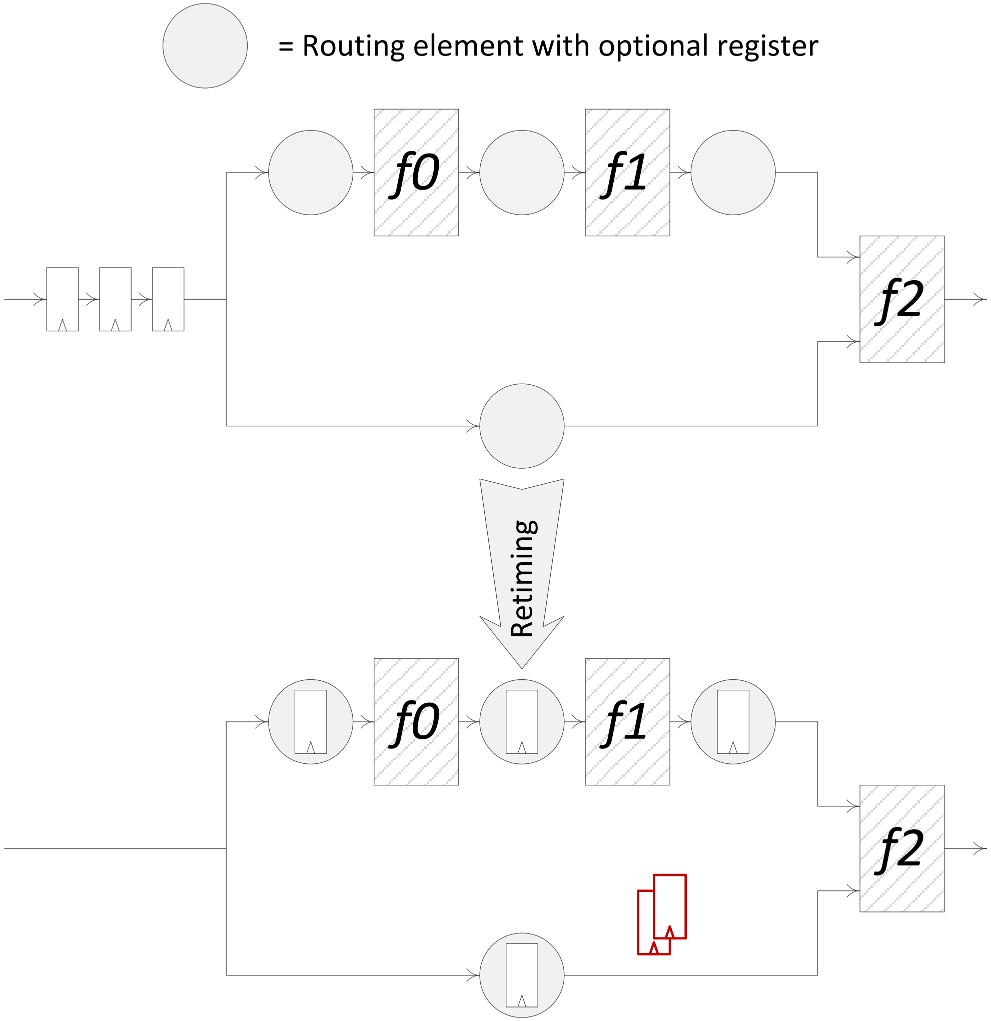
- At least one FF (*k*=1) in each routing mux
  - May need *k* > 1 in locations where multiple signals converge to balance latency mismatches
- Built CAD to selectively enables from 0 to *k* FF during retiming
- *k* = 1 is especially efficient implementation because can use pulse latch with flow through and no output mux
- *k* > 1 requires full edge triggered FF and bypass muxes for subsequent FFs: +10% area and increase delay per FF
- Experiments with larger *k* are useful to establish bounds on performance, but unlikely we would build *k* > 1

Architectural Model



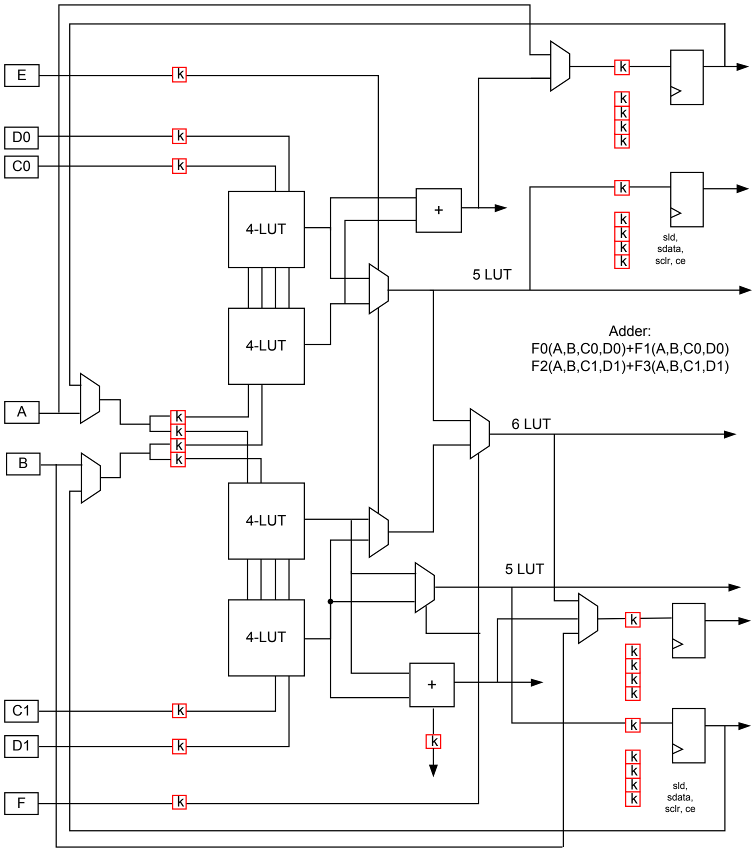
Why K > 1?

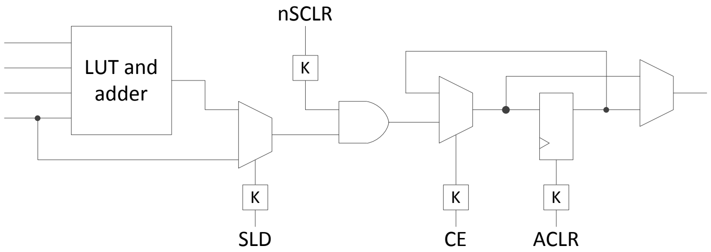
- Retiming reconvergent paths can often lead to latency balancing problems
- Can correct minor mismatches in latency by providing k > 1 wherever signals converge



Pipelined Logic Fabric

- Postulate *k* FFs in front of logic element
- Note shared AB inputs should have separate FFs to allow independent pipelining of 5-LUTs





Three Different Experimental Flows

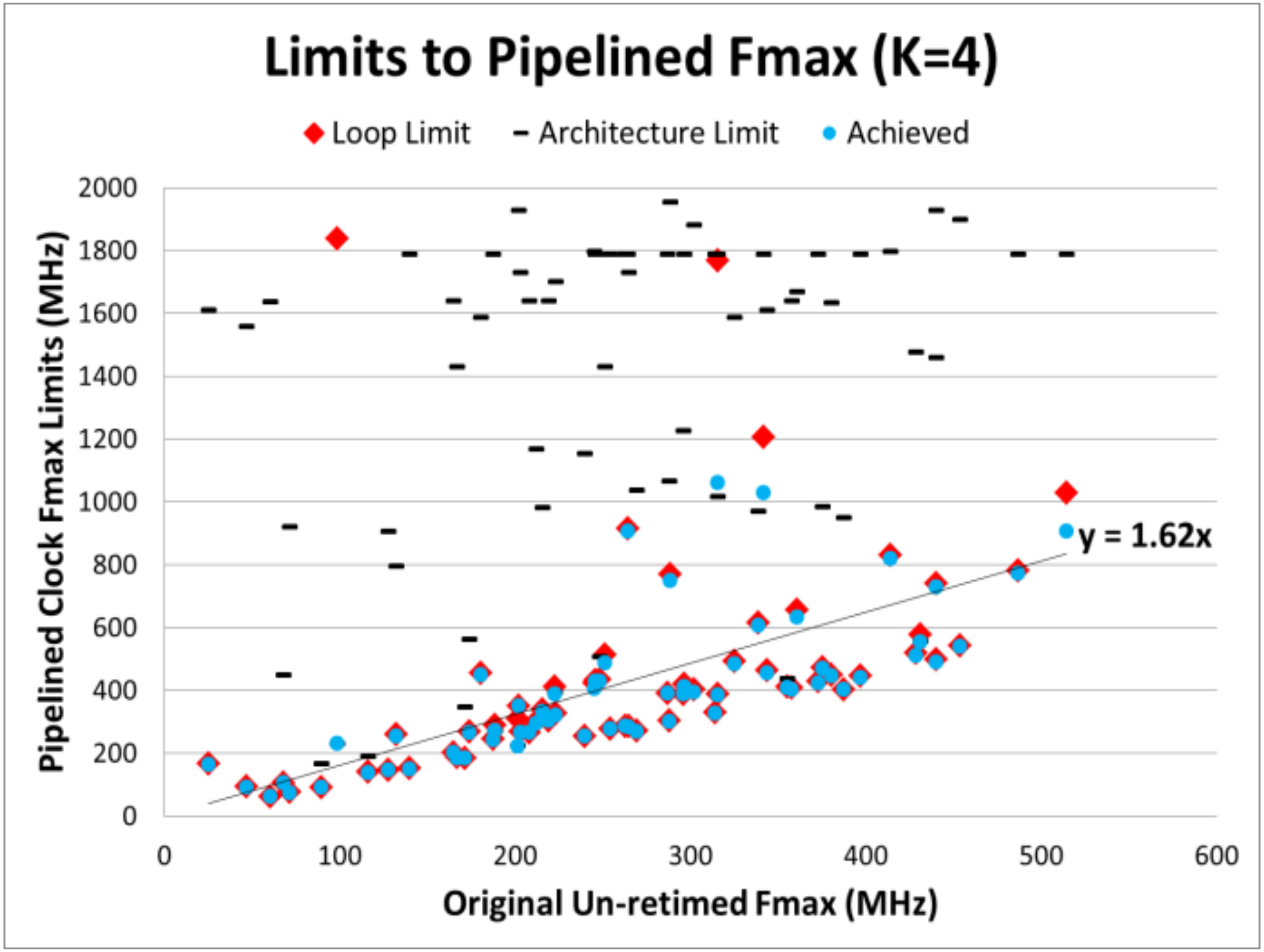
1. Retiming only: allow each FF in the user design to be retimed to anywhere there is a FF location
   - Preserve exact cycle by cycle behaviour
2. Pipelining: allow CAD flow to insert arbitrary latency in front of each clock domain and then retime
   - Preserves functional behaviour, but adds the same latency to all input to output paths
   - Note: can't ever put extra FFs in loops
3. Design modification: designer modifies the RTL to enable greater levels of pipelining while preserving functional requirements
   - Loops are critical in pipeline performance
   - Try and restructure design to minimize logic in loops + other techniques
   - Add pipelining in front of modules

Early Experimental Conditions/Parameters

- Used modified Arria 10 model and approximate pipelining hardware
- Pipeline largest clock domain in each circuit
- Use *k* = 4 to approximate lots of hardware
- Assume no constraints on clocks available to routing FFs
- Measure achieved fmax after retiming / pipelining
- Loop limit: delay through longest loop in circuit / number of FFs in the loop
  - Can't insert FFs in a loop without breaking functionality
- Arch limit: longest FF location to FF location delay in any path in the circuit

Early Limit Study Experimental Results

- Average 62% fmax increase; loop limited in most cases



Largest clock domain only

Refinements to CAD and Architecture

- Actual retiming done at the end of the CAD flow
  - After placement and routing

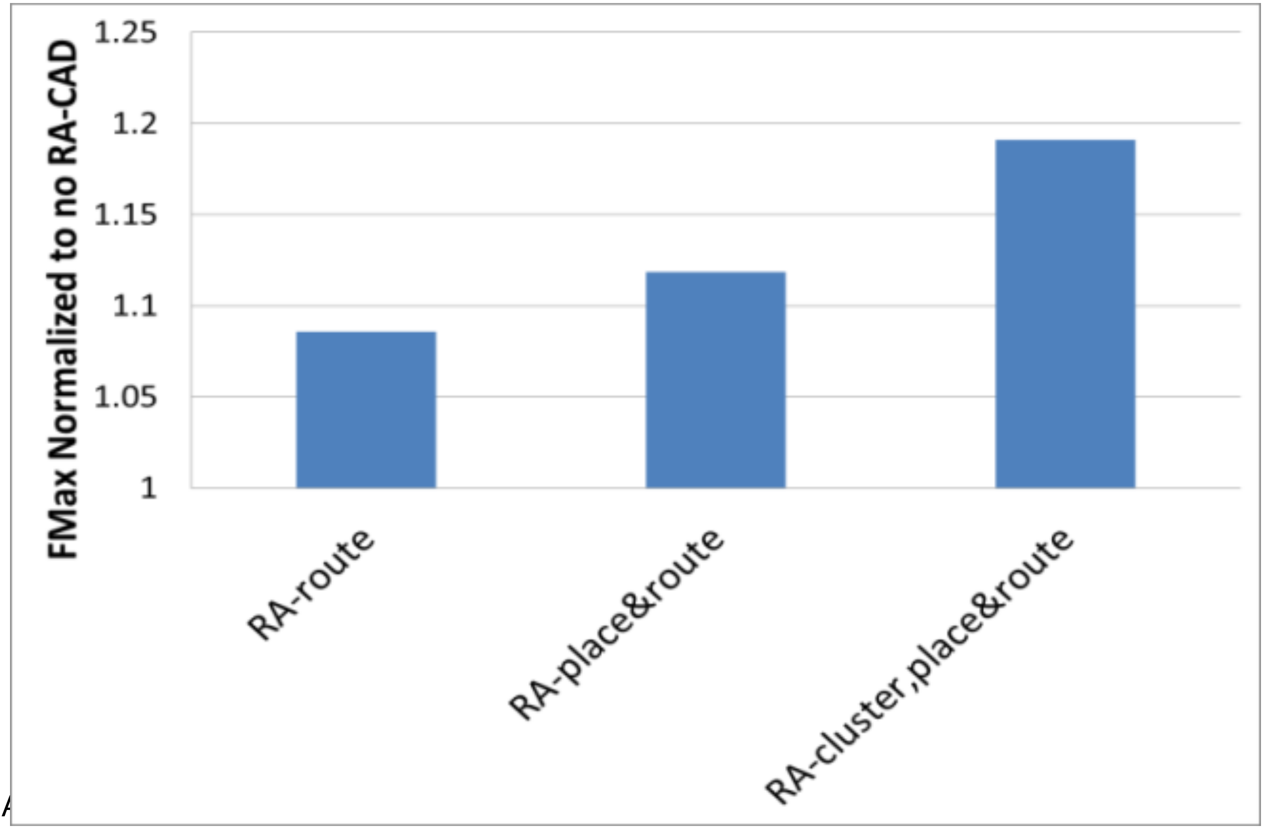
- Retiming-aware CAD flow uses continuous retiming based on skewed clocks at each FF
  - Skew clocks to optimize timing

- Paths that will be critical after retiming will have lower slack

- CAD can then target these paths for better clustering, placement, and routing

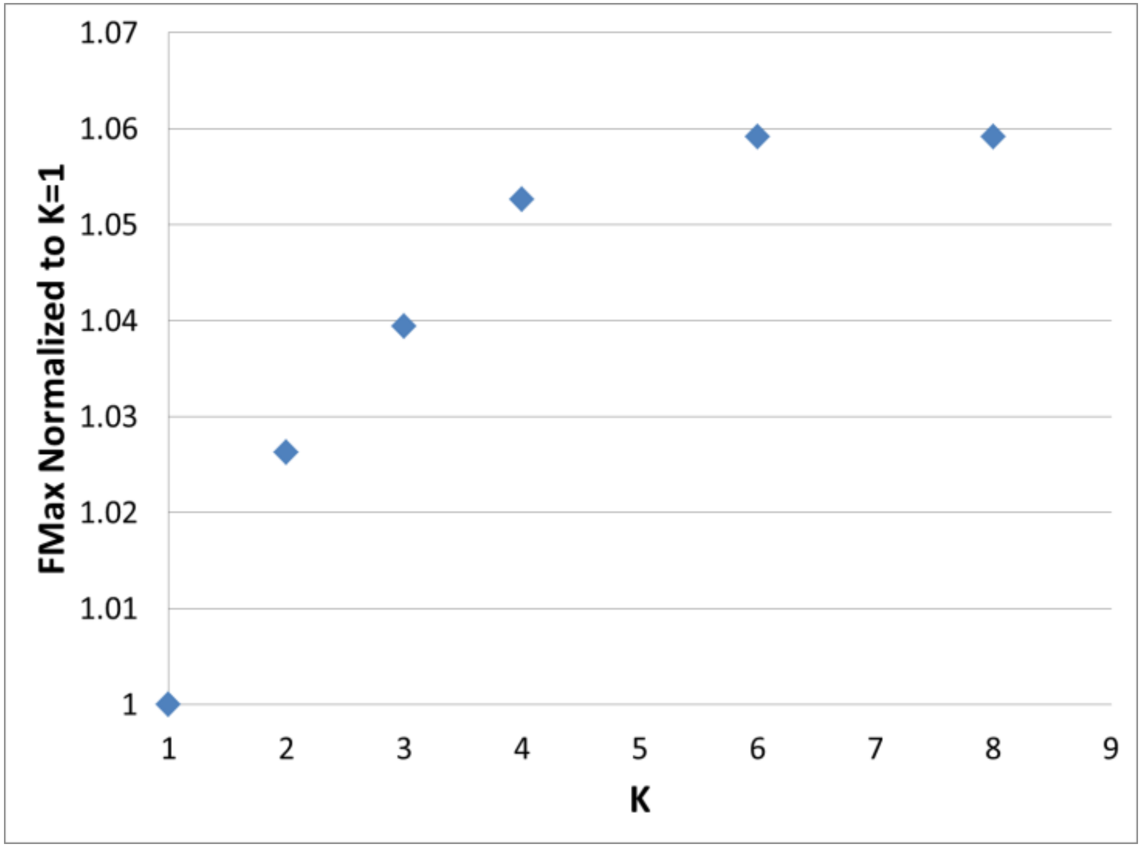
Impact of Retiming Aware CAD

- Up to 19% fmax improvement by focusing on paths that are critical for retiming
- Critical paths are generally those in loops
  - pure feedforward circuits can generally be deeply pipelined



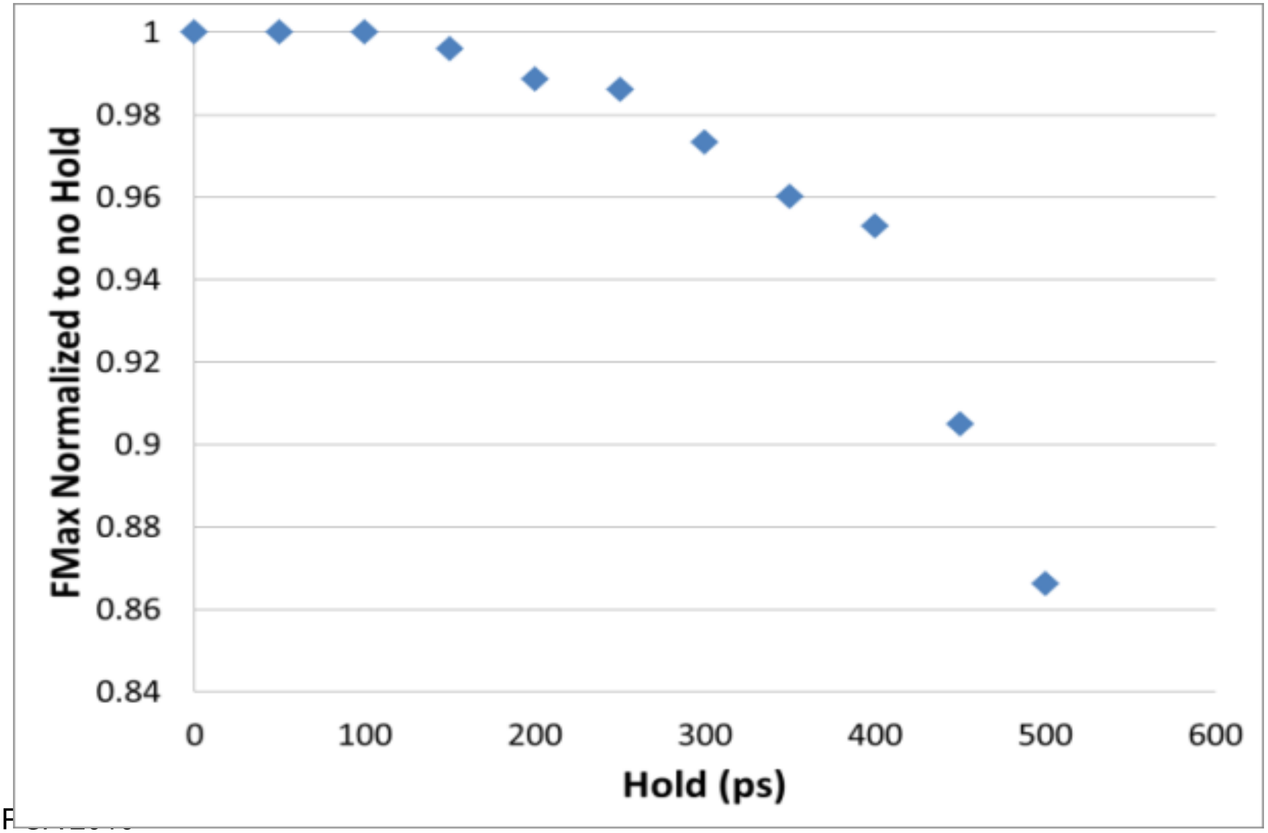
Effect of Number of FFs in Routing Mux: k

- About 5% fmax loss by dropping *k* to 1,
  - but much less than area cost of *k* = 4
  - Normalized to *k* = 1 result; we built in +5% fmax in original experiment



Hold Time Issue with Pulsed Latch

- Consequence of low-cost pulse latch: needs some hold time, else data can race through consecutive latches
- Don’t know exact value of hold time during early architecture experiments, but guess ~200ps

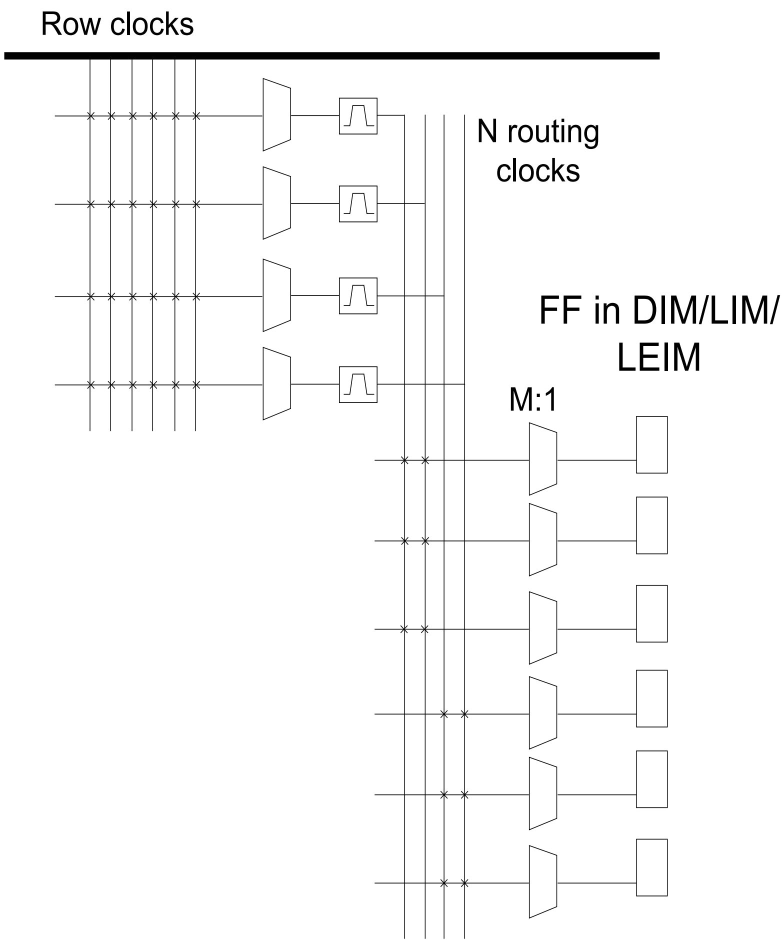


Clocking

- Prior academic work largely ignores clocking
- Real customer design set contains an average of 14 clocks per design, up to 67
- Stratix 10 global clock architecture is routable for better timing properties, but no changes relevant to the pipelined fabric
  - See Ebeling FPGA 2016 for details
- 6 clock lines available to provide global clocks to each LAB
- Approximately 160 routing mux FFs per LAB
  - Plus the 80 inside the LAB logic
- A 6:1 mux per latch would be too large/expensive
  - Several times larger than the FF
- Since many FF clock muxes, desire to minimize their size

Clocking Mux Architecture

- Divide the FFs into groups and make them share clocks
  - Ex: all short wires going left/right; all long wires going up/down, etc.
- Pick 1 or 2 clocks per group from the 6 available
- Each FF selects from those clocks, using a 2:1 mux or no mux at all
- Carefully tuning the groups resulted in **no** Fmax loss

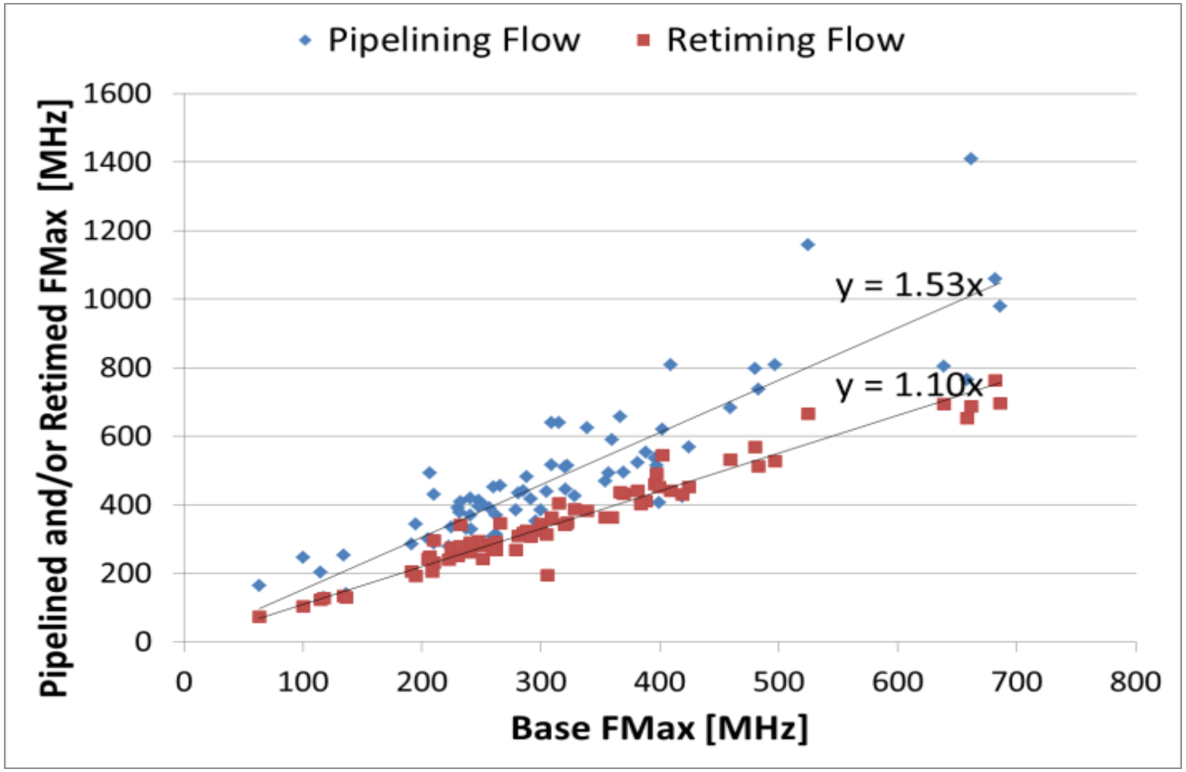


FPGA Architect's First Law of Entropy:

You have to run just to stay in place

Results Accounting for Realities

- With real CAD, *k* = 1, hold time, all domain fmax
- Retiming only: +10% fmax; pipelining: +53% fmax
  - Small domains have less benefit from pipelining



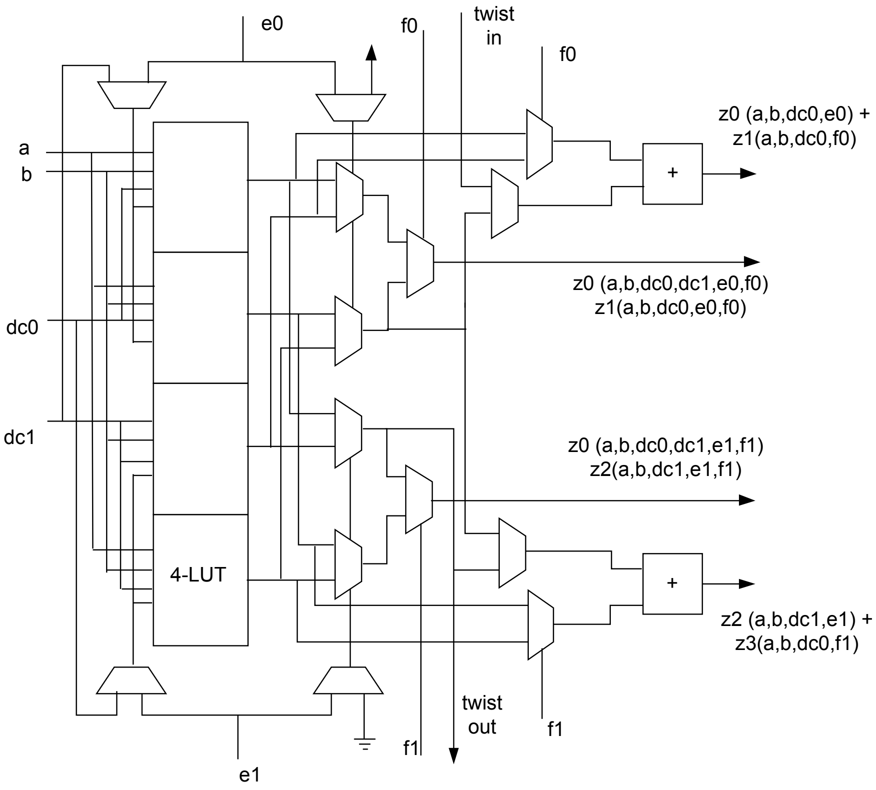
Other Architecture Changes - Logic Element Modification

- Stratix II to V have shared LUT mask (SLM)
- All provide 2 5-LUT with 8 inputs → 2 shared inputs, 3 unique
- SLM can build 2 6-LUT with identical functions, and 4 shared inputs
- Difficult for pipelining because internal stages of the LUT are used for two different logical functions
  - Can't independently retime
- Removed SLM
- Also removed complicated arithmetic (3 input adder and use of all 8 inputs)
- Push back part of adder into LUTs; simplify adder hardware
- Converted asynchronous clear and synchronous clear into 2 general purpose clears
- Synchronous load is now static only

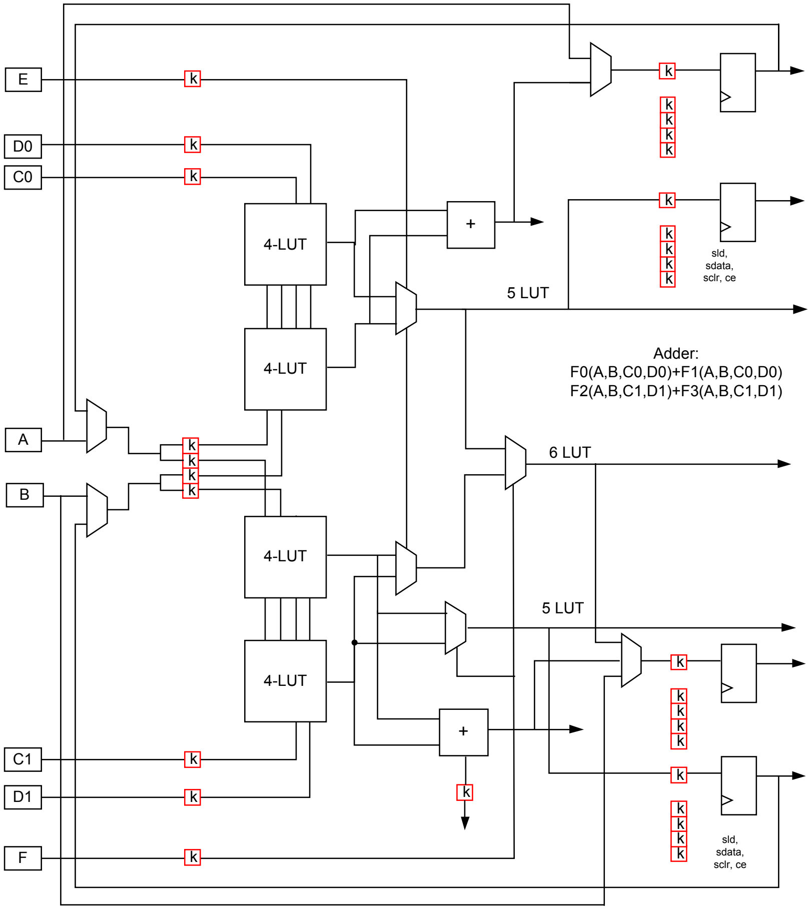
Simplified ALM

- Simpler, at least in comparison to previous, small fmax win

previous Stratix


e0
f0
twist in
f0
a
b
dc0
dc1
4-LUT
+
z0 (a,b,dc0,e0) + z1(a,b,dc0,f0)
z0 (a,b,dc0,dc1,e0,f0) z1(a,b,dc0,e0,f0)
z0 (a,b,dc0,dc1,e1,f1) z2(a,b,dc1,e1,f1)
+
z2 (a,b,dc1,e1) + z3(a,b,dc0,f1)
twist out
f1
f1
e1

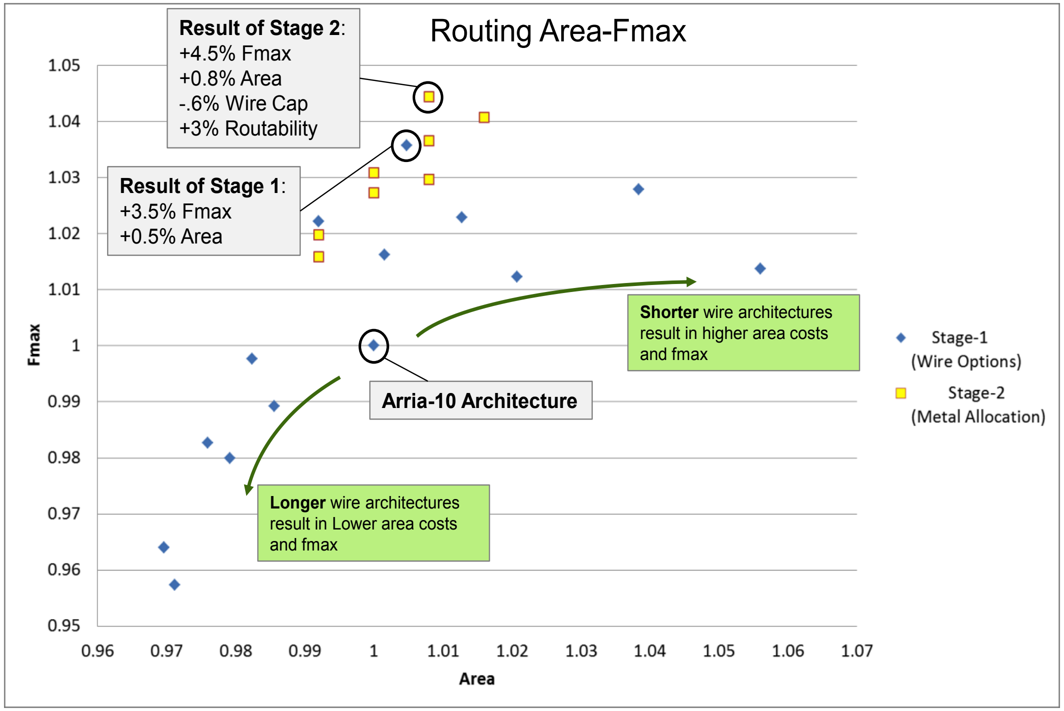
Stratix 10



Routing Optimization

- Larger variety of metal layers in Intel vs. TSMC
- 2 stage investigation of wire lengths and allocation to metal layers: +4.5% fmax at +0.8% area

H wire	H wire fraction	V wire	V wire fraction
H2	10%	V2	12%
H4	28%	V3	38%
H10	50%	V4	33%
H24	12%	V16	17%



Routing Area-Fmax

Result of Stage 2:
+4.5% Fmax
+0.8% Area
-.6% Wire Cap
+3% Routability

Result of Stage 1:
+3.5% Fmax
+0.5% Area

Shorter wire architectures result in higher area costs and fmax

Arria-10 Architecture

Longer wire architectures result in Lower area costs and fmax

How User Designs were Modified in Experiments

- Altera design expert worked with several customers
- Modified their designs to enable more pipelining and preserve functional requirements
- Full designs, not isolated cores
- Primarily reduced paths through loops
- Shannon decomposition of critical loop state
- Moving some computation that can be precomputed earlier in the pipeline
- Loop unrolling; works better with S 10 than previous
- Did back port design changes to Stratix V to measure architecture + design benefit

Results

- Redesign is 2.4X faster than S V
- Back port to S V is 1.23X faster
- Architecture + redesign + process is 92% faster than optimized design in S V

Module	S V (MHz)	Retime S10	Pipe S10	Redesign S10	Redesign SV
1A	320	380 (+19%)	460 (+44%)	489 (+53%)	329 (+3%)
1B	327	482 (+47%)	626 (+91%)		
1C	319	432 (+35%)	457 (+43%)		
2A	250	429 (+72%)	454 (+82%)	942 (+277%)	347 (+39%)
3A	191	290 (+52%)	291 (+52%)	748 (+292%)	359 (+88%)
4A	403	599 (+49%)	638 (+58%)	725 (+80%)	411 (+2%)
4B	384	555 (+45%)	570 (+48%)	695 (+81%)	391 (+2%)
Geo %		45%	59% (*)	136%	23%

(*) typo in paper: 49% should be 59%

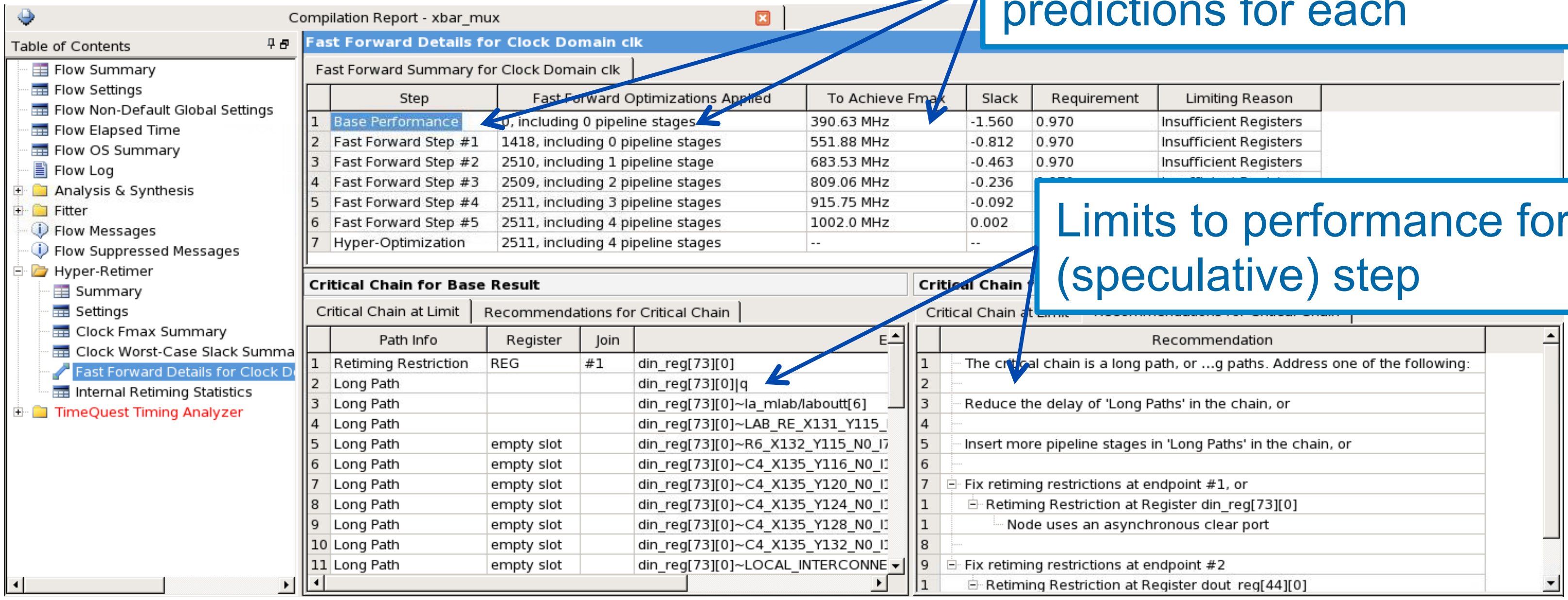
Quartus Enhancements for Stratix 10

- Constraint-based retiming to solve for minimum clock period

Design optimization advisor:

- Where should asynchronous resets be converted into synchronous?
- Where should pipeline registers be added to enable deeper pipelining?
- Which loops limit the performance?
- Use set of infeasible constraints from retimer to show where the conversions need to be done
- Can automatically modify the retiming graph to model the proposed change and report on potential fmax

Quartus Enhancements for Stratix 10



Compilation Report - xbar_mux

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Flow Messages
- Flow Suppressed Messages
- Hyper-Retimer
  - Summary
  - Settings
  - Clock Fmax Summary
  - Clock Worst-Case Slack Summa
  - Fast Forward Details for Clock D
  - Internal Retiming Statistics
- TimeQuest Timing Analyzer

Fast Forward Details for Clock Domain clk

Fast Forward Summary for Clock Domain clk

	Step	Fast Forward Optimizations Applied	To Achieve Fmax	Slack	Requirement	Limiting Reason
1	Base Performance	0, including 0 pipeline stages	390.63 MHz	-1.560	0.970	Insufficient Registers
2	Fast Forward Step #1	1418, including 0 pipeline stages	551.88 MHz	-0.812	0.970	Insufficient Registers
3	Fast Forward Step #2	2510, including 1 pipeline stage	683.53 MHz	-0.463	0.970	Insufficient Registers
4	Fast Forward Step #3	2509, including 2 pipeline stages	809.06 MHz	-0.236	0.970	Insufficient Registers
5	Fast Forward Step #4	2511, including 3 pipeline stages	915.75 MHz	-0.092		
6	Fast Forward Step #5	2511, including 4 pipeline stages	1002.0 MHz	0.002		
7	Hyper-Optimization	2511, including 4 pipeline stages	--	--		

Critical Chain for Base Result

Critical Chain at Limit | Recommendations for Critical Chain

	Path Info	Register	Join	E
1	Retiming Restriction	REG	#1	din_reg[73][0]
2	Long Path			din_reg[73][0]|q
3	Long Path			din_reg[73][0]~la_mlab/laboutt[6]
4	Long Path			din_reg[73][0]~LAB_RE_X131_Y115_
5	Long Path	empty slot		din_reg[73][0]~R6_X132_Y115_N0_I7
6	Long Path	empty slot		din_reg[73][0]~C4_X135_Y116_N0_I1
7	Long Path	empty slot		din_reg[73][0]~C4_X135_Y120_N0_I1
8	Long Path	empty slot		din_reg[73][0]~C4_X135_Y124_N0_I1
9	Long Path	empty slot		din_reg[73][0]~C4_X135_Y128_N0_I1
10	Long Path	empty slot		din_reg[73][0]~C4_X135_Y132_N0_I1
11	Long Path	empty slot		din_reg[73][0]~LOCAL_INTERCONNE

Critical Chain f

Critical Chain at Limit | Recommendations for Critical Chain

	Recommendation
1	The critical chain is a long path, or ...g paths. Address one of the following:
2	
3	Reduce the delay of 'Long Paths' in the chain, or
4	
5	Insert more pipeline stages in 'Long Paths' in the chain, or
6	
7	Fix retiming restrictions at endpoint #1, or
1	Retiming Restriction at Register din_reg[73][0]
1	Node uses an asynchronous clear port
8	
9	Fix retiming restrictions at endpoint #2
1	Retiming Restriction at Register dout_reg[44][0]

Speculative modifications and Fmax predictions for each

Limits to performance for particular (speculative) step

FPGA 2016

Conclusions

- Direct drive routing enables a very low cost FF in each routing mux
- Most of the cost is elsewhere
  - FFs in the logic
  - LAB level clock muxing
- Put one FF location everywhere you can think of, and carefully optimize the clock muxing
- Subtle interactions between pipelining and internals of a complicated logic element
- Optimized designs 92% faster in S 10 than S V
- Quartus support to help designers identify where to focus