Team *Geriatric Guys with Gates* Participates in the VDF Design Competition

It's All About Speed and Power

Kurt Baty
WSFDB Consulting

Steve Golson
Trilobyte Systems

~~March 18–19, 2020~~ CANCELLED

Outline

Steve talks

Kurt talks

Steve and Kurt talk some more

Power problems

Interesting questions from the audience

What is VDF – Verifiable Delay Function

- Verifiable Delay Functions
  - take a prescribed time to compute, even on a parallel computer
  - produce a unique output that can be efficiently and publicly verified
- Possible uses
  - public randomness beacons
  - leader election in consensus protocols
  - proofs of replication
- Potential problems
  - if malicious actors have access to specialized hardware,
    they can speed up VDF evaluation,
    breaking the security of the protocols that rely on VDFs

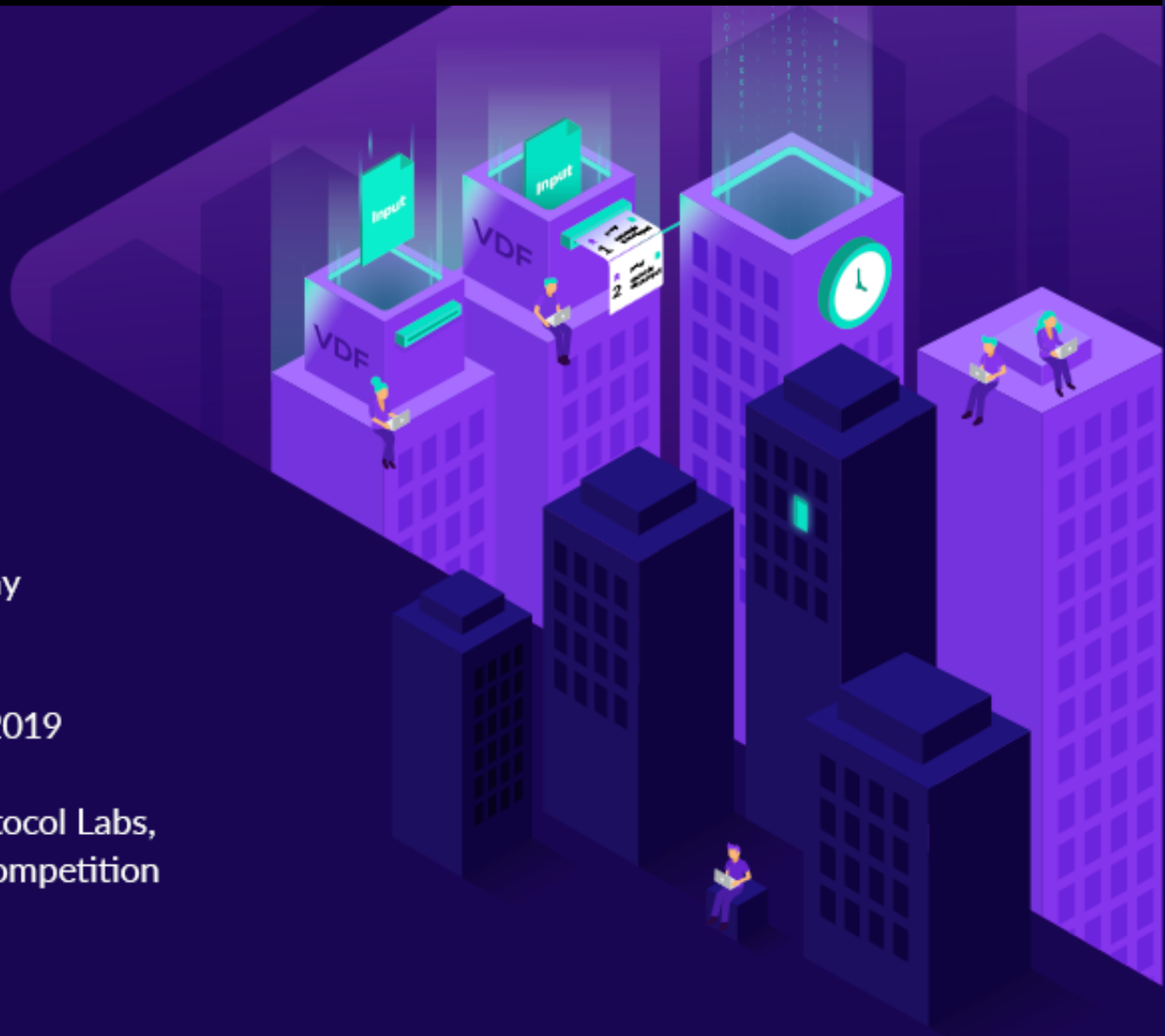
How fast can the VDF be calculated?

VDF | FPGA Design Competition

Are you up for the challenge?

$100,000 Prize to Forever Change Blockchain

- The problem: Given 1024-bit input x, compute the verifiable delay function h=x^(2^t) mod N as fast as possible.
- Timeline: The overall competition will run from Aug 1 - Dec 30, 2019
- Prize: The Ethereum Foundation, the Interchain Foundation, Protocol Labs, Supranational, Synopsys, and Xilinx are sponsoring a $100,000 competition with support from AWS.



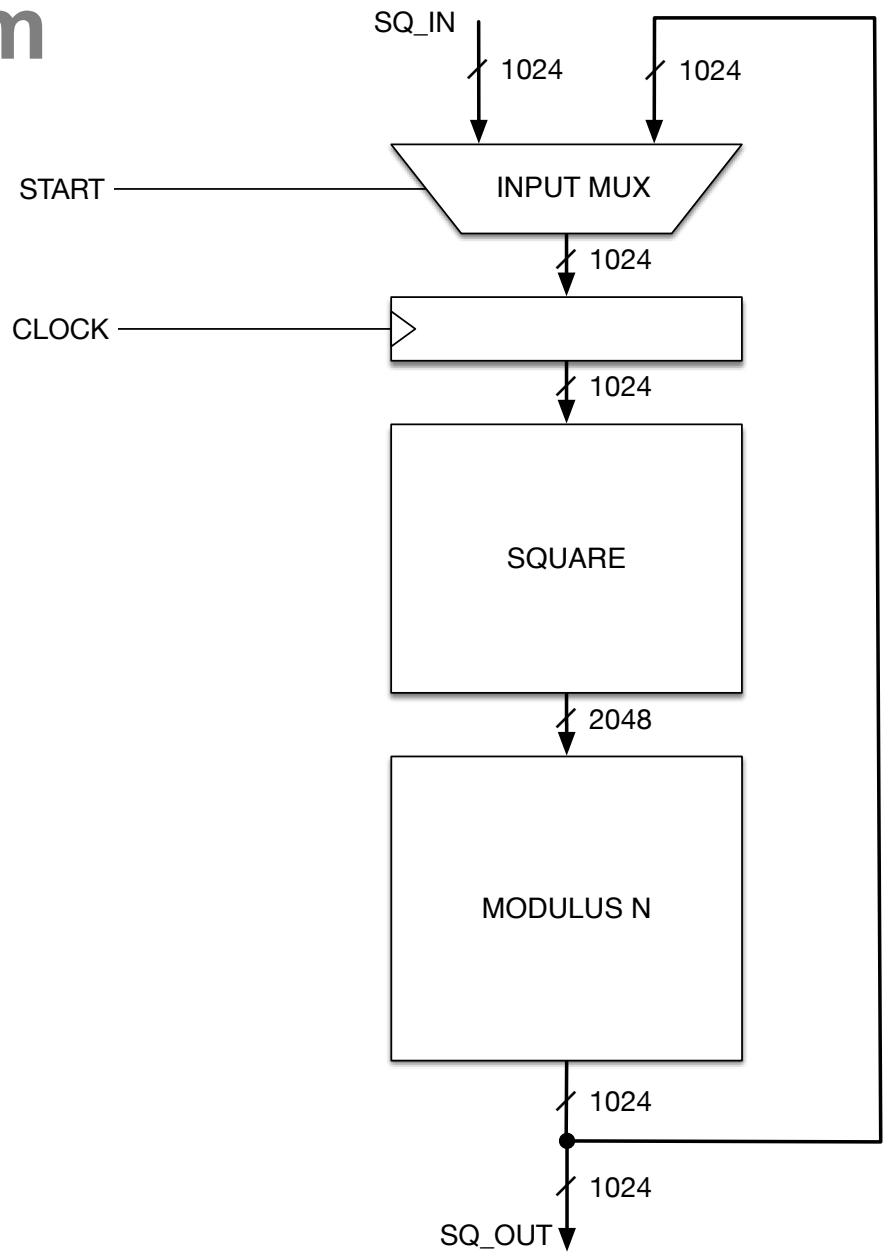
Read more
https://www.vdfalliance.org/contest

ethereum foundation
SUPRA NATIONAL
INTERCHAIN FOUNDATION
Protocol Labs

VDF Alliance FPGA Design Competition

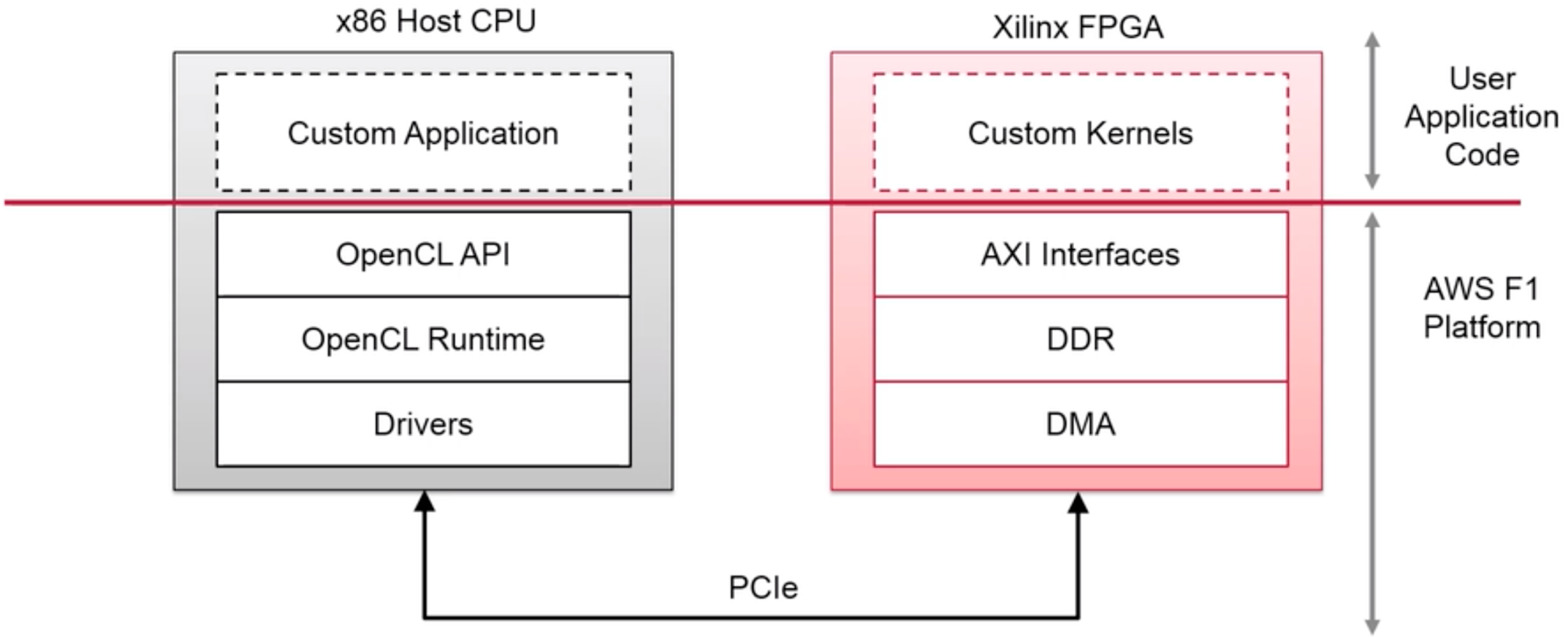
- Problem
  - compute function *h(x)* as fast as possible, where $h(x) = x^{2^t} \bmod N$
  - input *x* is any 1024-bit integer
  - modulus *N* is fixed 1024-bit integer: *N*=124066695684124741344 6…689826625594484331
  - time parameter is fixed: $t=2^{30}$
- Technology
  - Xilinx UltraScale+ device
  - Speed measured on AWS F1 target
  - SDAccel tool flow based on Vivado
- Round 1 : ~7 weeks : Aug 8 – Sept 30, 2019
- Prize : $3,000 for every ns faster than baseline 50 ns/sq latency

Conceptual Block Diagram



- Square and modulus logic can be flow-through or registered
- Iterate for 2^{30} clocks
  - about 50 seconds elapsed time

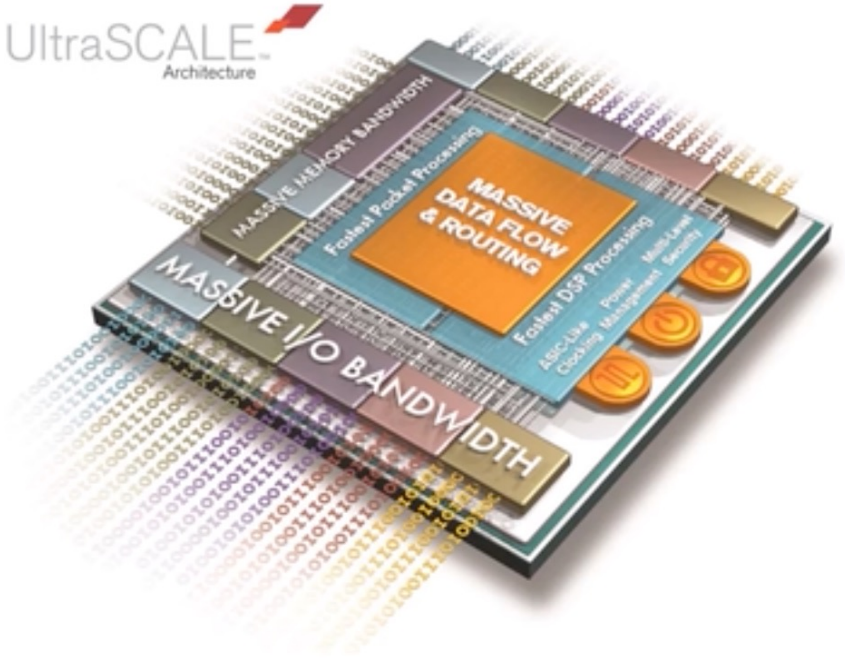
AWS F1 Platform Model



© Copyright 2017 Xilinx
XILINX ALL PROGRAMMABLE.

The AWS-VU9P-F1 Hardware Platform

- Xilinx UltraScale+ VU9P, 16nm process
- Approx. 2.5 million programmable logic cells
- Approx. 6,800 Digital Signal Processing engines
- 4 DDR4 channels, each accessing a 16 GiB, 72-bit wide, ECC-protected memory
- Dedicated PCIe x16 interface to the CPU
- Virtual JTAG interface for debugging

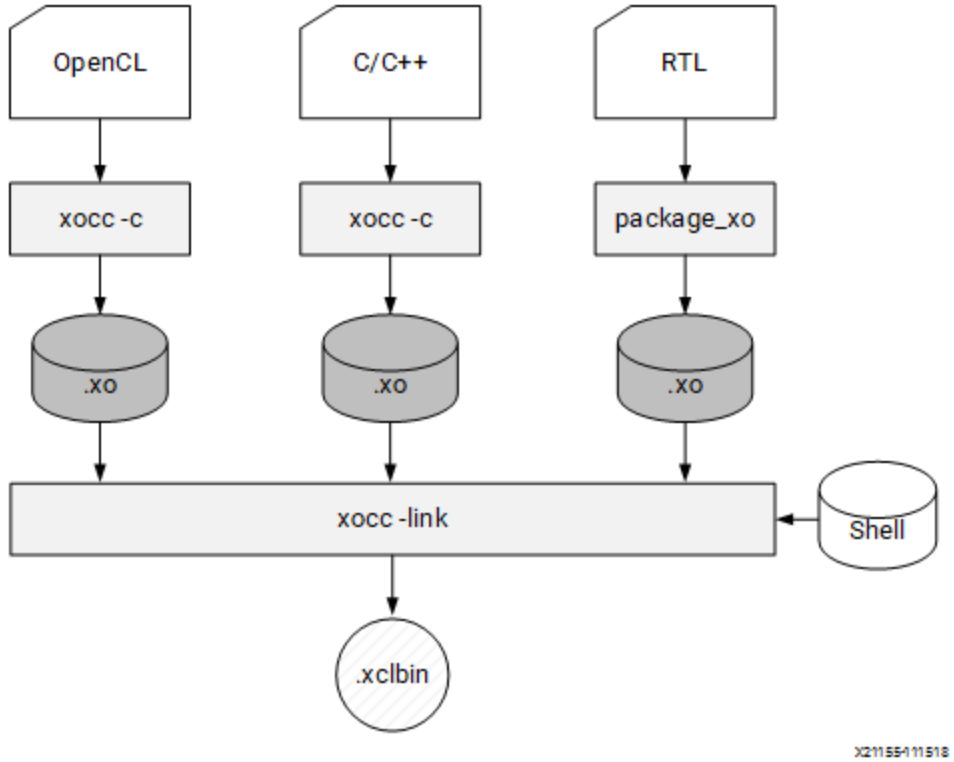


© Copyright 2017 Xilinx

XILINX ALL PROGRAMMABLE.

FPGA Build Process

- xocc compiler automatically calls Vivado HLS and Vivado Design Suite
- Predefined settings
  "proven to provide good quality of results"



"…hardware-savvy developers can fully leverage these tools and use all their available features…"

Baseline design: Ozturk architecture

"Modular Multiplication Algorithm Suitable For Low-Latency Circuit Implementations"

- by Erdinç Öztürk, Aug 26, 2019
- To be published in *IEEE Transactions on Circuits and Systems I: Regular Papers*

- Iterative design, reusing DSP hardware across multiple clock periods
- Latency is 8 clock cycles (sometimes 10), nominal 50 ns/sq
- Calculate using 16-bit coefficients
- Modulus lookup using block RAM (BRAM) configured as ROM
- 3-to-2 compressors
- modsqr logic fits in one die of 3-die Xilinx part

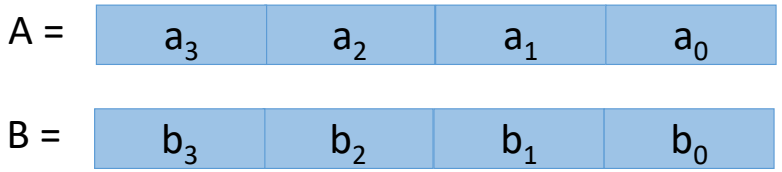
Low Latency Algorithm/Construction

- Represent an integer as a polynomial
- Redundant representation
  - Extra bit per coefficient
  - Extra coefficient per polynomial
- Suitable for a repeated-multiply setting
  - Exponentiation
  - Incomplete reduction

slide courtesy of Erdinç Öztürk

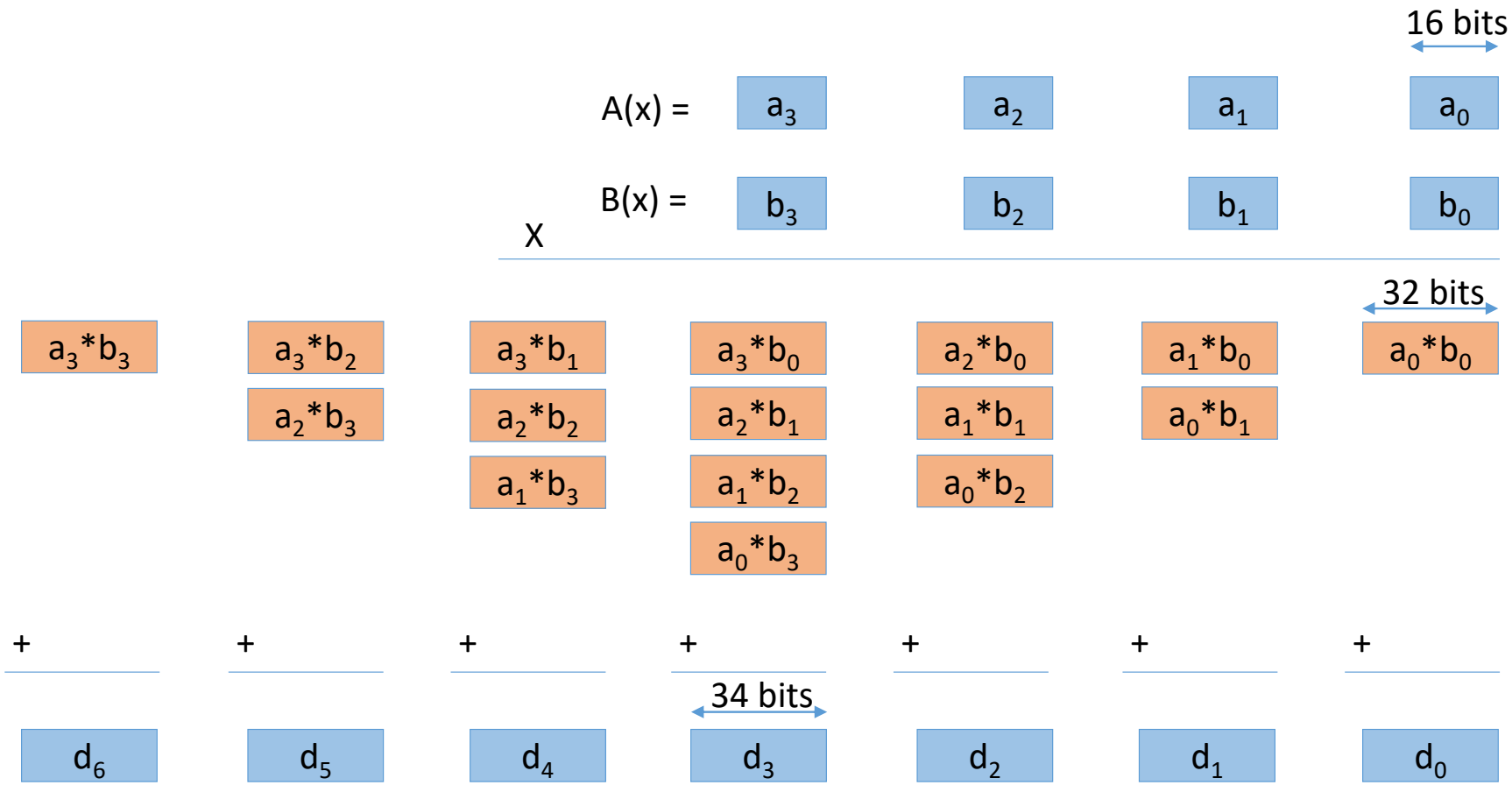
Low Latency Algorithm/Construction

- Example construction:
  - Coefficient length: 16 bits
  - Integer length: 64 bits
- Integers A and B are represented as follows:
  - $A = a_3*x^3 + a_2*x^2 + a_1*x + a_0$
  - $B = b_3*x^3 + b_2*x^2 + b_1*x + b_0$
  - $x=2^{16}$

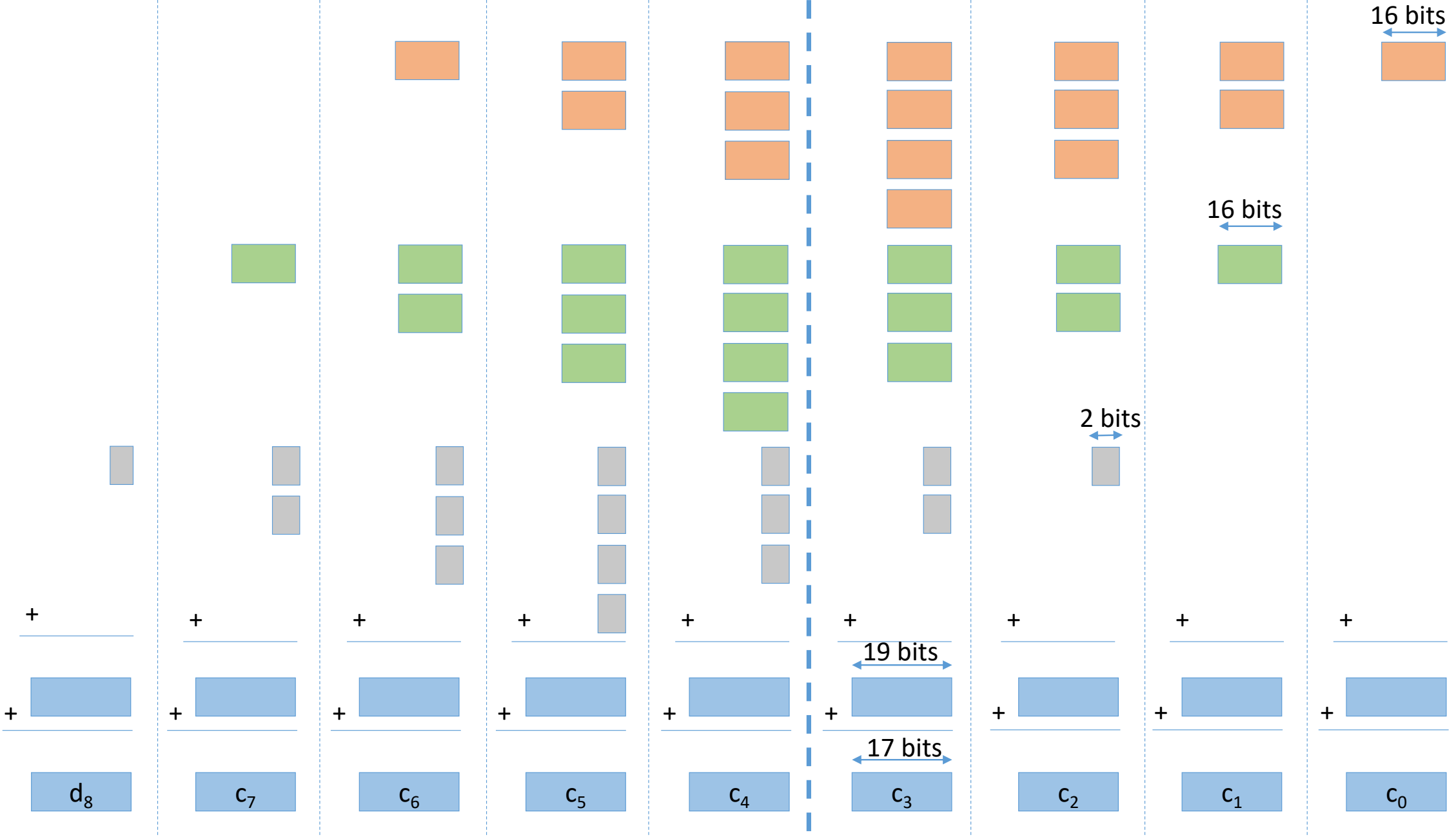


slide courtesy of Erdinç Öztürk

Low Latency Algorithm/Construction

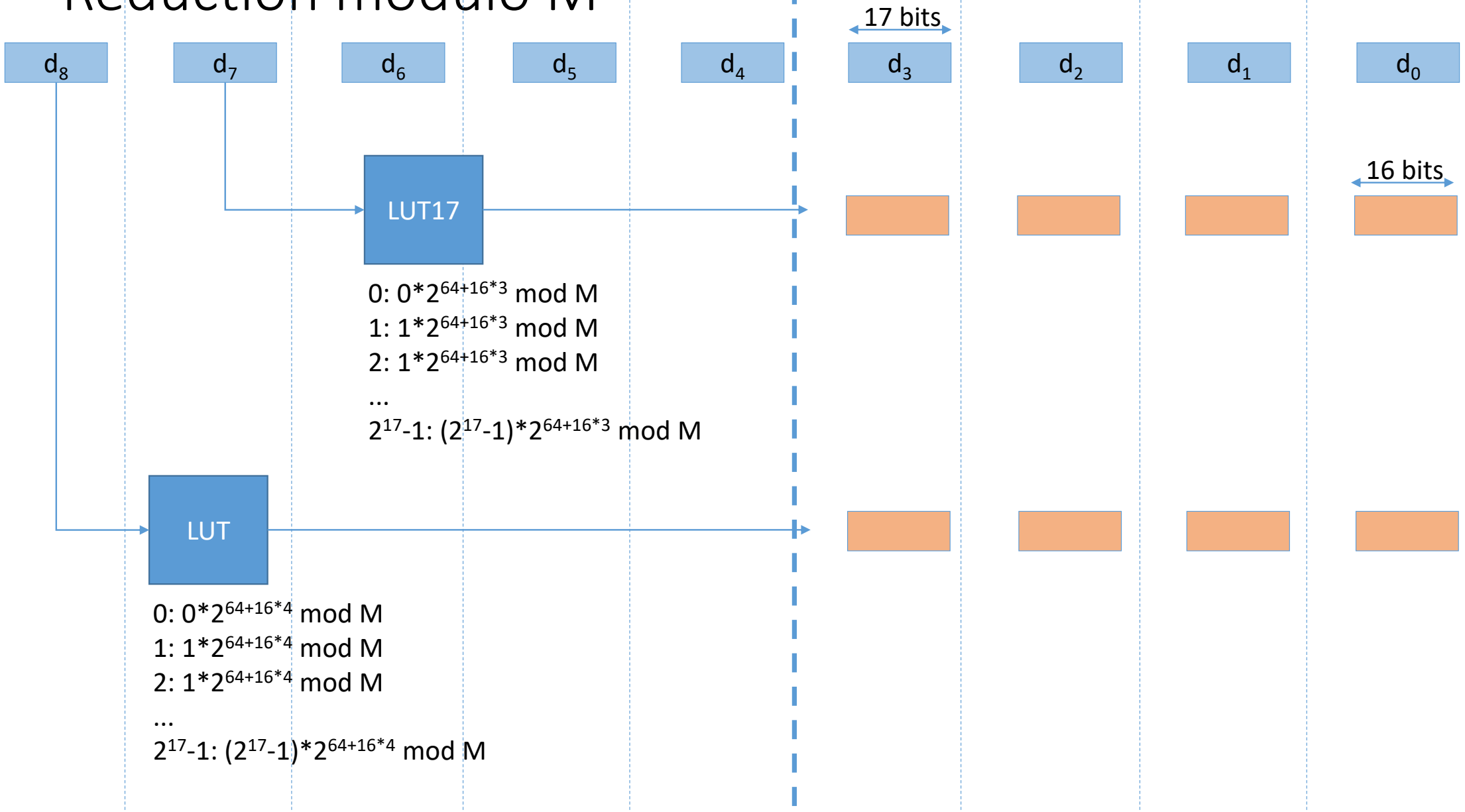


slide courtesy of Erdinç Öztürk

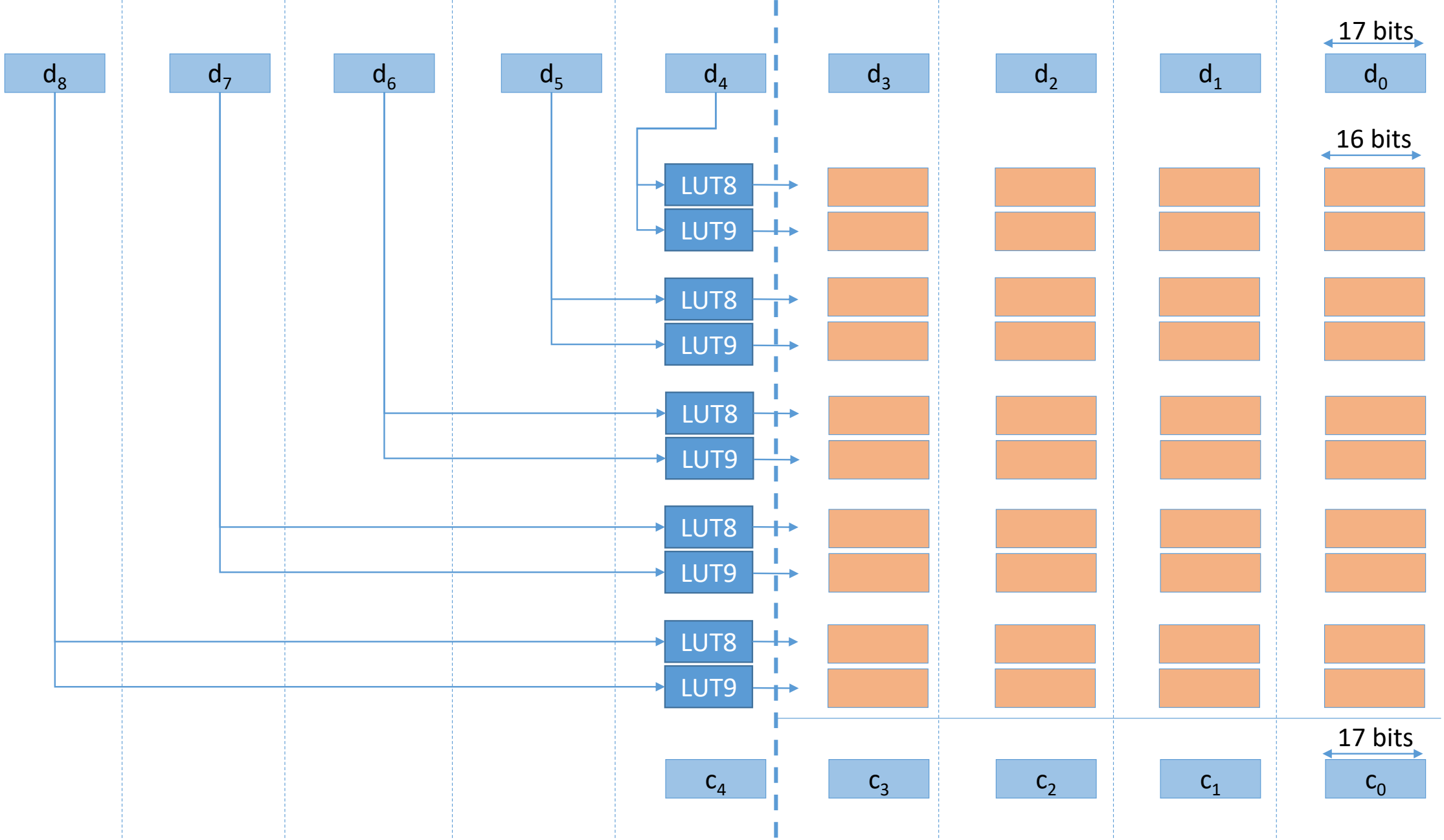


slide courtesy of Erdinç Öztürk

Reduction modulo M

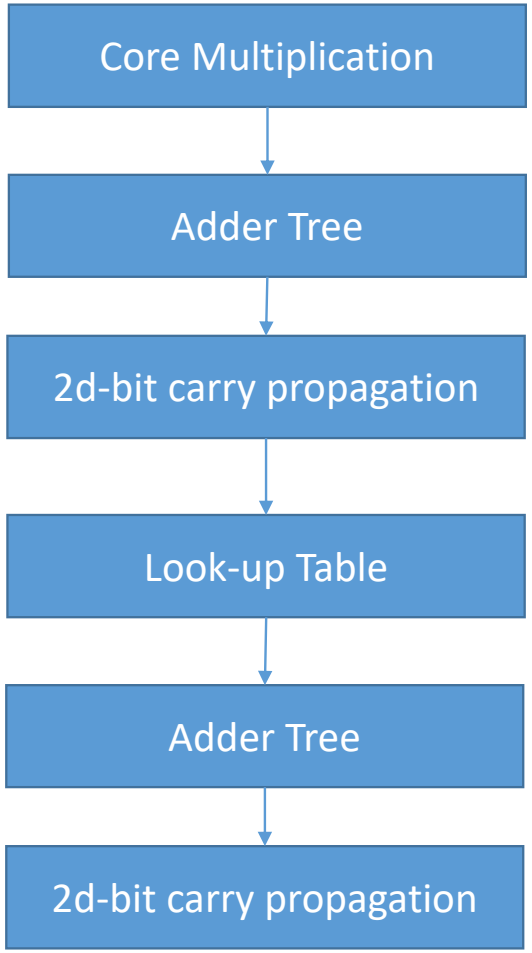


slide courtesy of Erdinç Öztürk



slide courtesy of Erdinç Öztürk

Critical Path



d: coefficient length
17 for example construction

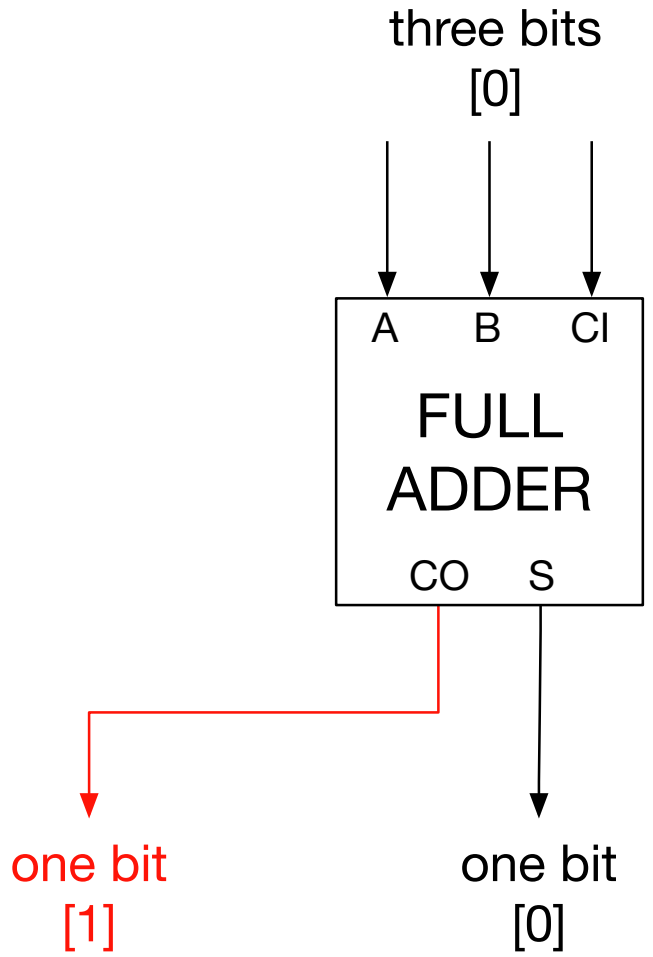
slide courtesy of Erdinç Öztürk

What is a compressor in computer arithmetic?

- Large bit-width arithmetic is all about *addition*
- Multipliers are built up from smaller units
  - Partial products must be *added* together
- Modulus is performed with ROM lookups
  - Lookup results must be *added* together
- These additions are performed with *compressors*

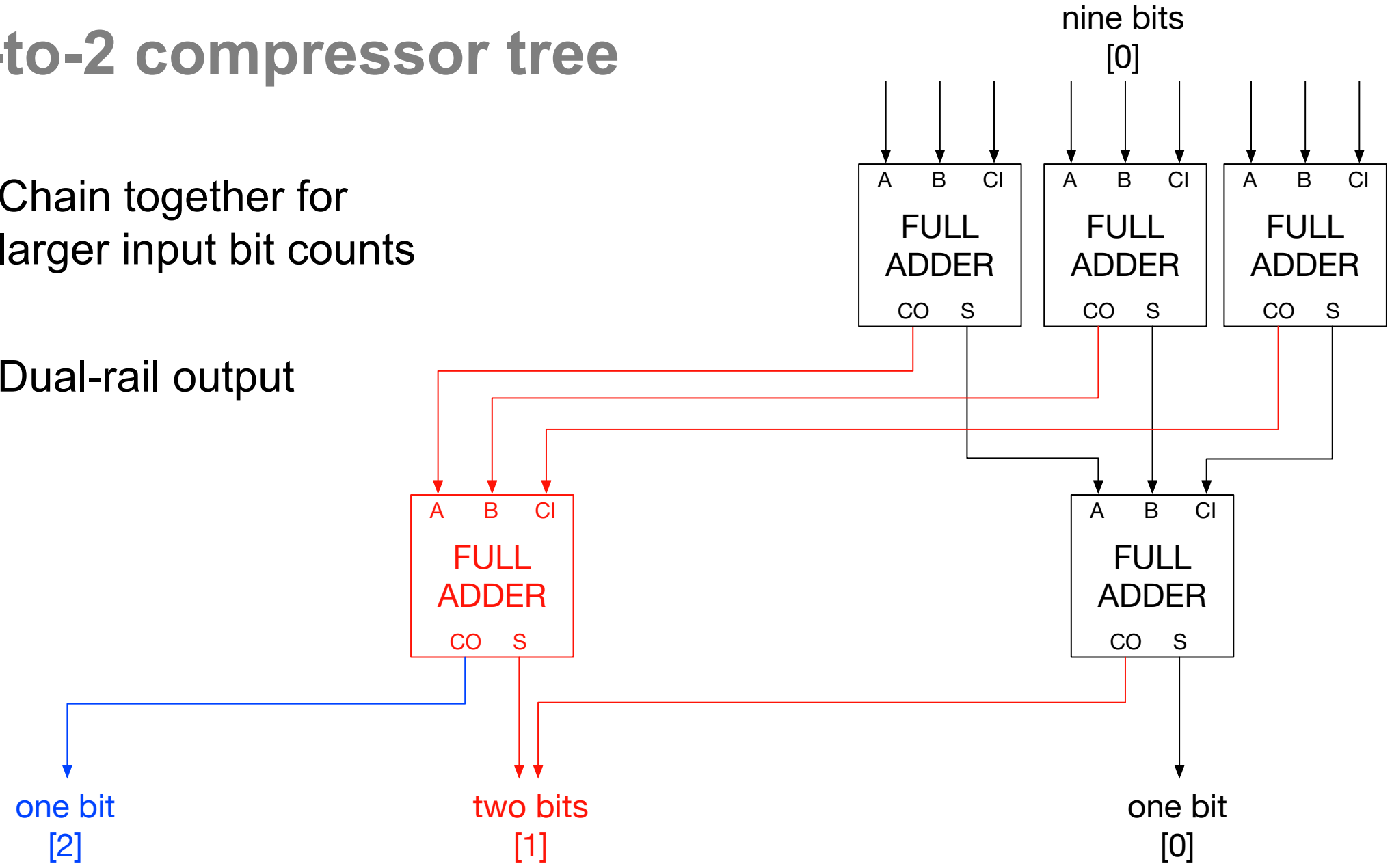
3-to-2 compressor

- Full adder is a 3-to-2 compressor
- Input : Three bits with same binary weight
- Output : Two bits, weighted

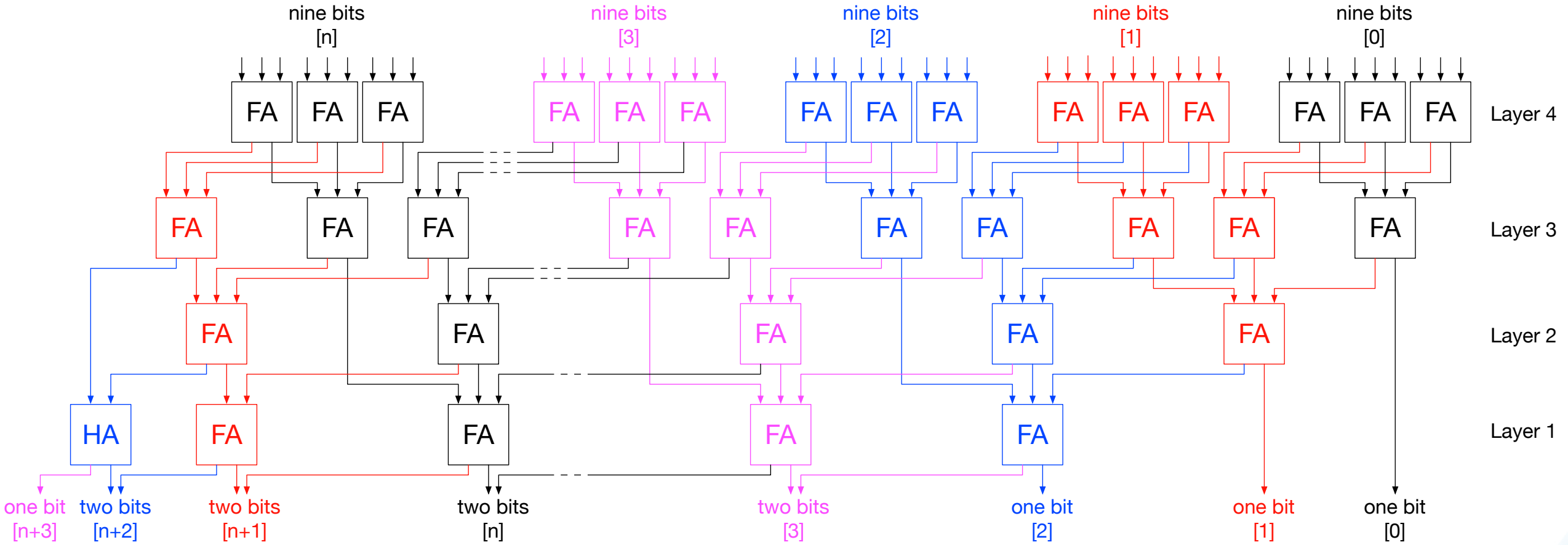


3-to-2 compressor tree

- Chain together for larger input bit counts
- Dual-rail output



3-to-2 compressor tree



Team *Geriatric Guys with Gates*

Why enter?

1. Fun
2. Fame
3. Financial reward

Team GGG in Round 1

- Started late, and didn't have much time to devote to the project
- Research better compressor in Xilinx FPGAs
- Build 6-to-3 compressor with three LUT6 (inputs I0-I5, outputs S,C2 and C4)
- SystemVerilog code infers LUT6 rather than hard-instancing of LUT6
- Improve implementation flow (Vivado settings)
- Build, simulate, synthesis, implementation

- Results, according to Vivado signoff STA
  - Clock frequency improved from 161 MHz to 177.9 MHz
  - Latency improved to 44.96 ns which is 5 ns faster than baseline (10% improvement)

- We never tried it on F1 hardware…

Round 1 Competition Results

- 30 individuals signed up, in 18 teams
- 5 teams submitted designs but only *two* had working designs!
- All entries are derivatives of Ozturk design
- GGG design on F1 ran at 48 ns, not the expected 45 ns
- Winning design latency 28.6 ns !! 21.4 ns faster than baseline !!

"The working designs were from professionals with prior hardware, FPGA, and infrastructure experience."

"…building hardware is simply hard and time consuming."

Round 1 Competition Results

Qualifies	Team Name	Directory	Notes
Yes	Eric Pearson	eric_pearson-1	Fully functional at targeted 31.5 ns. Contestant did not use Vivado behavioral simulation, required additional instructions.
Yes	Eric Pearson	eric_pearson-2	Fully functional at targeted 28.6 ns/sq.
No	FPGA Enthusiast	fpga_enthusiast-1	Clocking issues on AWS F1 caused design to run at 56ns/sq instead of 45.7.
No	FPGA Enthusiast	fpga_enthusiast-2	Design fails routing due to congestions
No	FPGA Enthusiast	fpga_enthusiast-3	Design fails to close timing
No	Silicon Tailor	silicon_tailor-30	Faults FPGA, crashes host. Likely caused by PLL locking issue.
No	Silicon Tailor	silicon_tailor-291	Faults FPGA, crashes host. Likely caused by PLL locking issue.
No	Silicon Tailor	silicon_tailor-296	Faults FPGA, crashes host. Likely caused by PLL locking issue.
Yes	Geriatric Guys with Gates	geriatric_guys_with_gates	48ns / sq
No	Andreas Brokalakis	andreas_brokalakis	Design did not simulate or synthesize. After adding missing files hit resource constraint problems. After adjusting pblocks and target clock frequency did not meet target timing. Achieved 40ns/sq (but got wrong answer).

Round 1 Competition Results

- GGG was one of only two teams with a working design
- Next time, we will push the flow all the way through to hardware on F1 target

Round 2

- Round 2 : ~11 weeks : Oct 16 – Dec 31, 2019
- Prize : $5,000 for every ns faster than baseline
- New baseline is Eric Pearson's Round 1 winning design at 28.6 ns/sq
- What had Eric done?
  - Unrolled to 2 cycles vs Ozturk design 8 cycles
  - Parallel 1024-bit square using 17x17 DSP multipliers
  - Modulus lookup using 5/6/6-bit LUTRAM instead of BRAM
  - Full adders as 2-to-1 compressors
  - CARRY8 adder trees
  - Design spread throughout all three SLR die
  - Used MMCM to generate arbitrary clock frequency

Team GGG in Round 2

- DSP reduction
  - Changed 17-bit arithmetic to 51-bit arithmetic
  - Number of DSPs dropped from Eric's 2,211 to 1,386
- Used 696 BRAMs which saved 109k LUTs
- Moved critical path logic into one die, saving SLR crossing penalty
- 3-to-1 compressor
- Final 51-bit mux/full-adder/flop, implemented as eight bits in a single CLB
- Unrolled pipeline to single clock stage with arbitrary internal phase edges
- Wrapper and AXI interface uses fewer pipeline stages and simple CDC

Team GGG in Round 2

- Synthesis and implementation improvements

- Flow-through design meant huge number of timing paths
  - Router was unhappy and frequently failed to route
  - Reduced the number of paths by adding registers to LUT addresses
  - These registers had bypass muxes, hard-wired to bypass the flops
  - `set_case_analysis` on muxes during route caused router to see the flops

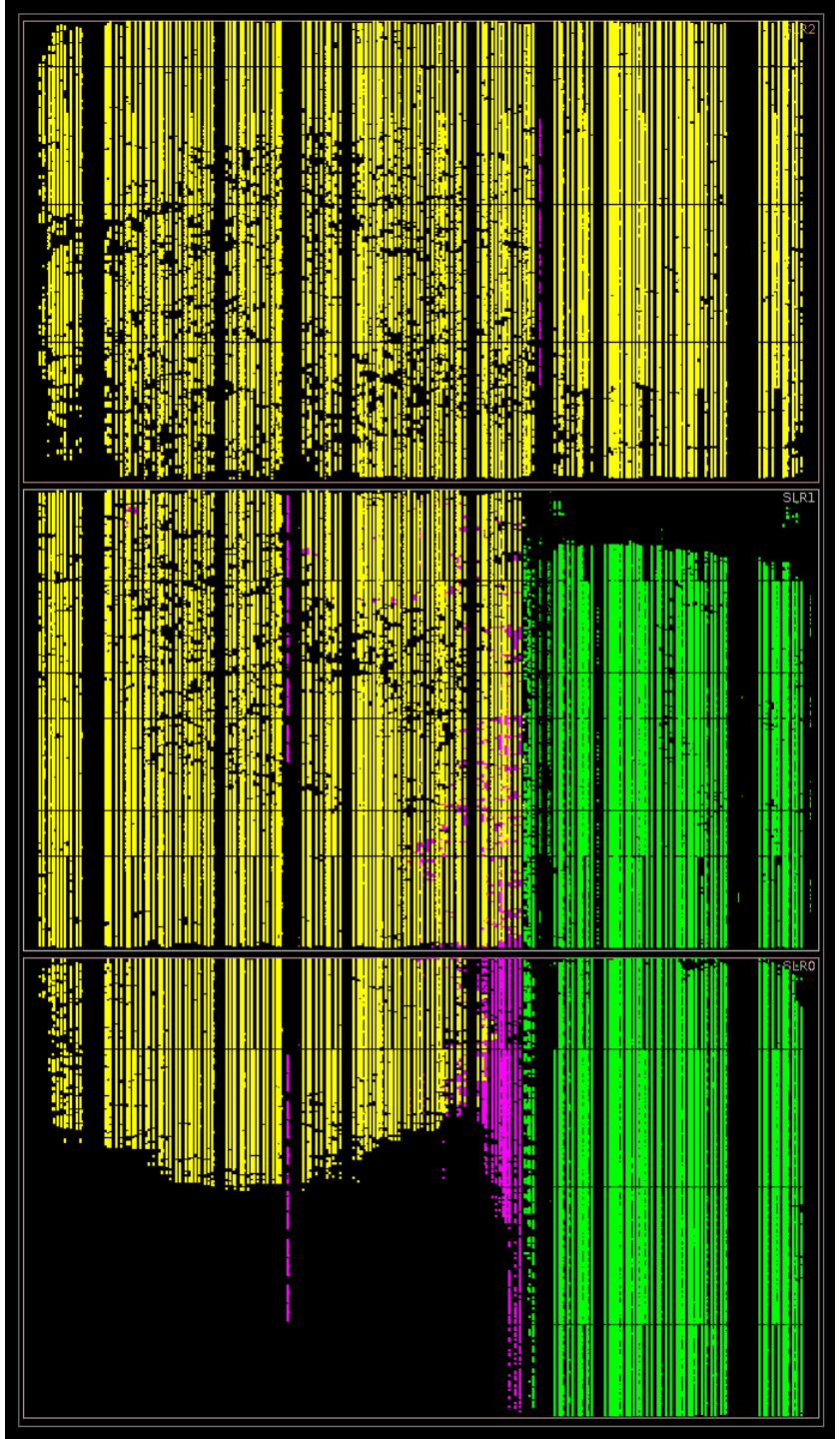
- MMCM programming problems
  - SDAccel flow and scripts behave in mysterious ways

- Latency 25.6 ns/sq, which is 3 ns improvement over baseline

Significant reduction in utilization

	Total LUT	Total nets	modsqr LUT	modsqr DSP	modsqr CARRY8	modsqr BRAM
Eric Pearson Round 1	624,773	3,257,865	467,109	2,211	46,879	0
GGG Round 2	412,891	2,564,229	254,503	1,527	29,416	696
	−34%	−21%	−46%	−31%	−37%	+∞%

Eric Pearson Round 1 design

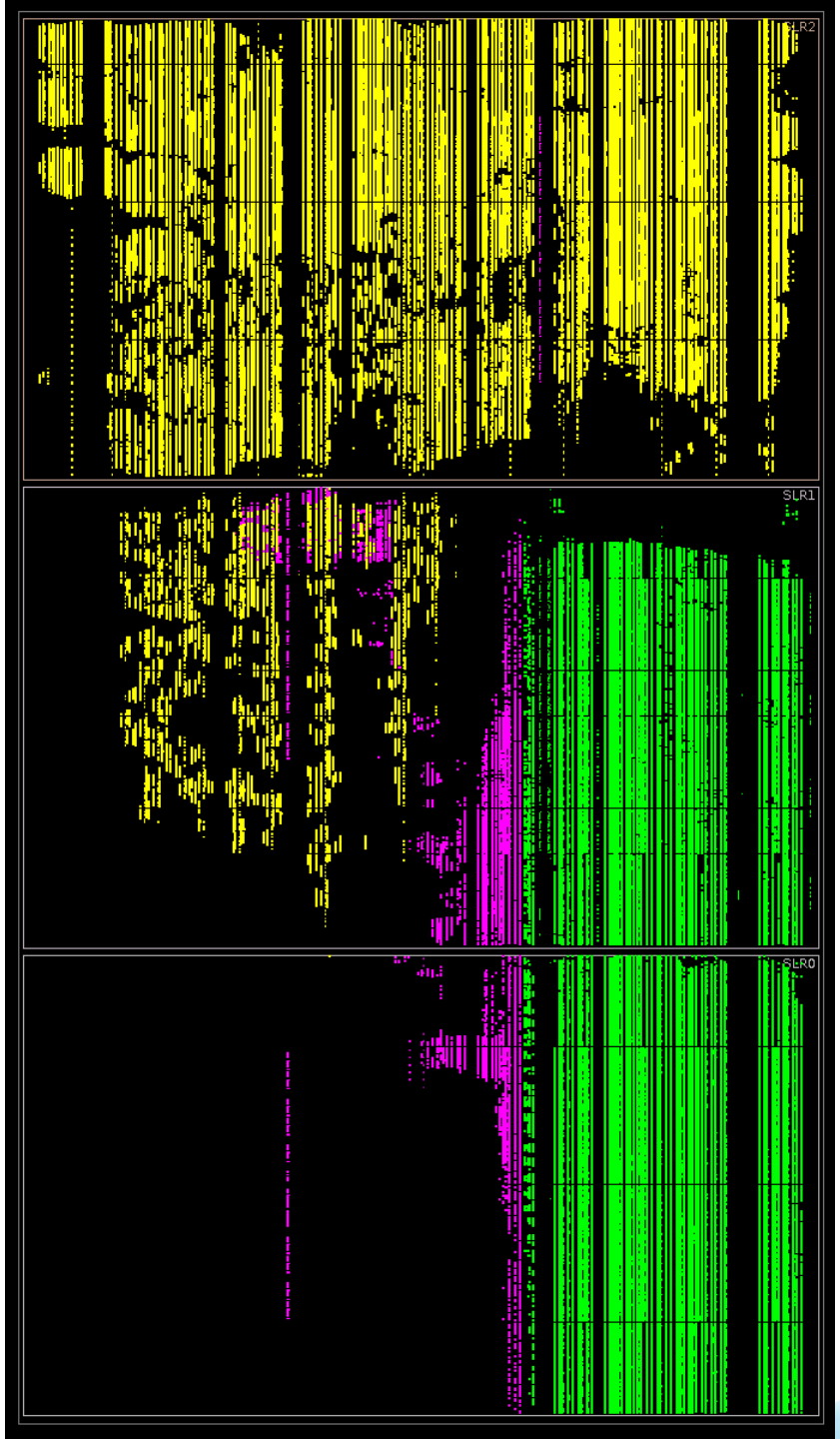


CLB Utilization per SLR

82%

85%

58%



72%

45%

37%

modsqr

AXI & interface

SDAccel logic

GGG Round 2 design

Round 2 Competition Results

Q?	Team Name	Directory	Expected	Actual
YES	Eric Pearson	eric_pearson-1	27 ns/sq	27 ns/sq
YES	Eric Pearson	eric_pearson-2	25.2 ns/sq	25.2 ns/sq
YES	Geriatric Guys with Gates	geriatric_guys_with_gates	26.4 ns/sq	26.4 ns/sq
NO	xjtu_asic	xjtu_asic	30.8 ns/sq	DRC check reported illegal clocks
NO	Ruslan	ruslan	28.4 ns/sq	Did not meet timing in synthesis
NO	Tulpar	tulpar	various	Not a finished design but promising projections from high level synthesis

Round 2 Competition Results

- Again, GGG was one of only two teams with a working design
- Once again, we came in second to Eric Pearson!

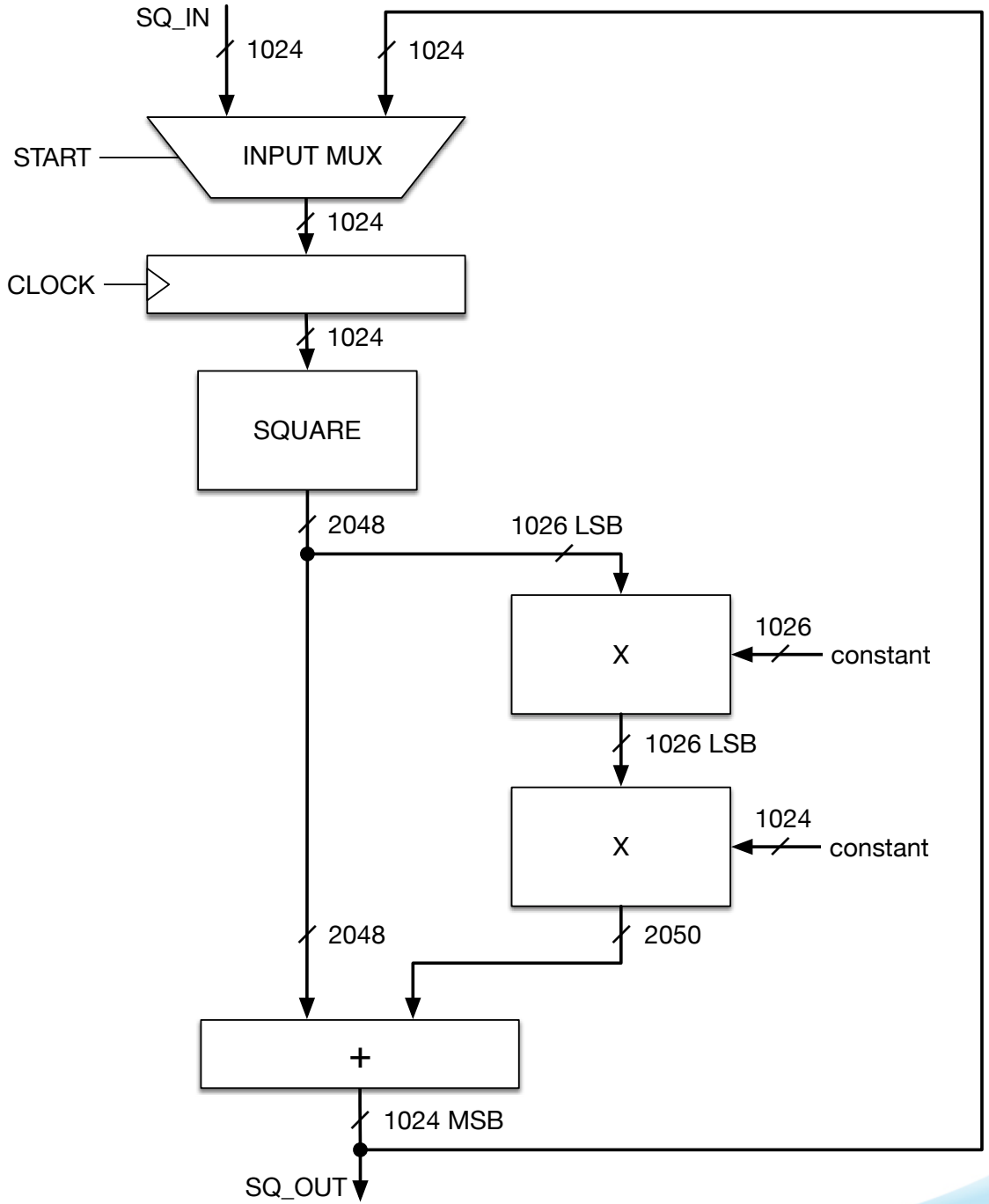
Round 3

- Official name: “Competition #2 Implementation Contest”
- ~13 weeks : Oct 30, 2019 – Jan 30, 2020

  …but if you were busy with Round 2, it’s really only 4 weeks
- Prize : $5,000
- “…awarded to the fastest VDF implementation on FPGA that uses an alternative approach than the ‘Ozturk’ algorithm.”
- Potential algorithmic approaches include
  - Montgomery
  - Barrett
  - Chinese Remainder Theorem

Team GGG in Round 3

- Montgomery reduction
  - Square and two multiplies
  - No modulus lookup!
- “Iterative” implementation
  - one multiplier (actually, a MAC)
  - used over 3 clock periods
- “3-phase” implementation
  - flow-through design
  - dedicated hardware for each operator
  - should be faster…



Round 3 Competition Results

Team Name	Directory	Expected	Actual
Ben Devlin	devlin0	60 ns/sq	60 ns/sq
Ben Devlin	devlin1	49.2 ns/sq	49.2 ns/sq
Ben Devlin	devlin2	46 ns/sq	46 ns/sq
Geriatric Guys with Gates: iterative	ggg	60.2 ns/sq	60.2 ns/sq
Geriatric Guys with Gates: 3-phase	ggg	55.5 ns/sq	F1 fails

Round 3 Competition Results

- Again, GGG was one of only two teams with a working design
- Once again, we came in second! (at least it was not Eric Pearson)

Power

	Total LUT	modsqr Latency	Reported Power	Measured F1 Power	Build Time
		ns/sq	W	W	hh:mm
Ozturk baseline	274,737	49.6	42.8		4:24
GGG Round 1	271,721	44.8	43.7		7:12
Eric Pearson Round 1	624,773	28.6	42.5		14:08
GGG Round 2	412,891	25.6	42.7	~22	9:59
GGG Round 3 iter	289,519	60.2	45.7	~22	3:54
GGG Round 3-phase	479,042	55.5	43.2	fails	15:04

Future ASIC Competition

- VDF Alliance plans to hold an ASIC competition in late 2020
- 2048-bit design
- Differences between FPGA and ASIC
  - Algorithm and architecture for FPGA may not apply to ASIC
  - ASIC performance should be far better than FPGA
- Difficult to administer ASIC competition
  - How to score? Cannot build actual hardware
  - Need access to expensive tools
  - Need access to proprietary libraries
  - Perhaps use STA results from placement-aware synthesis tool?

Q & A

Thank You