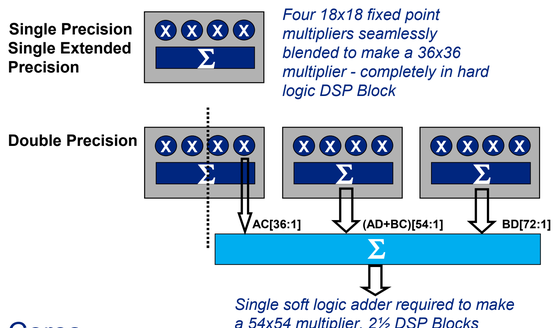
Altera FPGAs deliver:

- A powerful mix of fixed and floating point performance
- Extensive hard DSP capabilities
  - IEEE754 single and double precision specifically supported
    - 100s 36x36 multipliers
    - ~100 54x54 multipliers
- Superior computational density per Watt than other solutions
  - In a recent National Science Foundation benchmark, a Stratix® IV FPGA delivered 171 GFLOPS, and was the clear overall leader in highest GFLOPs/Watt.
- Sustained peak performance

Devices

- Floating Point density largely determined by hard multiplier density
  - Altera Multipliers efficiently support floating point mantissa sizes

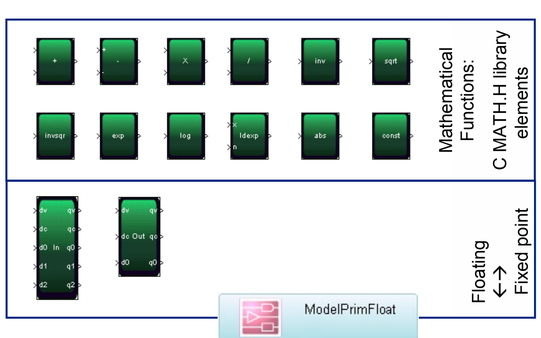


Four 18x18 fixed point multipliers seamlessly blended to make a 36x36 multiplier - completely in hard logic DSP Block

Single soft logic adder required to make a 54x54 multiplier, 2½ DSP Blocks

Cores

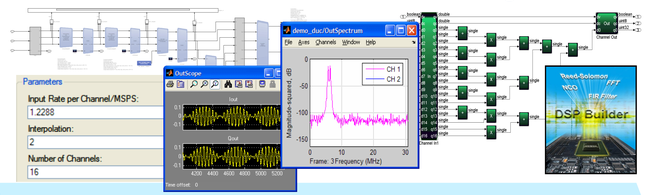
- Optimized 'MATH.H' function library
  - Multiplier-based algorithms give low latency, low power, high performance and consistent results
  - EXP, LOG, SQRT, INVSQRT, SIN, COS available now, others in development
- Available as
  - Altera Floating point MegaFunctions
  - Floating Point block library in DSP Builder model-based design tool
    - Currently: separate blocks – α demonstrator
    - Future: Polymorphic with fixed point equivalents
- Also higher-level IP: e.g. Matrix Inversion, Matrix Multiply



Tools: Floating Point Compiler (FPC)

- Conventional Wisdom : IEEE754 system level design too complex for FPGAs
  - Floating Point core based design requires more soft logic than a fixed point FPGA supports
- But IEEE754 data-path inefficient: functional redundancy between operators
  - Required for processors – unlimited operation combinations
    - Arithmetic unit requires all inputs and outputs to be of a known format
  - Not required for data-path – *a priori* knowledge of inter-operator relationships
    - Data-path unit requires all inputs and outputs to be IEEE754 format
    - Internal format only has to guarantee that casting to and from IEEE754 is correct
- Build an FPC tool that exploits *a priori* knowledge of inter-operator relationships & has freedom to apply data-path level optimizations

Tools: DSP Builder Advanced / FPC Integration



Effortless FPGA Implementation

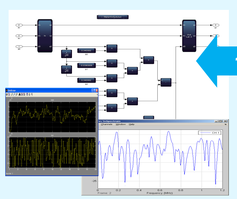
Fixed Point domain:
- Automatic pipelining to meet required Fmax
  - Similar performance as optimized HDL
  - Easy timing closure - fewer iterations

Floating Point domain:
- Apply Floating Point Compiler fused data-path optimizations

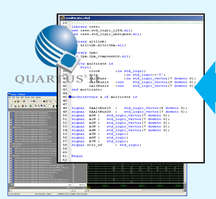
Fast Design Space Exploration

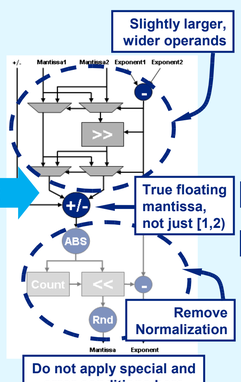
- Integrated design, simulation and generation
- Fast multi-channel design implementation
- Automatic generation of control plane logic
- Efficient pipelining for multi-channel data paths
- Driven by system level parameters
- Effortless FPGA device family retargeting

Floating Point Model-Based Design

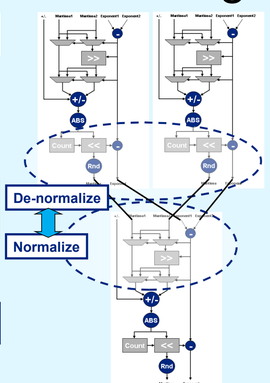


1. Develop & design schematically with primitive, math.h & core functions; in fixed and floating point domains
2. Tool applies global data-path optimizations to floating point domains and many fixed-point domain optimizations
3. Tool generates integrated and optimized high-performance HDL on each simulation





FPC Adder/Subtractor core implementation



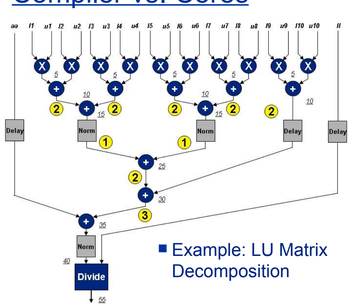
FPC Inter-operator redundancy

- Set up and solve linear programming problems for scheduling and optimal pipelining for target f_{max}
- Optimize adder trees
- Optimize DSP block use
- Optimize memory block use where access is scheduled (e.g. FIRs)
- Swap arithmetic expressions for identical, lower resource equivalents
- Duplicate code removal
- Time-Division Multiplexing
- Threshold trade-offs
- CSD constant multipliers
- …

Fixed point domain optimizations (currently separate)

Fused data-path optimizations typically achieve 50% reduction in soft logic & latency - allowing 100% utilization of a device's floating point capability at fixed-point speeds

Compiler vs. Cores



- Example: LU Matrix Decomposition

- Abstracted data-path design in DSPB
- DSP Builder floating point blocks mapped to FPC blocks
- FPC restructures data-path to avoid overflows and balances it
- FPC optimizations applied independently of DSP Builder fixed-point optimizations

- Compiled data-path is about 50% the size of the equivalent core-based design
  - DSP resources same
- Latency also 50%
- Corresponding power reduction
  - most of the data-path dynamic power consumption in soft logic, not multipliers
- Allows 100% of a device's floating point capability to be used at still run at 250MHz

- Vector Logic:
  - *compiled data-path*
- Logic:
  - *compiled data-path + application*
- Core Logic:
  - *equivalent data-path constructed from discrete cores*

Matrix Size	Logic	DSP	Vector Logic	Core Logic
12x12	5197 (sp) / 8652 (dp)	75	4587 (sp) / 7855 (dp)	7800 (sp) / 9000 (dp)
64x64	21457 (sp) / 27346 (dp)	283	20681 (sp) / 26004 (dp)	41600 (sp) / 48000 (dp)

Examples: Floating Point IP using FPC

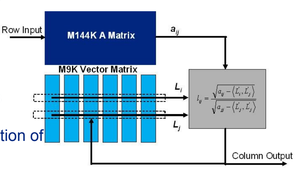
Parameterizable Cholesky Decomposition

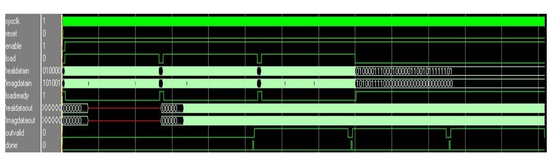
- 100% multiplier usage with fraction of logic usage
- Can fill the device with floating point operations and still achieve pushbutton fit
- 300 ALUT / 400 register per operator pair
  - Less than half of core methodology

Matrix Size	Core	ALUT	Register	18x18
8x8	Vector	2400	5800	32
	Root	500	1800	18
	Total	2900	7600	50
32x32	Vector	9000	20000	128
	Root	500	1800	18
	Total	9500	21800	146
64x64	Vector	21000	29000	256
	Root	500	1800	18
	Total	21500	30800	274
128x128	Vector	40000	56000	512
	Root	500	1800	18
	Total	40500	57800	530

4x4 High Speed Cholesky

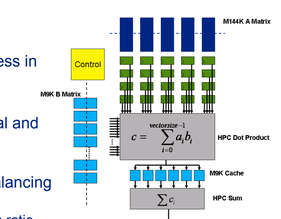
- Design Brief: High performance, low cost
  - 15M Matrixes/s
- Result:
  - 20M Matrixes/s SIII
  - 5K ALUTs, 70 18x18 multipliers
  - Not only faster/cheaper than processor alternative, but a fraction of the power consumption





Matrix Multiplier Core

- Feed forward architecture
  - Rows and columns blocked
  - Partial results cached and process in secondary pipe
- Extensible and parameterizable
  - Single and double precision, real and complex numbers
  - Matrix dimensions
  - Area, performance, resource balancing
    - Memory depth and bandwidth
    - Dot product to matrix dimension ratio



Matrix Operators 3–7 GFLOPs/Watt—Single Precision

Matrix multiply core examples	Vector size	Logic usage					GFLOPS	Performance (Stratix® IV FPGA)	Power (mW)			
		Adaptive logic modules (ALMs)	18x18 mults	M9K	M144K	Memory (Kbits)			Static	Dynamic	I/O	Total
(36x112)x(112x36)	8	4,604	32	43	2	576	4	291 MHz	2,008	1,063	300	3,334
(36x224)x(224x36)	16	7,882	64	77	4	1,102	9	291 MHz	2,045	1,821	300	4,165
(36x448)x(448x36)	32	14,257	128	137	8	2,153	18	291 MHz	2,110	3,448	300	5,858
(64x64)x(64x64)	32	13,154	128	41	8	1,333	18	292 MHz	2,112	2,604	306	5,023
(128x128)x(128x128)	64	25,636	256	141	16	3,173	37	293 MHz	2,244	5,384	306	7,934