# *Extracting low-precision floating-point adders from embedded hard FP DSP Blocks on FPGAs*

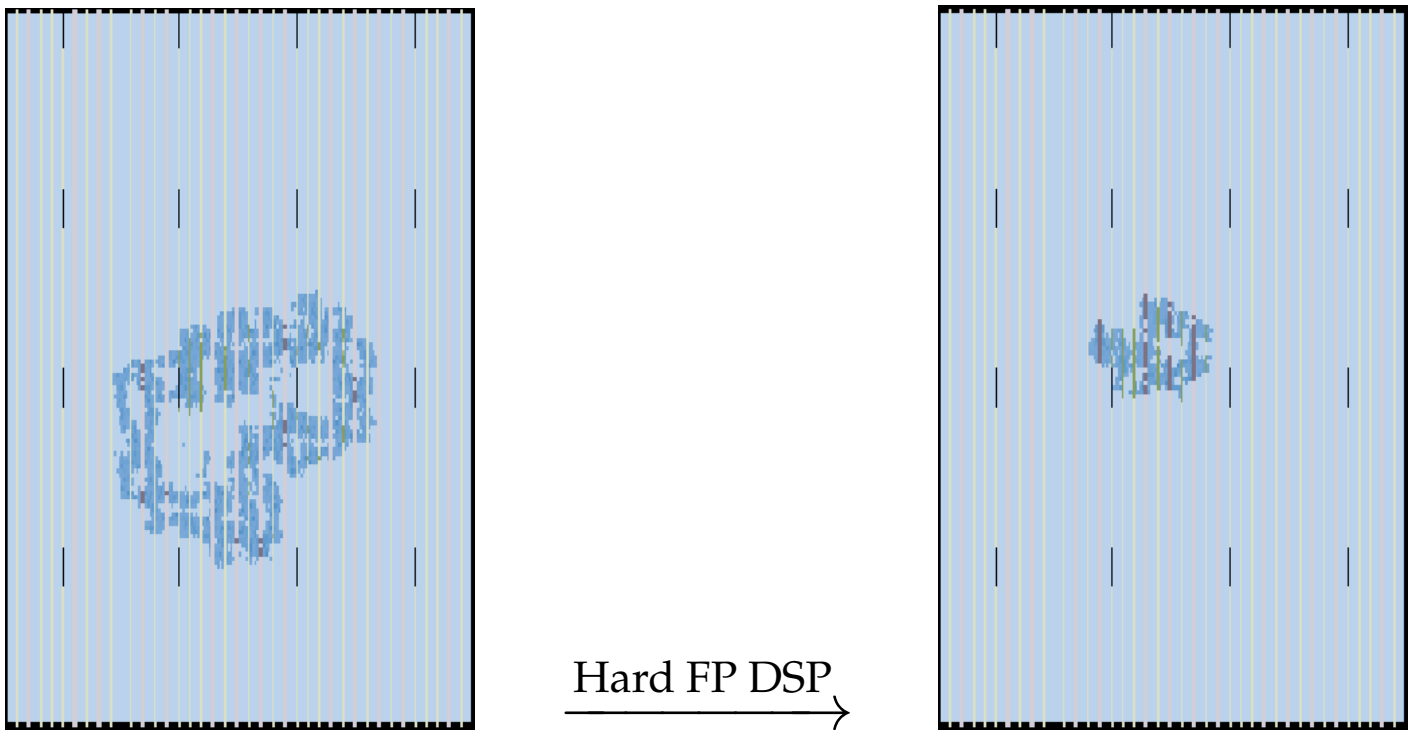
Bogdan Pasca, Martin Langhammer

Intel Corporation

ARITH 2023
4-6 September, 2023
Portland, USA

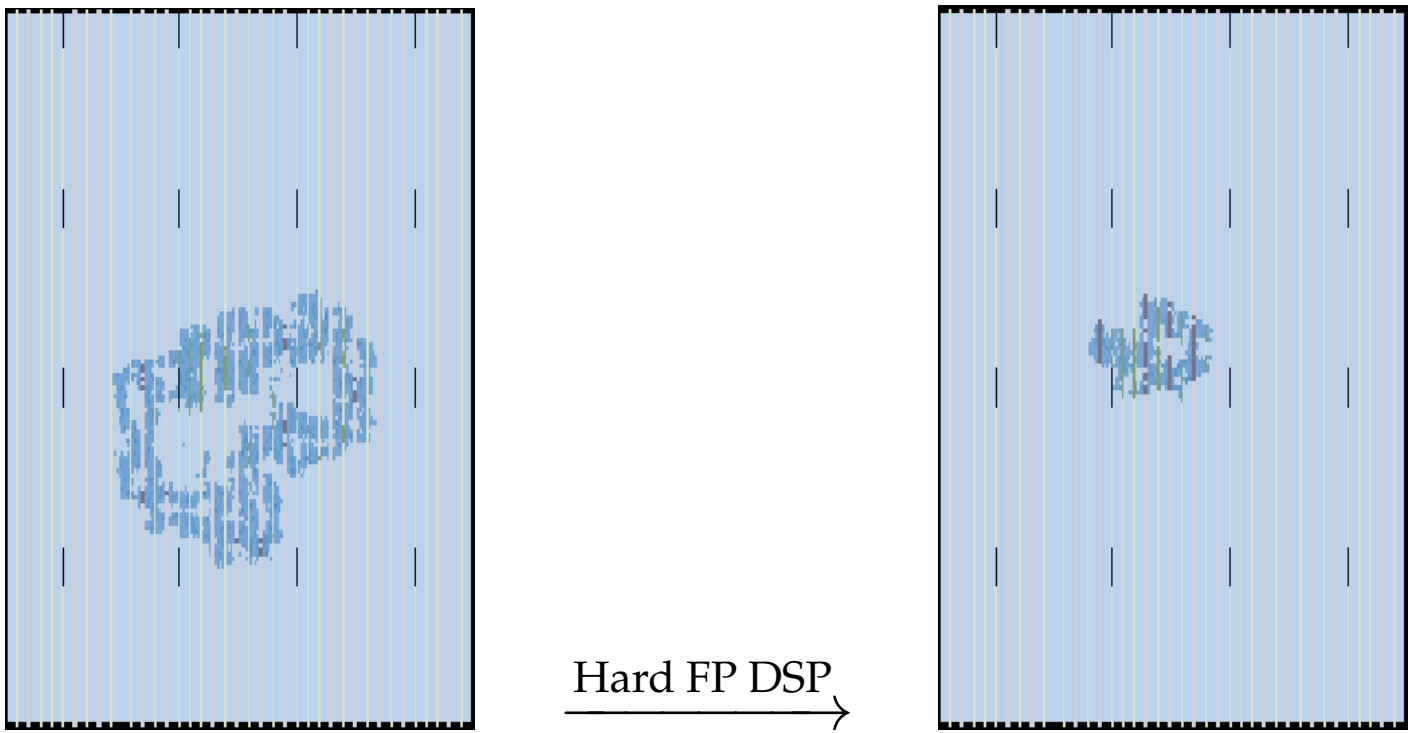
# Context

Contemporary FPGAs embed DSP Blocks with new FP functionality



# Context

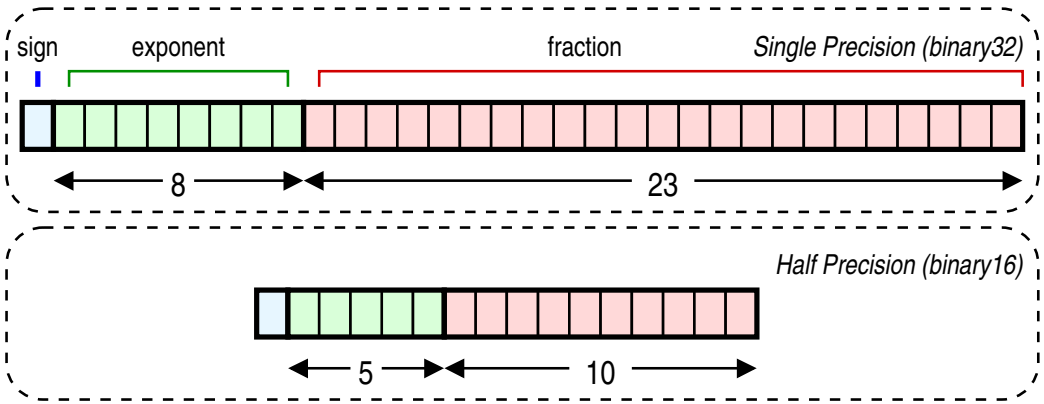
Contemporary FPGAs embed DSP Blocks with new FP functionality




How to use this functionality for low-precision FP adders?

# Background

## IEEE-754 formats single (32-bit) and half (16-bit) precision

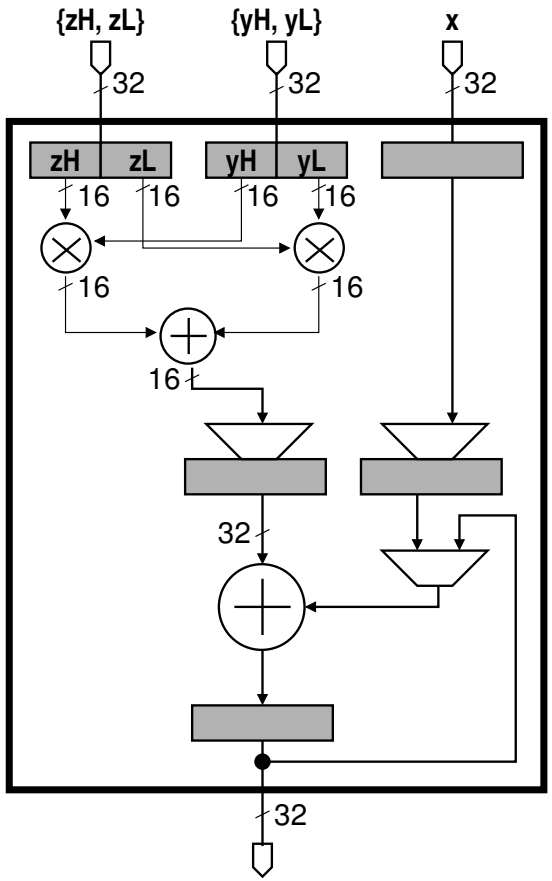



## Exception Encoding

| Encoded Value | Exponent | Fraction |
|---|---|---|
| Zero (flush mode) | 0 | Any |
| Zero (subnormals) | 0 | 0 |
| Subnormal | 0 | $\neq 0$ |
| Regular | 00..01 $\rightarrow$ 11..10 | Any |
| Infinity | 11..11 | 0 |
| NaN | 11..11 | $\neq 0$ |

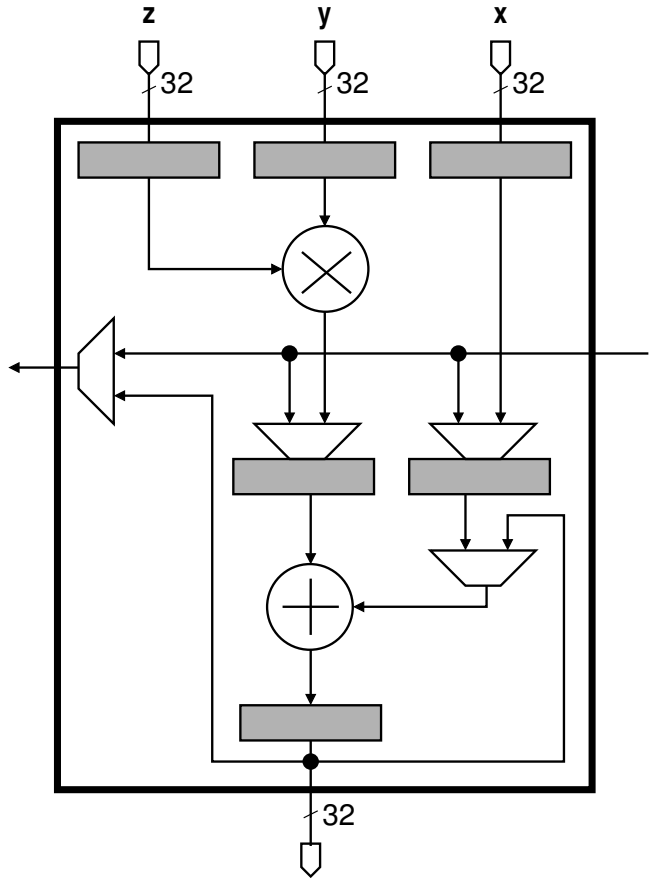
# Background

Agilex DSP Block
in low-precision FP mode

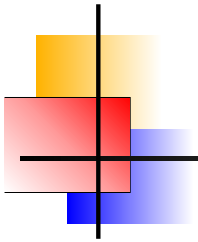



$$D = (yH \cdot zH + yL \cdot zL) + x$$

SP DSP Block
Agilex, Stratix 10, Arria 10




$$D = (y \cdot z) + x$$



# HP FP Add Architecture - Agilex

- Goal:

$$\boxed{S = (a + b)}$$

with $a$, $b$, and $S$ are HP values.

# HP FP Add Architecture - Agilex

- Goal:

$$\boxed{S = (a + b)}$$

with $a$, $b$, and $S$ are HP values.

- How?

Agilex DSP Block in `fp16multadd_sum` flushed mode.
$D = (yH \cdot zH + yL \cdot zL) + x$

$yY$, $yL$, $zH$ and $zL$ are all HP values,
$x$ and $D$ are in SP.

# HP FP Add Architecture - Agilex

- Goal:

$$\boxed{S = (a + b)}$$

with $a$, $b$, and $S$ are HP values.

- How?

Agilex DSP Block in `fp16multadd_sum` flushed mode.
$D = (yH \cdot zH + yL \cdot zL) + x$

$yY$, $yL$, $zH$ and $zL$ are all HP values,
$x$ and $D$ are in SP.

- Use the mapping:

$yH = a$, $zH = +1$ (HP)
$yL = b$, $zL = +1$ (HP)
$x = -0$ (SP).

# HP FP Add Architecture - Agilex

- Goal:

$$\boxed{S = (a + b)}$$

with $a$, $b$, and $S$ are HP values.

- How?

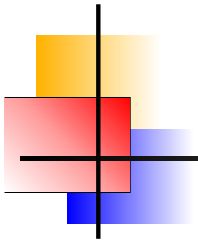
Agilex DSP Block in `fp16multadd_sum` flushed mode.
$D = (yH \cdot zH + yL \cdot zL) + x$

$yY$, $yL$, $zH$ and $zL$ are all HP values,
$x$ and $D$ are in SP.

- Use the mapping:

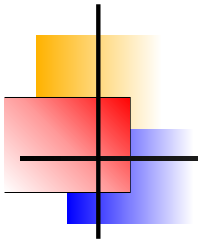
$yH = a$, $zH = +1$ (HP)
$yL = b$, $zL = +1$ (HP)
$x = -0$ (SP).

- Subtlety that $x = -0$ allows maintaining the correct sign for 0.



# HP FP Add Architecture - Agilex

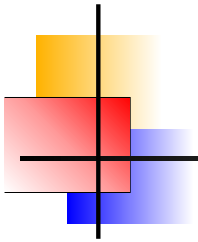
- *D* is now a SP value containing a 10-bit populated fraction.
- fraction extracted directly: *D*[22:13].



# HP FP Add Architecture - Agilex

- *D* is now a SP value containing a 10-bit populated fraction.
- fraction extracted directly: *D*[22:13].

- biased 8-bit exponent $e_D^b$[30:23] $\xrightarrow{convert}$ a 5-bit HP exponent.



# HP FP Add Architecture - Agilex

- *D* is now a SP value containing a 10-bit populated fraction.
- fraction extracted directly: *D*[22:13].

- biased 8-bit exponent $e_D^b$[30:23] $\xrightarrow{convert}$ a 5-bit HP exponent.

- How?
  bias difference between SP and HP is 112:

$$\begin{aligned} e_S^b &= (e_D^b - 127) + 15 \\ &= e_D^b - 112. \end{aligned}$$

# HP FP Add Architecture - Agilex

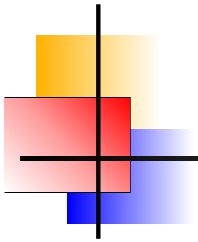
- *D* is now a SP value containing a 10-bit populated fraction.
- fraction extracted directly: *D*[22:13].

- biased 8-bit exponent $e_D^b$[30:23] $\xrightarrow{convert}$ a 5-bit HP exponent.

- How?
  bias difference between SP and HP is 112:

$$\begin{aligned} e_S^b &= (e_D^b - 127) + 15 \\ &= e_D^b - 112. \end{aligned}$$

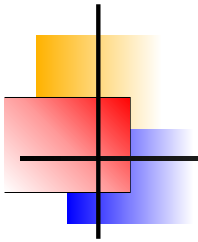
What about special cases?



# HP FP Add Architecture - Agilex

Check the effect of exponent subtraction on exception case encodings:
For **Infinity** and **NaN** $e_D^b = 255$:

$$\begin{aligned} e_S^b &= 255 - 112 = 143 \\ &= 10001111(\text{in binary encoding}). \end{aligned}$$



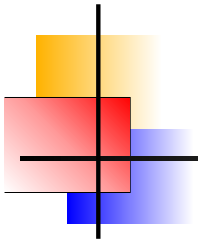
# HP FP Add Architecture - Agilex

Check the effect of exponent subtraction on exception case encodings:
For **Infinity** and **NaN** $e_D^b = 255$:

$$\begin{aligned} e_S^b &= 255 - 112 = 143 \\ &= 10001111(\text{in binary encoding}). \end{aligned}$$

For **Zero** (FTZ) $e_D^b = 0$:

$$\begin{aligned} exp_S^b &= 0 - 112 = -112 \\ &= 10010000(\text{in binary encoding}). \end{aligned}$$



# HP FP Add Architecture - Agilex

Check the effect of exponent subtraction on exception case encodings:
For **Infinity** and **NaN** $e_D^b = 255$:

$$\begin{aligned} e_S^b &= 255 - 112 = 143 \\ &= 10001111(\text{in binary encoding}). \end{aligned}$$

For **Zero** (FTZ) $e_D^b = 0$:

$$\begin{aligned} exp_S^b &= 0 - 112 = -112 \\ &= 10010000(\text{in binary encoding}). \end{aligned}$$

Lower 4-bits match desired encoding, bit 5 needs inverting.

# HP FP Add Architecture - Agilex

| $e_D^b$ | $e_D^u$ | Binary | Class | $e_D^b$-112 | Binary | Goal |
|---|---|---|---|---|---|---|
| 255 | - | 1111 1111 | Inf/NaN | 143 | 1 000 1111 | 1 1111 |
| 142 | 15 | 1000 1110 | Regular | 30 | 0 001 1110 | 1 1110 |
| 141 | 14 | 1000 1101 | Regular | 29 | 0 001 1101 | 1 1101 |
| ... | | | | | | |
| 128 | 1 | 1000 0000 | Regular | 16 | 0 001 0000 | 1 0000 |
| 127 | 0 | 0111 1111 | Regular | 15 | 0 000 1111 | 0 1111 |
| 126 | -1 | 0111 1110 | Regular | 14 | 0 000 1110 | 0 1110 |
| ... | | | | | | |
| 114 | -13 | 0111 0010 | Regular | 2 | 0 000 0010 | 0 0010 |
| 113 | -14 | 0111 0001 | Regular | 1 | 0 000 0001 | 0 0001 |
| 0 | - | 0000 0000 | Zero | -112 | 1 001 0000 | 0 0000 |

# HP FP Add Architecture - Agilex

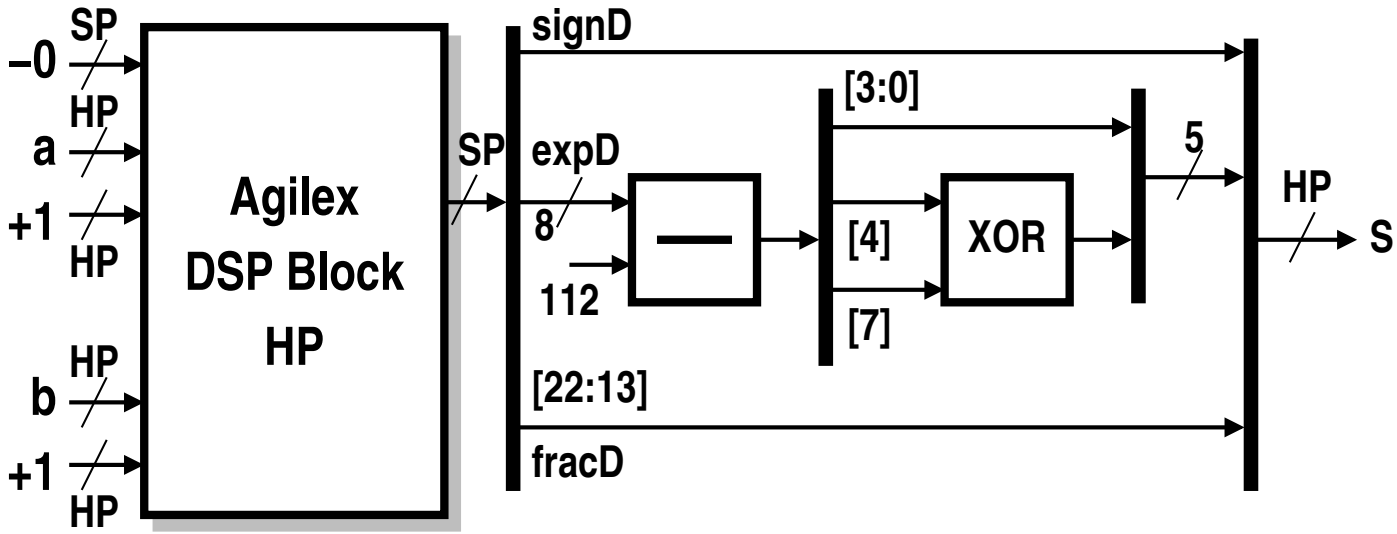
| $e_D^b$ | $e_D^u$ | Binary | Class | $e_D^b$-112 | Binary | Goal |
|---|---|---|---|---|---|---|
| 255 | - | 1111 1111 | Inf/NaN | 143 | 1 000 1111 | 1 1111 |
| 142 | 15 | 1000 1110 | Regular | 30 | 0 001 1110 | 1 1110 |
| 141 | 14 | 1000 1101 | Regular | 29 | 0 001 1101 | 1 1101 |
| ... | | | | | | |
| 128 | 1 | 1000 0000 | Regular | 16 | 0 001 0000 | 1 0000 |
| 127 | 0 | 0111 1111 | Regular | 15 | 0 000 1111 | 0 1111 |
| 126 | -1 | 0111 1110 | Regular | 14 | 0 000 1110 | 0 1110 |
| ... | | | | | | |
| 114 | -13 | 0111 0010 | Regular | 2 | 0 000 0010 | 0 0010 |
| 113 | -14 | 0111 0001 | Regular | 1 | 0 000 0001 | 0 0001 |
| 0 | - | 0000 0000 | Zero | -112 | 1 001 0000 | 0 0000 |

# HP FP Add Architecture - Agilex

| $e_D^b$ | $e_D^u$ | Binary | Class | $e_D^b$-112 | Binary | Goal |
|---|---|---|---|---|---|---|
| 255 | - | 1111 1111 | Inf/NaN | 143 | 1 000 1111 | 1 1111 |
| 142 | 15 | 1000 1110 | Regular | 30 | 0 001 1110 | 1 1110 |
| 141 | 14 | 1000 1101 | Regular | 29 | 0 001 1101 | 1 1101 |
| ... | | | | | | |
| 128 | 1 | 1000 0000 | Regular | 16 | 0 001 0000 | 1 0000 |
| 127 | 0 | 0111 1111 | Regular | 15 | 0 000 1111 | 0 1111 |
| 126 | -1 | 0111 1110 | Regular | 14 | 0 000 1110 | 0 1110 |
| ... | | | | | | |
| 114 | -13 | 0111 0010 | Regular | 2 | 0 000 0010 | 0 0010 |
| 113 | -14 | 0111 0001 | Regular | 1 | 0 000 0001 | 0 0001 |
| 0 | - | 0000 0000 | Zero | -112 | 1 001 0000 | 0 0000 |

# HP FP Add Architecture - Agilex

Correctly rounded HP implementation based on the Agilex DSP Block



# Alternative HP FP Add Architecture - Agilex

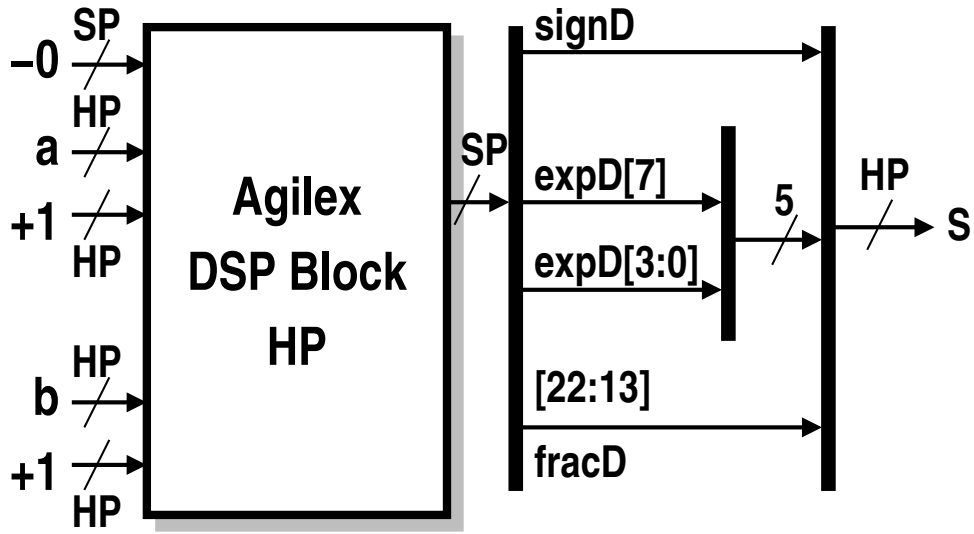
| $e_D^b$ | $e_D^u$ | Binary | Class | $e_D^b$-112 | Binary | Goal |
|---|---|---|---|---|---|---|
| 255 | - | 1111 1111 | Inf/NaN | 143 | 1000 1111 | 11111 |
| 142 | 15 | 1000 1110 | Regular | 30 | 0001 1110 | 11110 |
| 141 | 14 | 1000 1101 | Regular | 29 | 0001 1101 | 11101 |
| ... | | | | | | |
| 128 | 1 | 1000 0000 | Regular | 16 | 0001 0000 | 10000 |
| 127 | 0 | 0111 1111 | Regular | 15 | 0000 1111 | 01111 |
| 126 | -1 | 0111 1110 | Regular | 14 | 0000 1110 | 01110 |
| ... | | | | | | |
| 114 | -13 | 0111 0010 | Regular | 2 | 0000 0010 | 00010 |
| 113 | -14 | 0111 0001 | Regular | 1 | 0000 0001 | 00001 |
| 0 | - | 0000 0000 | Zero | -112 | 1001 0000 | 00000 |

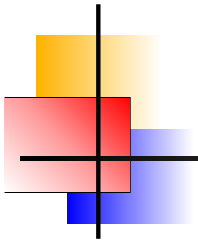
# Alternative HP FP Add Architecture - Agilex

| $e_D^b$ | $e_D^u$ | Binary | Class | $e_D^b$-112 | Binary | Goal |
|---|---|---|---|---|---|---|
| 255 | - | 1111 1111 | Inf/NaN | 143 | 1000 1111 | 11111 |
| 142 | 15 | 1000 1110 | Regular | 30 | 0001 1110 | 11110 |
| 141 | 14 | 1000 1101 | Regular | 29 | 0001 1101 | 11101 |
| ... | | | | | | |
| 128 | 1 | 1000 0000 | Regular | 16 | 0001 0000 | 10000 |
| 127 | 0 | 0111 1111 | Regular | 15 | 0000 1111 | 01111 |
| 126 | -1 | 0111 1110 | Regular | 14 | 0000 1110 | 01110 |
| ... | | | | | | |
| 114 | -13 | 0111 0010 | Regular | 2 | 0000 0010 | 00010 |
| 113 | -14 | 0111 0001 | Regular | 1 | 0000 0001 | 00001 |
| 0 | - | 0000 0000 | Zero | -112 | 1001 0000 | 00000 |

# Alternative HP FP Add Architecture - Agilex

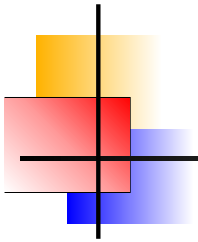
Correctly rounded HP implementation based on the Agilex DSP Block





# HP FP Add Architecture - SP DSP Block Mapping

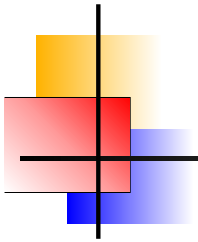
- Use SP DSP Block present in Arria 10, Stratix 10 or Agilex
- First step: HP $\xrightarrow{convert}$ SP
  - fraction: right padding with 13 zeros
  - exponent: 5-bit $\rightarrow$ 8-bit can be done via table lookup



# HP FP Add Architecture - SP DSP Block Mapping

- Use SP DSP Block present in Arria 10, Stratix 10 or Agilex
- First step: HP $\xrightarrow{convert}$ SP
  - fraction: right padding with 13 zeros
  - exponent: 5-bit $\rightarrow$ 8-bit can be done via table lookup

- Use custom conversion, $x_{\mathrm{HP}} \neq x_{\mathrm{SP}}$
- Benefit from exception handling of SP FP Adder



# HP FP Add Using SP DSP Blocks - Custom mapping

$$LUT1[0] = 0;$$
$$LUT1[i + 15] = i + 255 - 16; i \in \{-14, 16\}$$

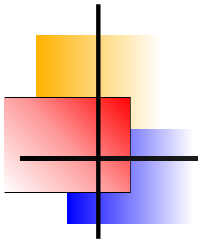
- preserve mapping for 0
- regular values map $x_{\mathrm{SP}} = 2^{112} x_{\mathrm{HP}}$
- NaN and Inf exponent encoding is preserved

# HP FP Add Using SP DSP Blocks - Custom mapping

$$LUT1[0] = 0;$$
$$LUT1[i + 15] = i + 255 - 16; i \in \{-14, 16\}$$

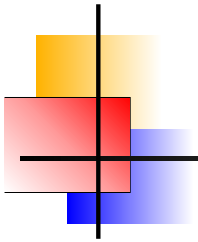
- preserve mapping for 0
- regular values map $x_{\mathrm{SP}} = 2^{112} x_{\mathrm{HP}}$
- NaN and Inf exponent encoding is preserved

Reliably recover output Inf and NaN conditions



# HP FP Add Using SP DSP Blocks - Correct Rounding

- correct rounding for RNE on 8-bit exponents and 10-bit fractions
- generically the fraction can be anywhere up to 11 bit
- simply round the 23-bit SP fraction to the 10-bit HP fraction

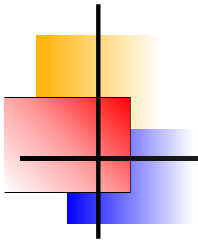


# HP FP Add Using SP DSP Blocks - Correct Rounding

- correct rounding for RNE on 8-bit exponents and 10-bit fractions
- generically the fraction can be anywhere up to 11 bit
- simply round the 23-bit SP fraction to the 10-bit HP fraction

Why does this work? What about double-rounding?

- If no SP rounding occurs $\rightarrow$ easy to prove
- SP rounding has occurred when exponent difference $\geq 14$
- HP fraction is only 10 bits wide
- Rounded result far from HP midpoint.



# HP FP Add Using SP DSP Blocks - Exponent Recovery

- perform the ’reverse’ mapping to the HP format.
- subtract 224 = 255 - 31 from the computed 8-bit exponent
- handle special cases separately

# HP FP Add Using SP DSP Blocks - Exponent Recovery

- perform the 'reverse' mapping to the HP format.
- subtract 224 = 255 - 31 from the computed 8-bit exponent
- handle special cases separately

- another solution is tabulation-based
- tabulation would require 8-bit table inputs $\rightarrow$ inefficient
- minimum 8-bit exponent 224-10=214 (smallest subnormal).
- observe binary exponent pattern

| Exponent Value | Binary Encoding |
|---|---|
| 255 | 1111 1111 |
| ... | 111x xxxx |
| 224 | 1110 0000 |
| 223 | 1101 1111 |
| ... | 1101 xxxx |
| 215 | 1101 0111 |
| 214 | 1101 0110 (smallest denormal, half) |
| 0 | 0000 0000 |

# HP FP Add Using SP DSP Blocks - Exponent Recovery

- perform the 'reverse' mapping to the HP format.
- subtract 224 = 255 - 31 from the computed 8-bit exponent
- handle special cases separately

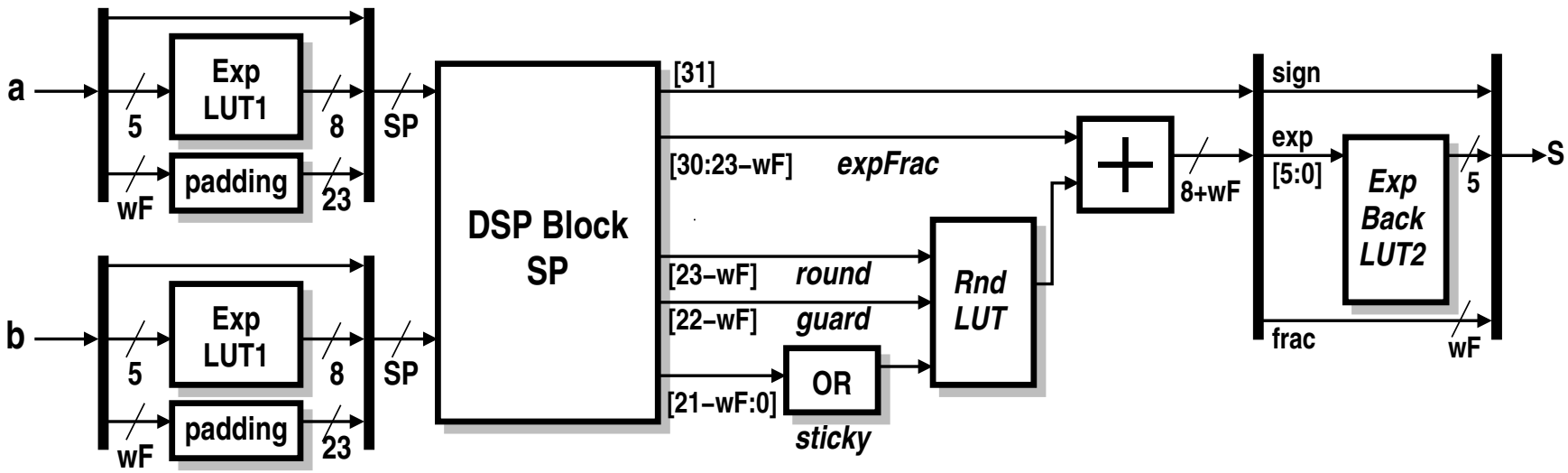
- another solution is tabulation-based
- tabulation would require 8-bit table inputs $\rightarrow$ inefficient
- minimum 8-bit exponent 224-10=214 (smallest subnormal).
- observe binary exponent pattern

| Exponent Value | Binary Encoding |
|---|---|
| 255 | 1111 1111 |
| ... | 111x xxxx |
| 224 | 1110 0000 |
| 223 | 1101 1111 |
| ... | 1101 xxxx |
| 215 | 1101 0111 |
| 214 | 1101 0110 (smallest denormal, half) |
| 0 | 0000 0000 |

Reverse exponent mapping can be performed with only 6 bits

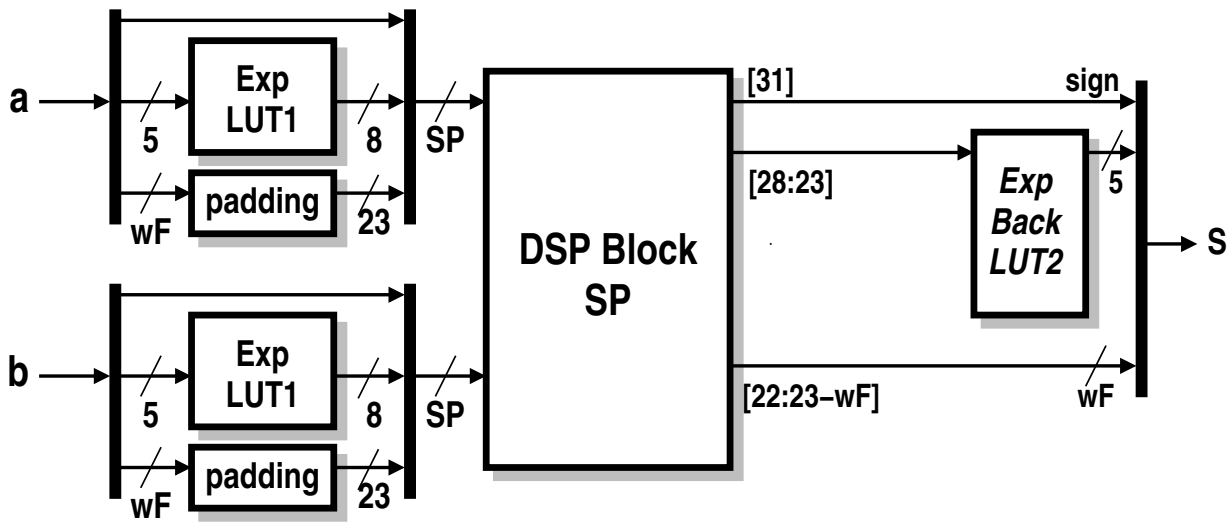
# HP FP Add Architecture - SP DSP Block Mapping

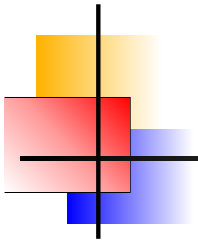
Correctly rounded HP implementation using the SP Hard DSP Block



# HP FP Add Using SP DSP Blocks - Faithful rounding

- faithful rounding can be obtained by means of truncation.
- fraction is directly truncated to 10 bits.

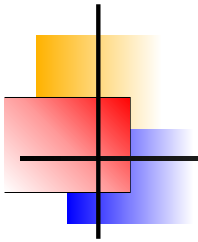




# Results

- logic implementation → Quartus 22.4 fp_functions Megacore.
- generated for a target frequency of 500 MHz.
- synthesis on the fastest speedgrades.

| Arch | Target | Latency | ALMs | DSPs | Ratio |
|---|---|---|---|---|---|
| Proposed-A1 | Agilex | 5 | 5 | 1 | 147 |
| Proposed-A2 | Agilex | 5 | 0 | 1 | 152 |
| Logic | 500MHz, -1 | 11 | 152 | 0 | - |
| Proposed-B | Statix10 | 6 | 26 | 1 | 174 |
| Logic | 500MHz, -1 | 16 | 200 | 0 | - |

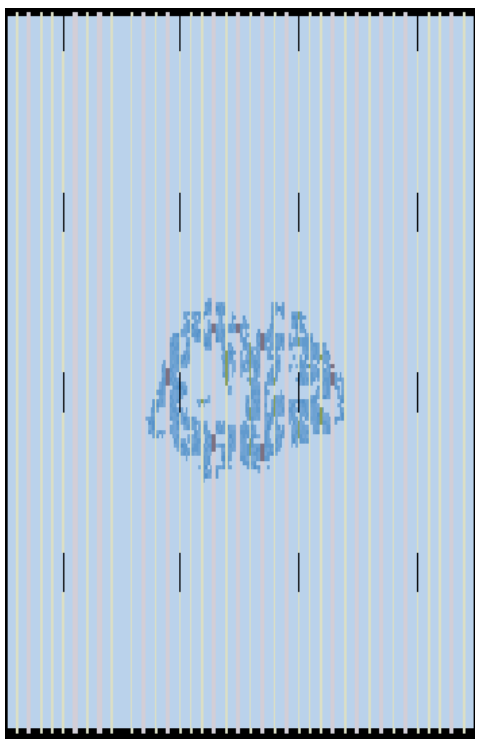


# System Level Impact

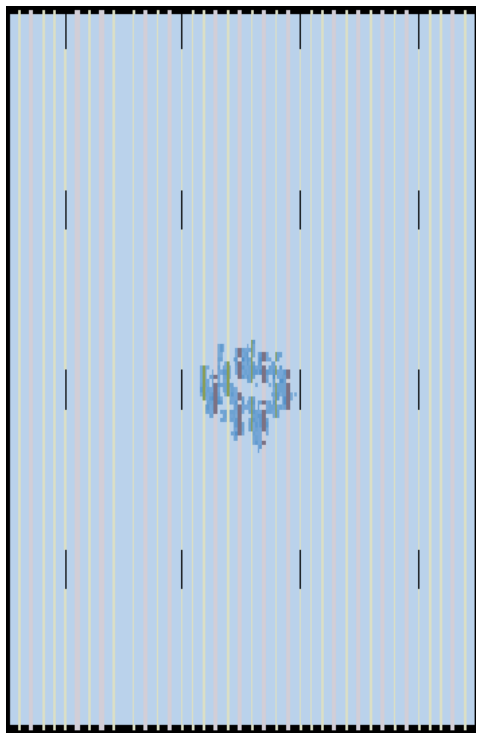
- 8K-point FP FFT design from DSP Builder Advanced
- push-button resource tradeoff using new adder architectures
- target is an Agilex FPGA
- resource utilization: 64 ALMs reduction, 32 DSPs increase
- ratio: 185 ALMs / DSP

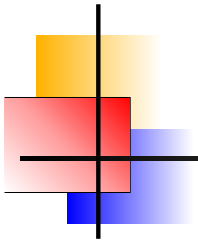
| Architecture | ALMs | M20K | DSPs |
|---|---|---|---|
| Old | 8116 | 62 | 12 |
| Proposed | 2183 | 46 | 44 |

# System Level Impact - Chip-Planner View



Proposed Architectures





# Conclusion

- FPGA DSP Blocks can support HP FP adder architectures.
- Bounding exponent values can help reduce resources.
- Bit-pattern representations exploits can also improve resources.
- System-level improvements may exceed local savings.
- Reduced routing improves system performance.
- Alternative architecture always useful.