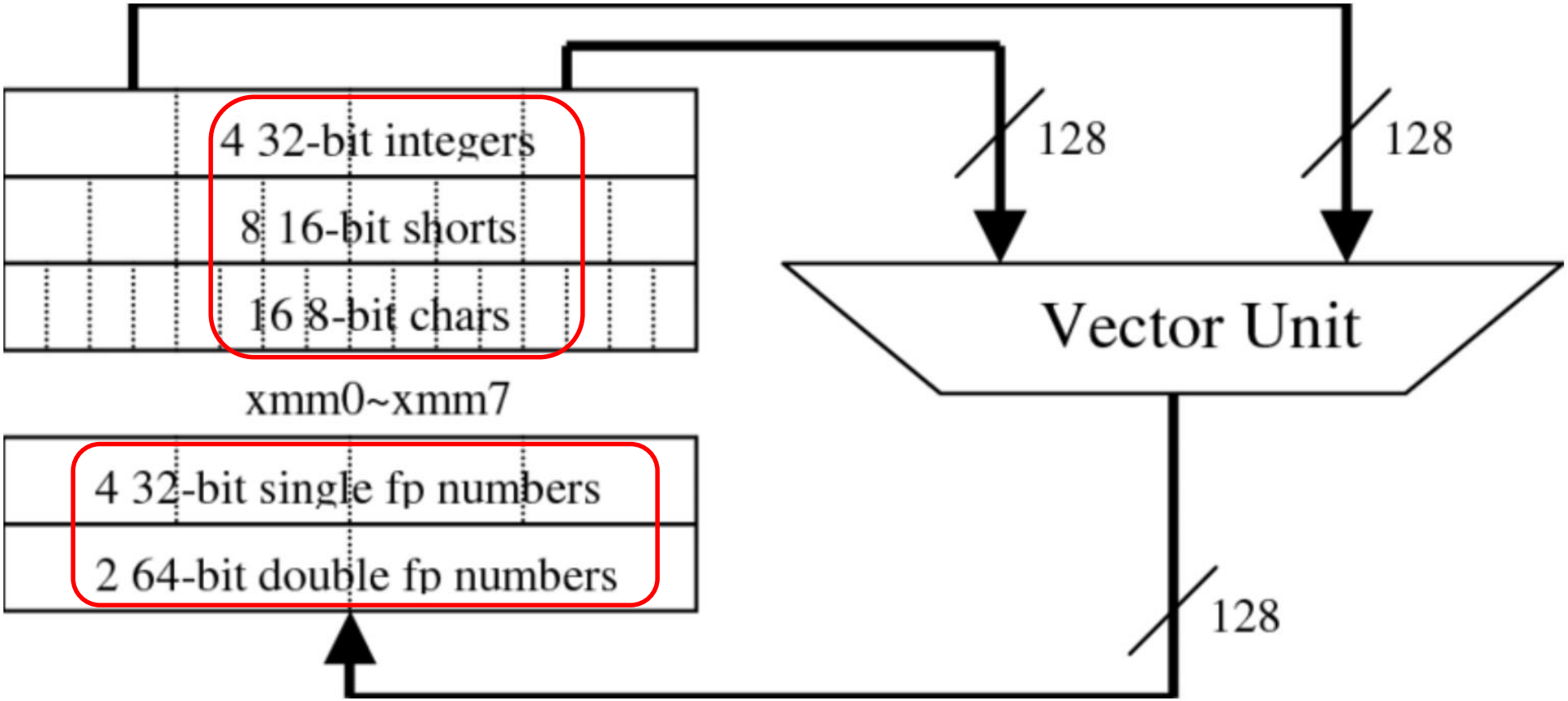
4.3 SIMD Instruction Set Extensions for Multimedia

SIMD Extensions

- **Many media applications operate on narrower data types than the 32-bit word size.**
  - Graphics: 8-bit color
  - Audio samples: 8-16 bits

- **By partitioning carry chains within a 256-bit adder, a processor could perform simultaneous operations on short vectors**
  - 32 * 8-bit operands
  - 16 * 16-bit operands
  - 8 * 32-bit operands

Example: 128-bit adder



SIMD Extensions vs. Vector Arch.

- **Limitations, compared to vector instructions:**
  - **Fix # of data operands** encoded into op code
    - led to the addition of hundreds of instructions
  - **No sophisticated addressing modes**
    - Not support strided, scatter-gather
  - **No mask registers**.
    - Not support conditional branching

Typical SIMD multimedia support

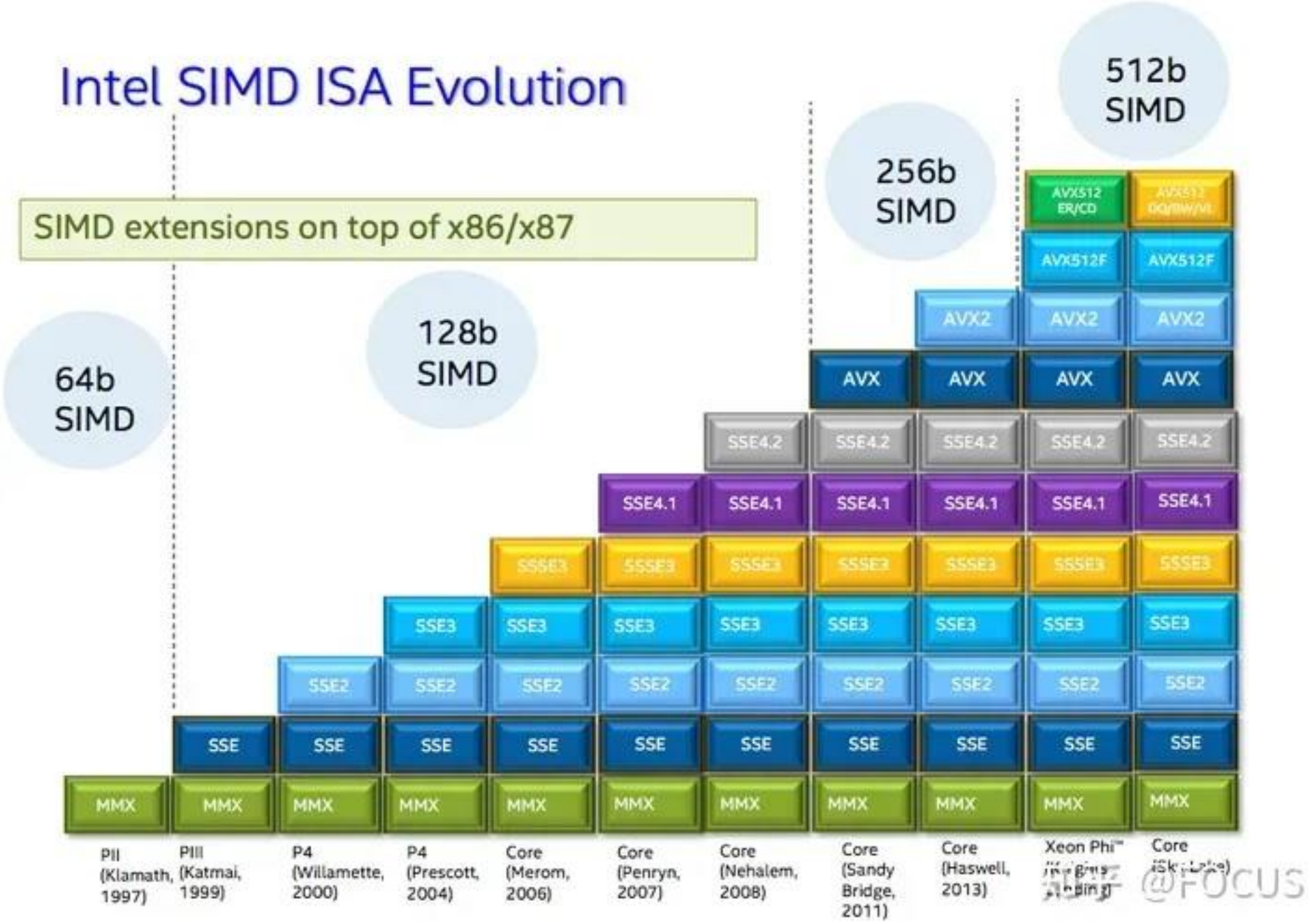
- Summary of typical SIMD multimedia support for 256-bit-wide operations

Instruction category	Operands
Unsigned add/subtract	Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit
Maximum/minimum	Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit
Average	Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit
Shift right/left	Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit
Floating point	Sixteen 16-bit, eight 32-bit, four 64-bit, or two 128-bit

SIMD Implementations

- **Intel MMX (1996)**
  - Eight 8-bit integer ops or four 16-bit integer ops
- **Streaming SIMD Extensions (SSE) (1999)**
  - Eight 16-bit integer ops
  - Four 32-bit integer/fp ops or two 64-bit integer/fp ops
- **Advanced Vector Extensions (AVX) (2010)**
  - Four 64-bit integer/fp ops
- **AVX-512 (2017)**
  - Eight 64-bit integer/fp ops

Intel SIMD ISA Evolution



RISC-V SIMD code for DAXPY

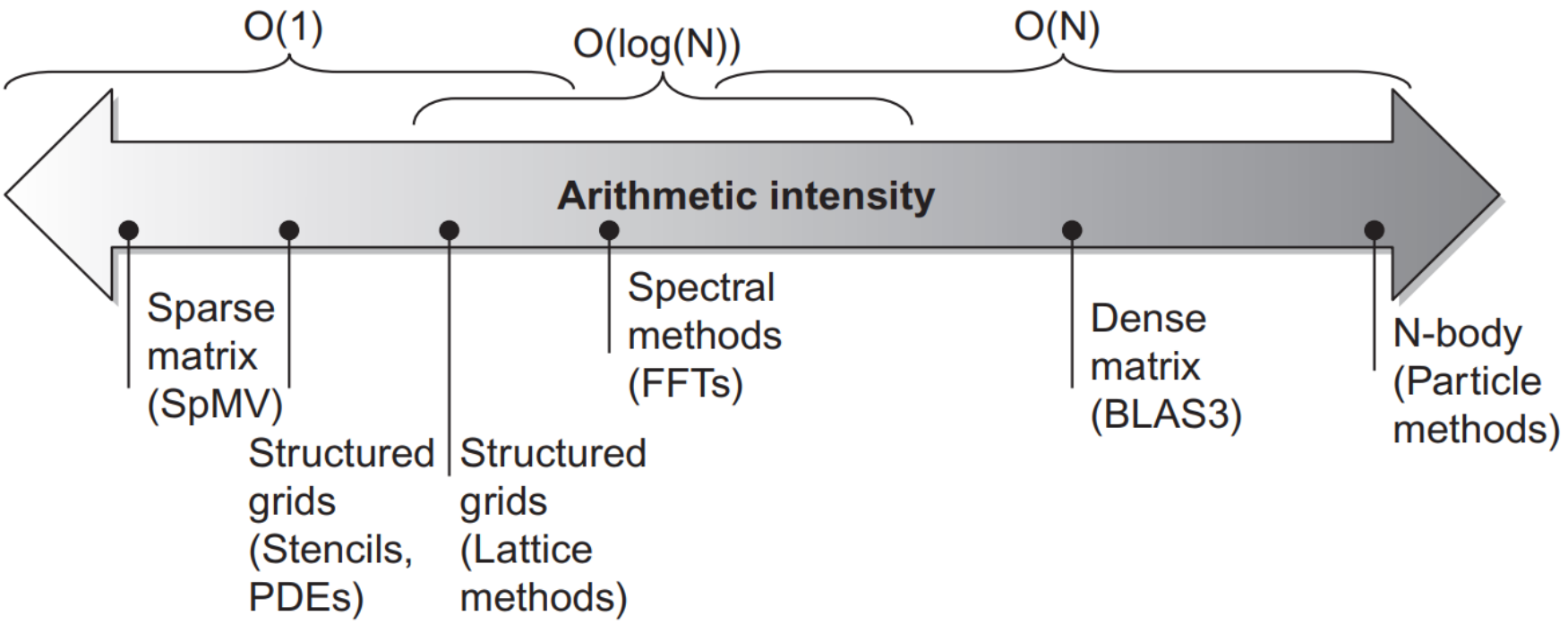
```
        fld        f0,a          #Load scalar a
        splat.4D   f0,f0         #Make 4 copies of a
        addi       x28,x5,#256   #Last address to load
Loop:   fld.4D     f1,0(x5)      #Load X[i] ... X[i+3]
        fmul.4D    f1,f1,f0      #a×X[i] ... a×X[i+3]
        fld.4D     f2,0(x6)      #Load Y[i] ... Y[i+3]
        fadd.4D    f2,f2,f1      # a×X[i]+Y[i]...
                                 # a×X[i+3]+Y[i+3]
        fsd.4D     f2,0(x6)      #Store Y[i]... Y[i+3]
        addi       x5,x5,#32     #Increment index to X
        addi       x6,x6,#32     #Increment index to Y
        bne        x28,x5,Loop   #Check if done
```

NOTE: We add the suffix "4D" on instructions that operate on four double-precision operands at once

Roofline Performance Model

Arithmetic intensity

- **Arithmetic intensity:** the ratio of floating-point operations per byte of memory accessed.



Roofline Performance Model

- One intuitive way to **compare variations of SIMD architectures** is the **Roofline model**

- **Basic idea:**
  - Plot peak floating-point throughput as a function of arithmetic intensity
  - Ties together floating-point performance and memory performance for a target machine

Examples

$$\text{Attainable GFLOPs/s} = \text{Min}(\text{Peak Memory BW} \times \text{Arithmetic Intensity}, \text{Peak Floating} - \text{Point Perf.})$$