EE290C - Fall 2018

Advanced Topics in Circuit Design

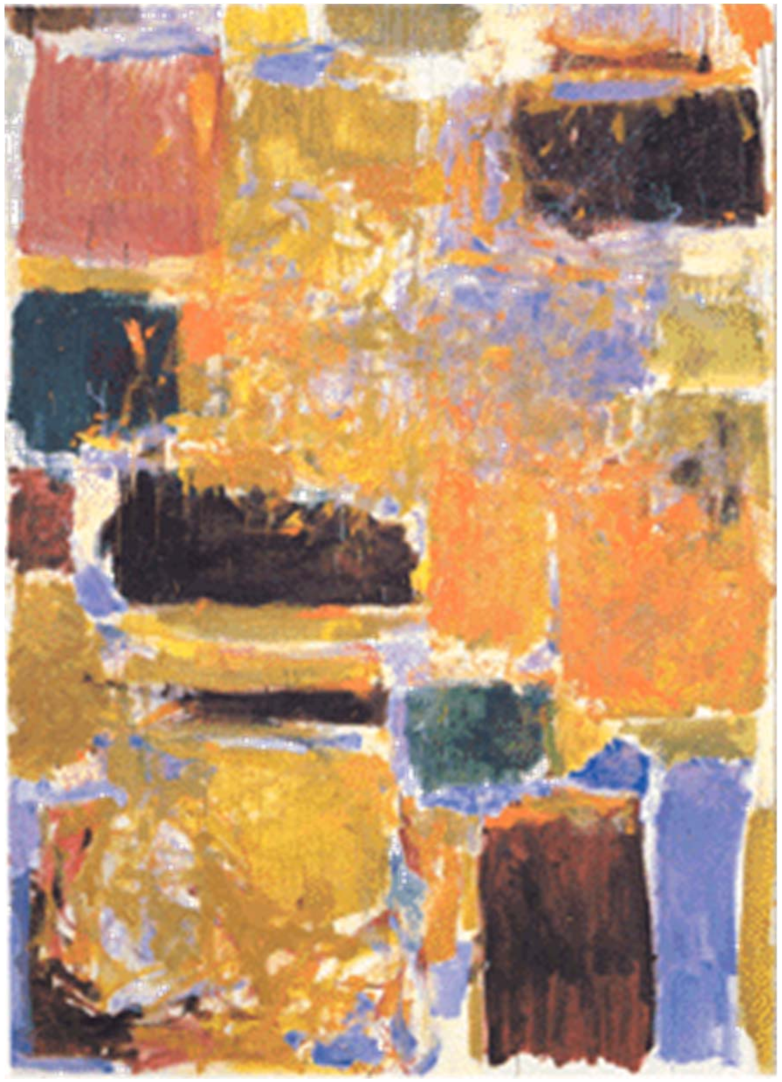
VLSI Signal Processing

Lecture 2: Computation Models

DSP Arithmetic

Announcements

- Assignment 1 – due on Thursday, 8/30
- Assignment 2 – due on Thursday 9/6
  - Will be posted on Thursday 8/30
  - If you have finished Bootcamp 2.2, just continue…



Advanced Topics in Circuit Design

VLSI Signal Processing

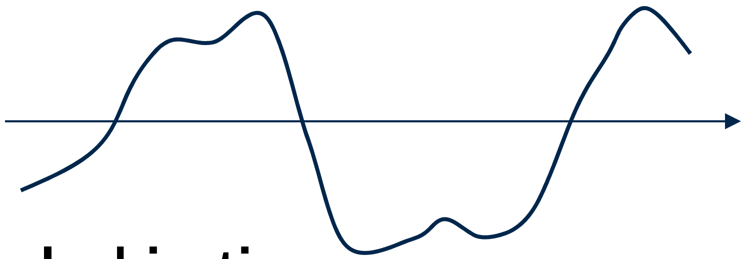
Computational Models

Reading

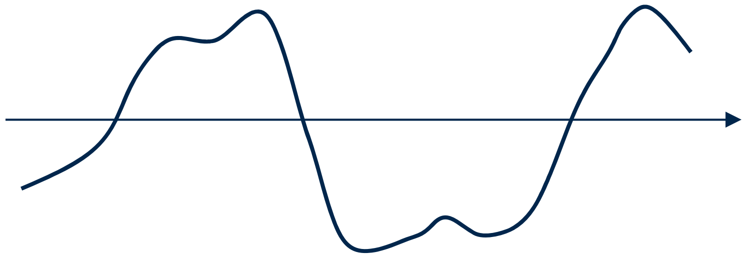
- Models
  - Parhi, VLSI Digital Signal Processing Systems, Wiley'99
  - Woods, McAllister, Lightbody, Yi, FPGA Implementation of DSP Systems, 2017 (Ch. 8)
  - Khan, Digital Design of Signal Processing Systems, 1999.
- Numbers
  - Markovic, Brodersen, DSP Architecture Design Essentials, 2012, (Ch. 5)
- CORDIC
  - Markovic, Brodersen, DSP Architecture Design Essentials, 2012, (Ch. 6)
  - Any other book above

Sampling and Quantization

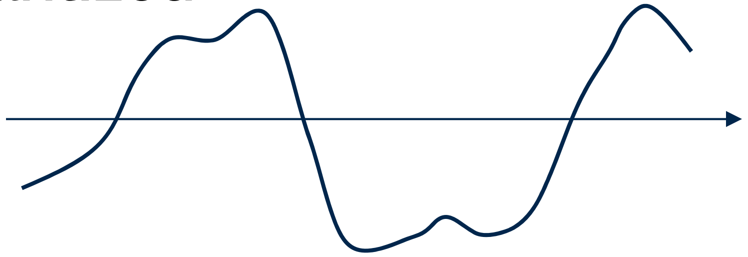
- Analog waveform



- Sampled in time



- Quantized

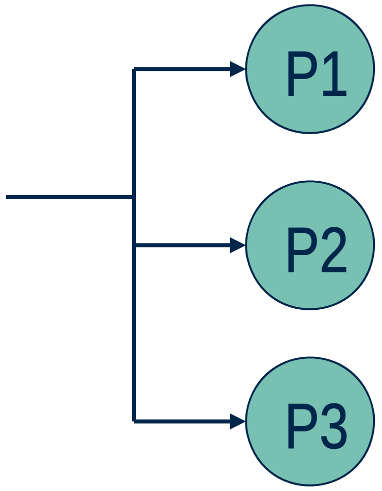


Throughput and Latency

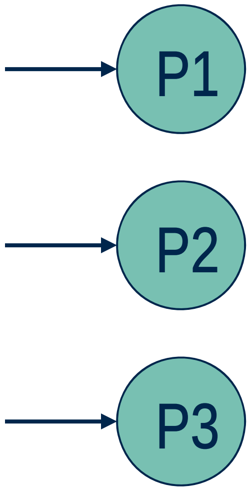
- Sampling rate
- Throughput (rate)
- Clock rate
- Latency

Parallelism

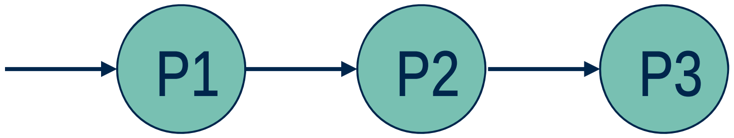
- Three processes, P1, P2 and P3



Single source



Multiple sources

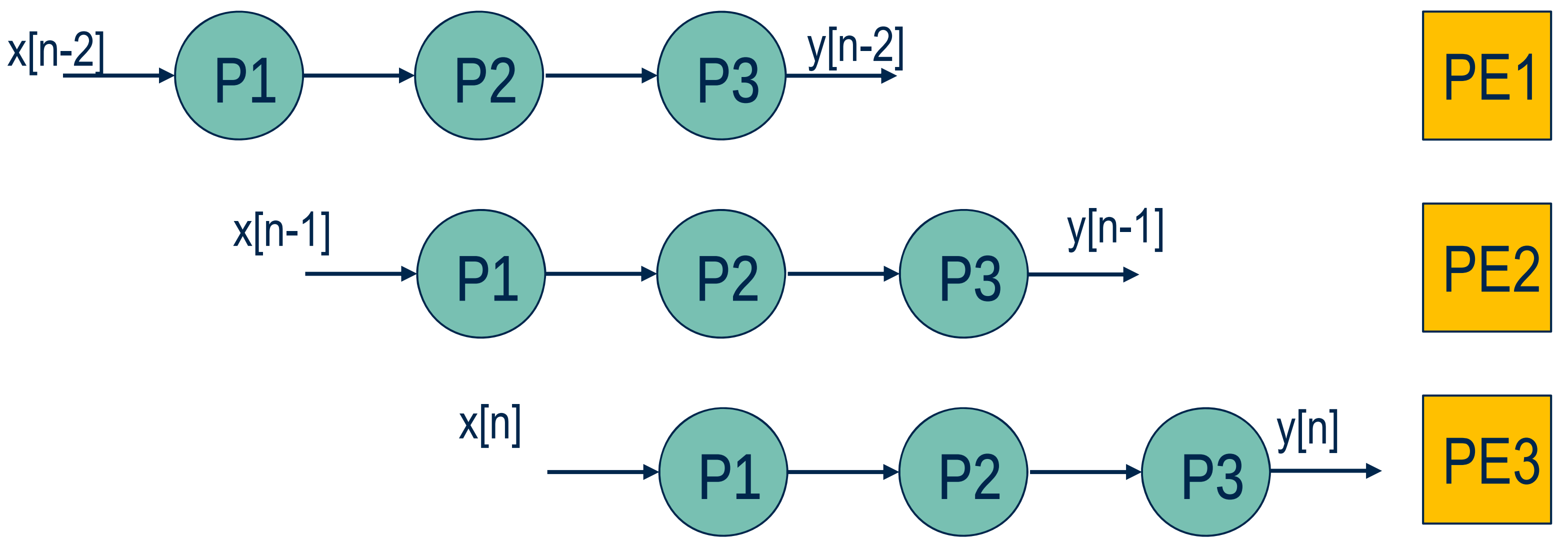


Sequential processing of a single source

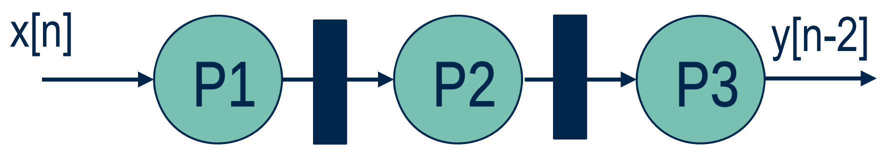
Throughput =

Interleaving

- Processes – P1, P2, P3
- Processing elements



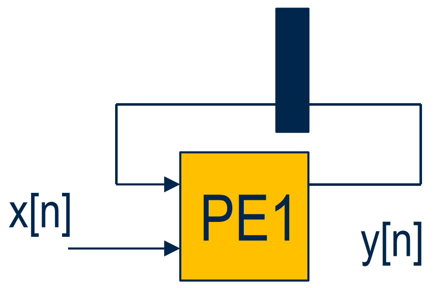
Pipelining



- Throughput^{-1} = max {P1, P2, P3}
- Latency increased by clock 2 cycles

Recursions and Pipelining

- Single-cycle recursion



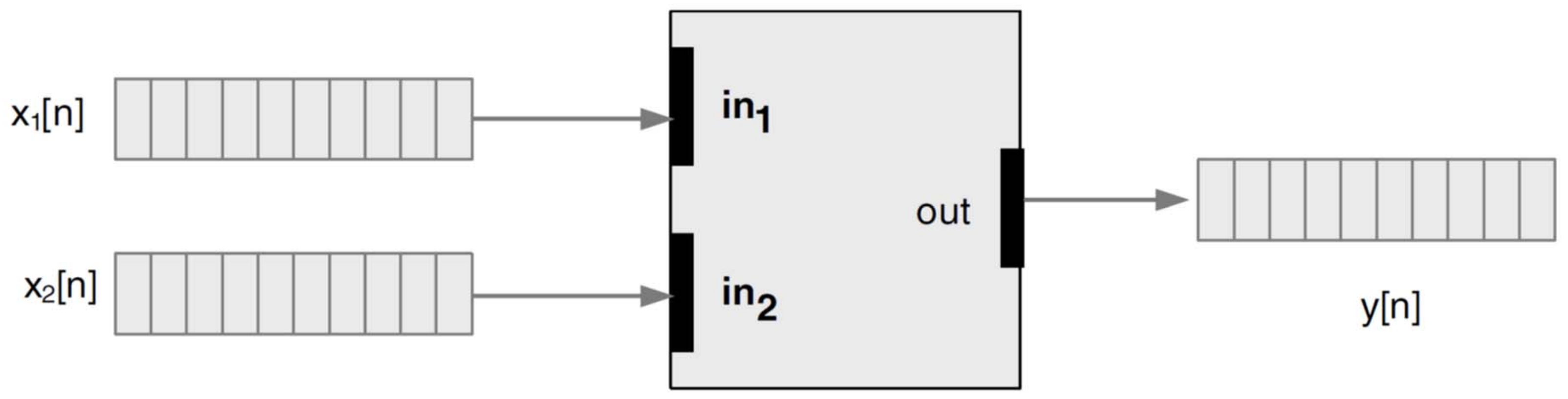
- Pipelining PE1:

DSP Algorithm Representations

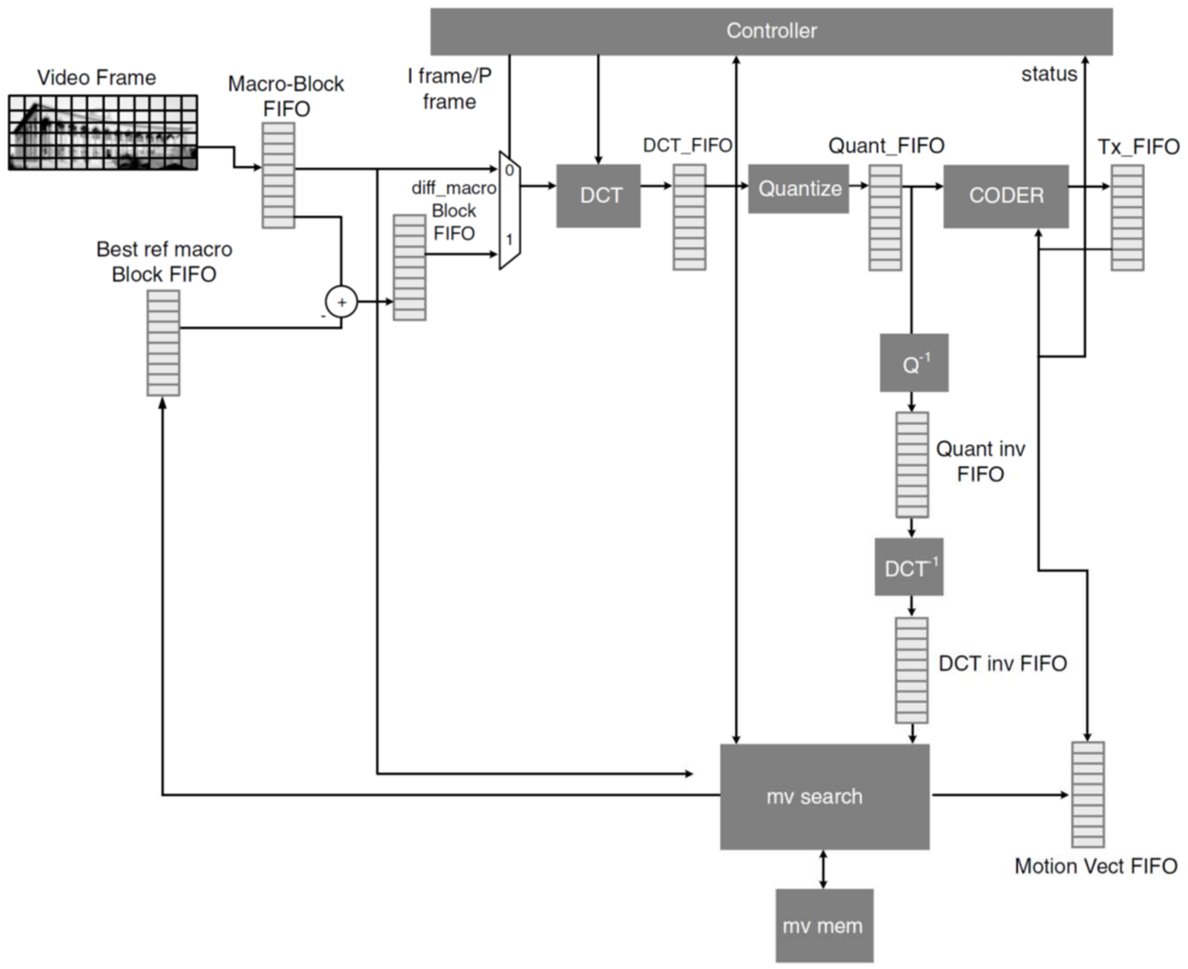
- Common representations:
  - Kahn Process Network (KPN)
  - Signal Flow Graph (SFG)
  - Dataflow Graph (DFG)
  - Block diagram
- Formal rules yield formal transformations

Kahn Process Network

- A node fires when tokens are available at input FIFOs
  - Straightforward method of mapping DSP into hardware
  - Each node executes a sequential program, and can either wait or execute on inputs
  - G. Kahn, 1974.



KPN for MPEG Compression



Khan, Digital Design of Signal Processing Systems, 1999.

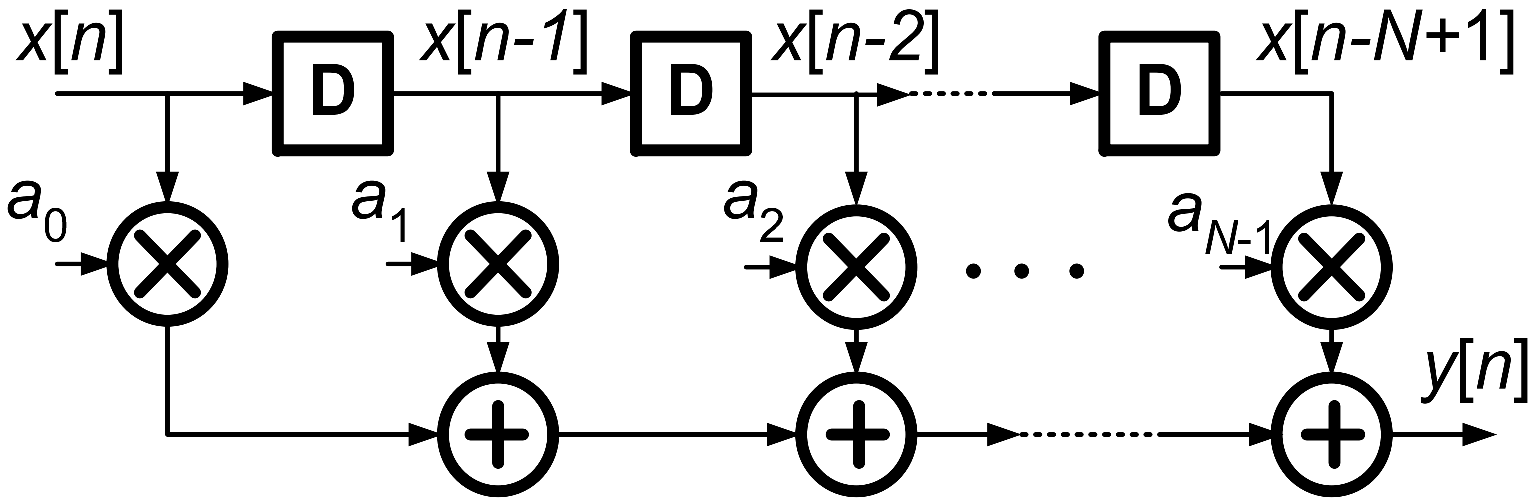
Limitations of KPN

- FIFO – hard to reorder data
- Sparse data (token always consumed)
- Iterations on the same data

Block Diagrams

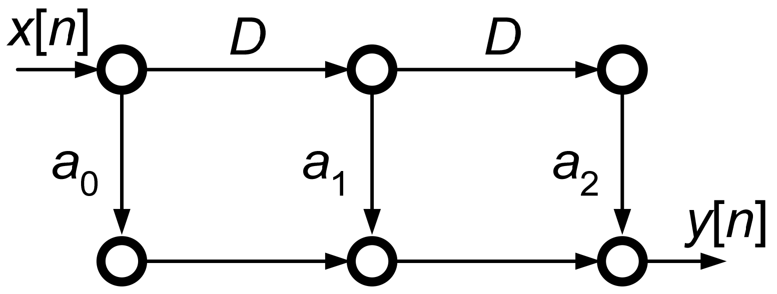
- Equation
- FIR filter, direct form

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \ldots + a_N x[n-N+1]$$



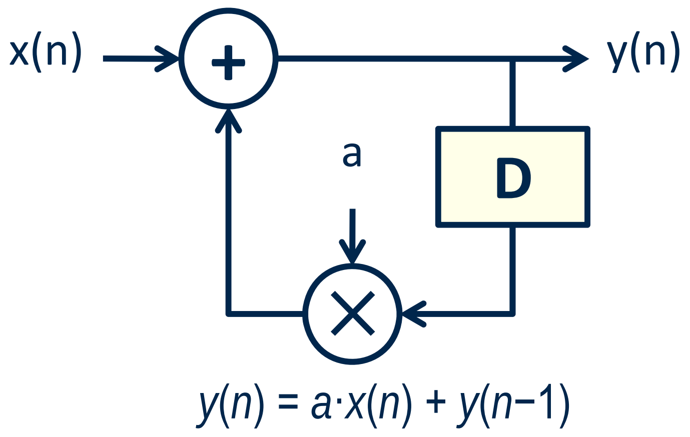
Signal Flow Graph

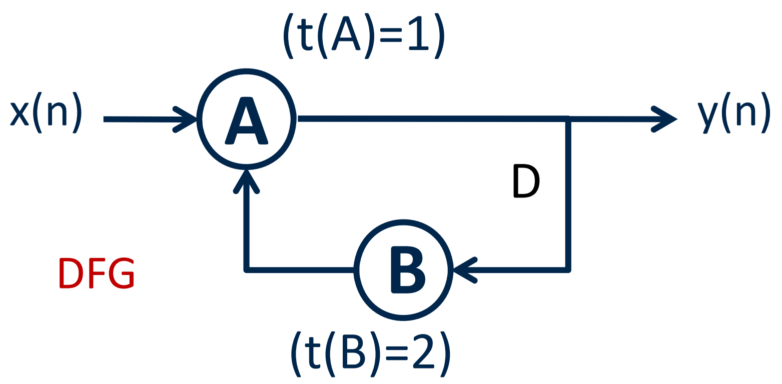
- Collection of nodes and directed edges [Crochiere, Oppenheim, 1975]
- Edges – constant multipliers and delays
- Source and sink nodes
- Effective graph transformations
- 3-tap FIR



Dataflow Graph

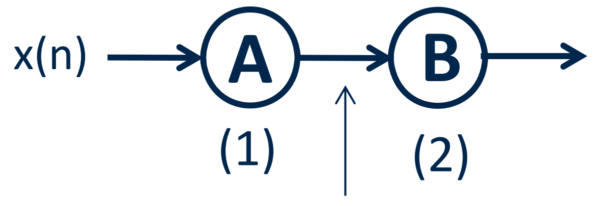
- Dataflow Graph (DFG) Model



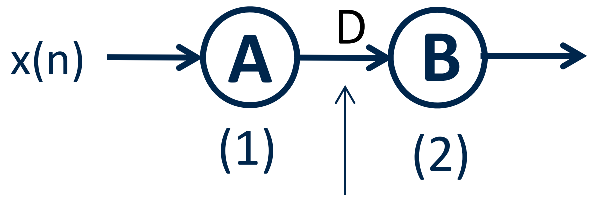


  - Computational functions represented as nodes A, B
  - Delays shown on edges (kD for k delays, representing z^{-k})
  - Computation time in brackets next to the nodes: t(A), t(B), …
  - A(i), B(i), …. represent **i**th iteration of functions

Precedence Constraints

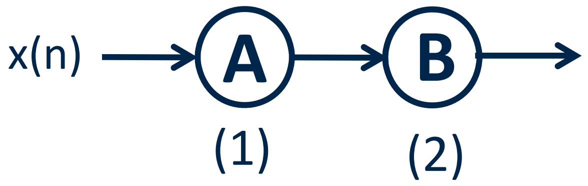


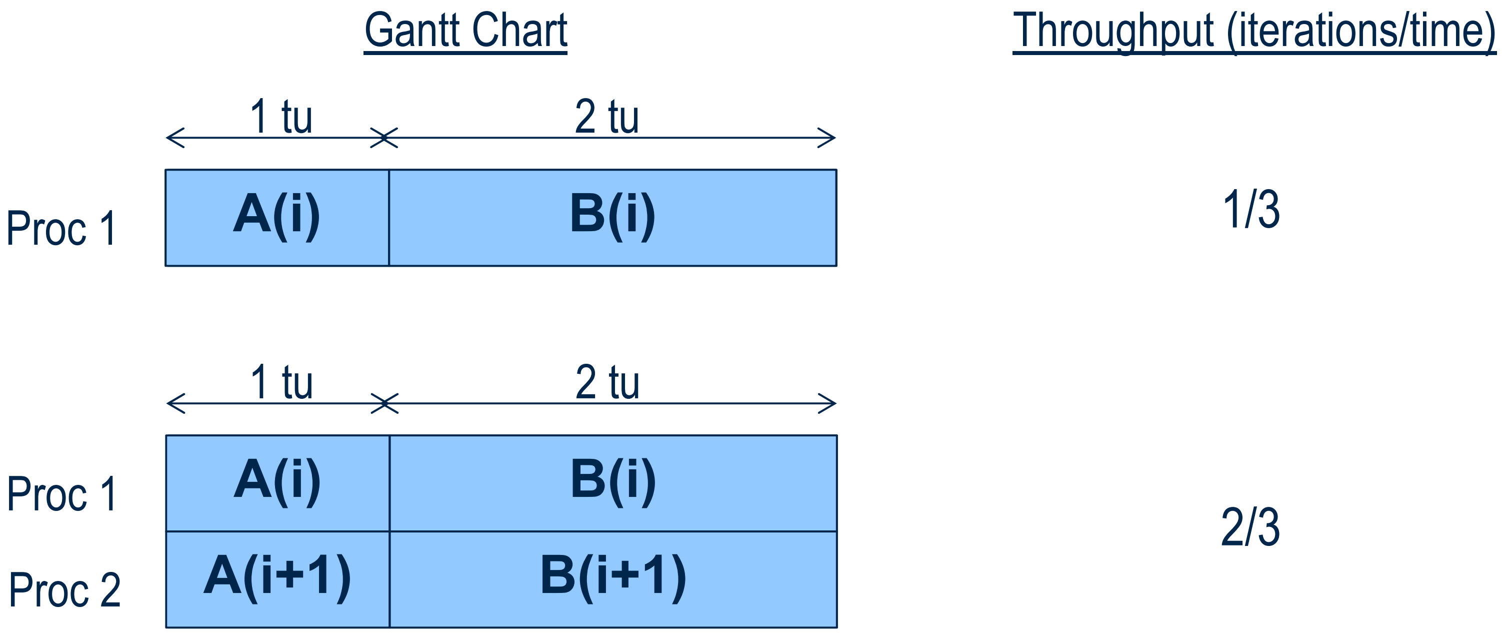
Edge represents intra-iteration precedence constraint –
B(i) must occur after A(i)



Delay represents inter-iteration precedence constraint –
B(i) must occur after A(i-1)

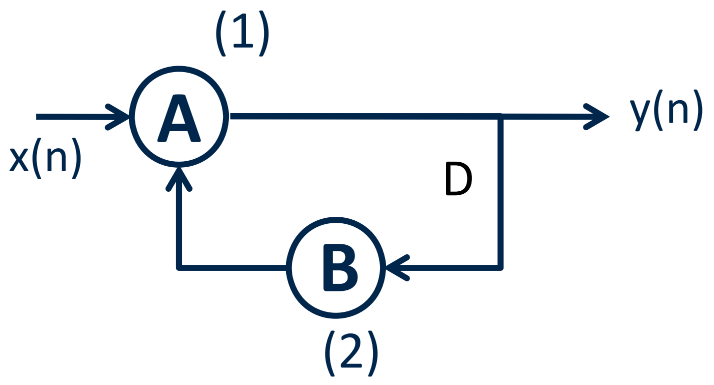
Schedules and Throughput

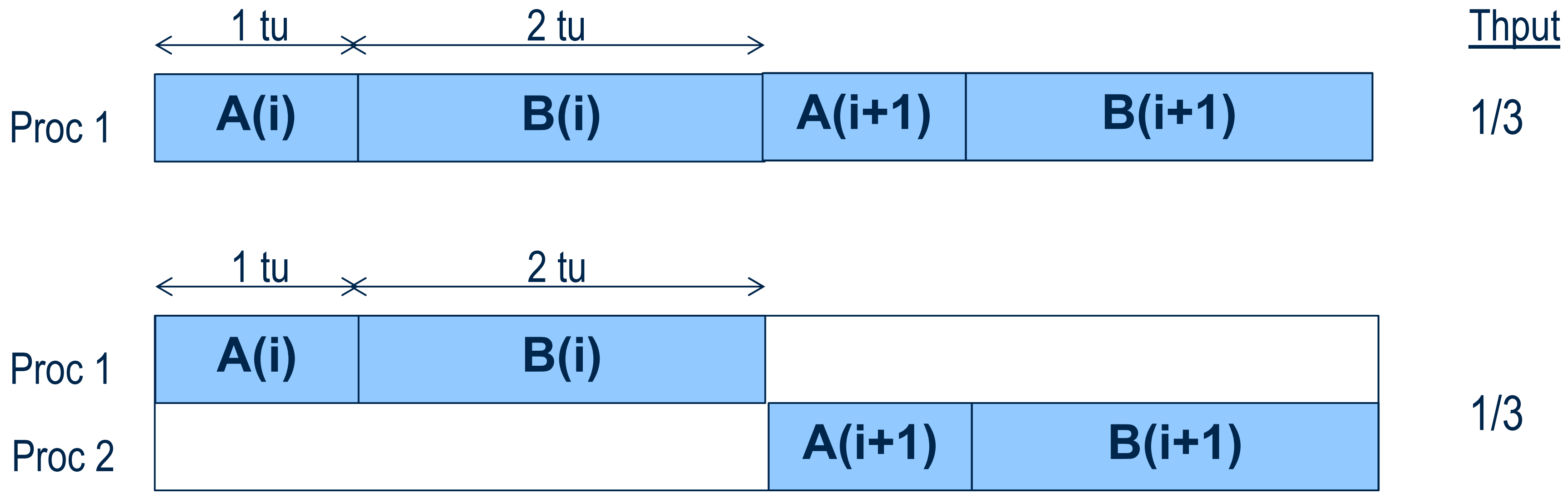




- 3 processors? 4 processors? ∞ processors?

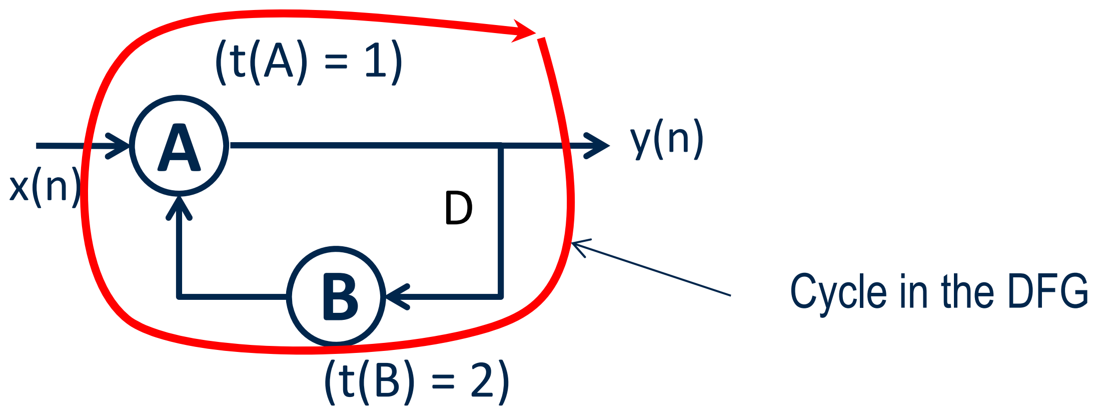
Recursive Computation





- 3 processors? 4 processors? ∞ processors?

Throughput Limit: Iteration Bound

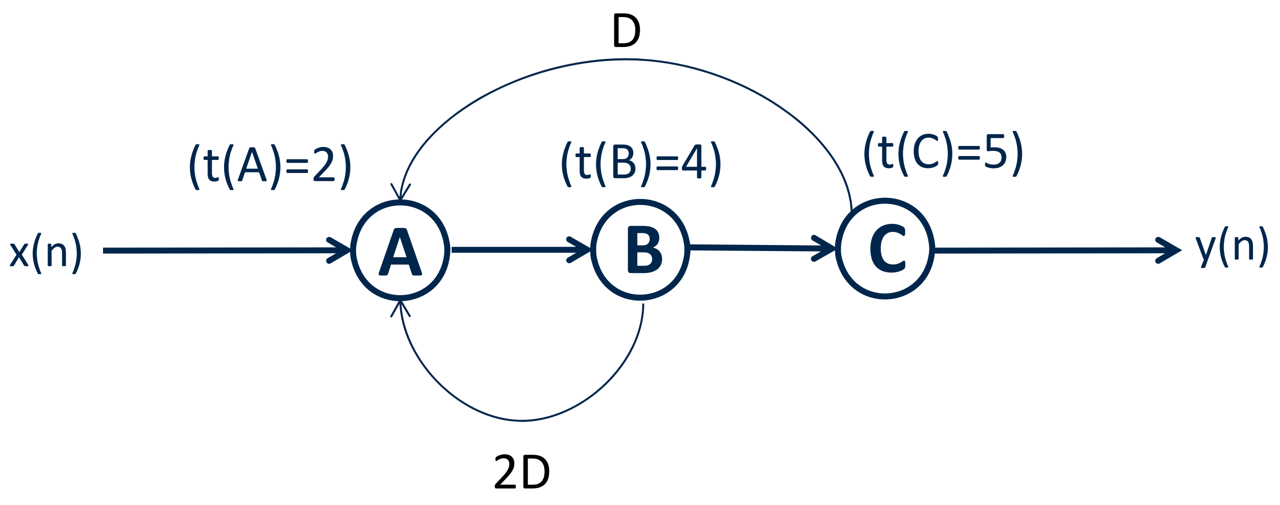


- Theorem: DFG is dead-lock free if and only if every cycle has non-zero sum-delay on it's edges
- If *C* is the set of cycles, *t(v)* is the computation delay of node *v*, and *d(e)* is the delay on edge *e*, then the Iteration Bound T_∞ is given by the Maximum Cycle Mean:

$$\boxed{T_\infty = \max_{cycles\ C}\left[\frac{\sum_{v\in C} t(v)}{\sum_{e\in C} d(e)}\right]}$$

$\boxed{\text{Fundamental Limit: Max Throughput} \leq T_\infty}$

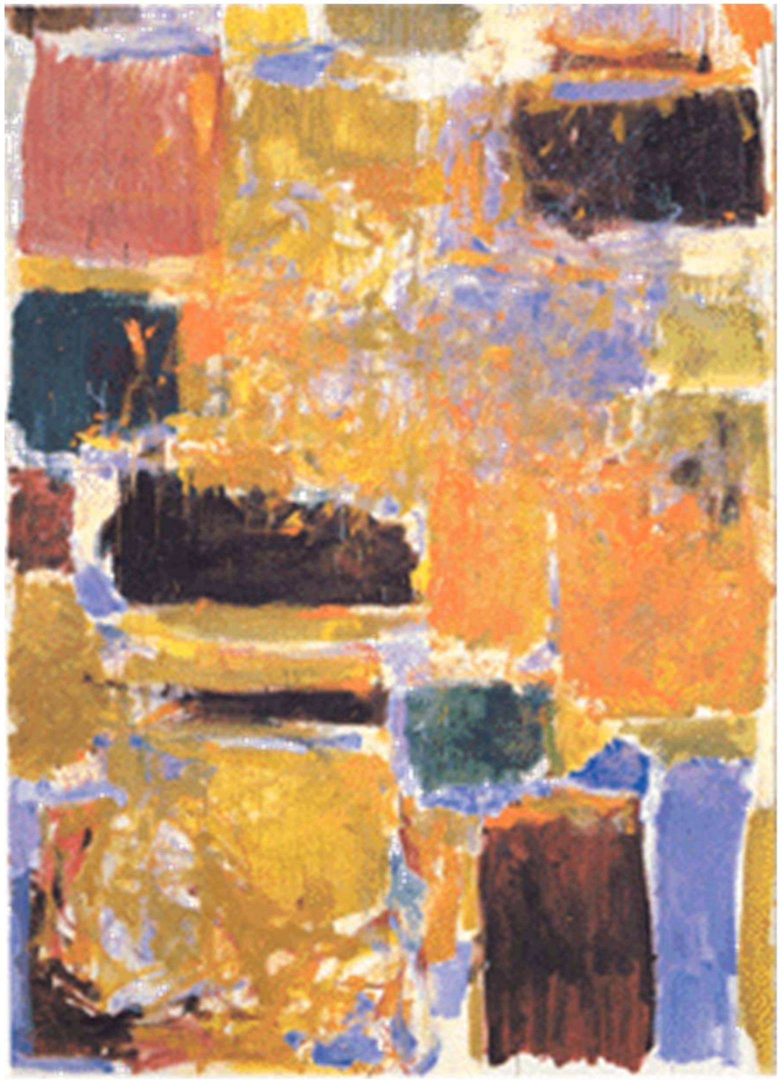
Iteration Bound



- Iteration Bound = max cycle mean = *max* {6/2, 11/1} = 11 t.u.
- Very useful for understanding limits to achievable thput
  - Ito-Parhi Algorithm for computing MCM: O(# nodes x # edges)

DFG Transformations

- There are many graph transformations that can be used to trade off power, performance, area
  - Pipelining
  - Paralleism
  - Interleaving
  - Folding
  - Loop unrolling
- Instead of going through theoretical concepts now, we will get to them through practical examples



Advanced Topics in Circuit Design

VLSI Signal Processing

DSP Arithmetic

Number Systems: Algebraic

Algebraic Expressions

e.g. $a = \pi + b$

- High-level abstraction
- Infinite precision
- Often easier to understand
- Good for theory/algorithm development
- Hard to implement
- Area/power vs. bitwidth and representation

[1] C. Shi, Floating-point to Fixed-point Conversion, Ph.D. Thesis, University of California, Berkeley, 2004.

Number Systems: Floating Point

- Widely used in CPUs
- Multiple precisions
- Often ‘golden model’ is in FP

Chisel type: DSPReal

$$\text{Value} = (-1)^{\text{Sign}} \times \text{Fraction} \times 2^{(\text{Exponent} - \text{Bias})}$$ [2]

IEEE 754 standard	Sign	Exponent	Fraction	Bias
Single precision [31:0]	1 [31]	8 [30:23]	23 [22:0]	127
Double precision [63:0]	1 [63]	11 [62:52]	52 [51:00]	1023

[2] J.L. Hennesy and D.A. Paterson, Computer Architecture: A Quantitative Approach.

Floating-Point Standard: IEEE 754

- Property #1
  - Rounding a “half-way” result to the nearest float (picks even)
  - Example:

    6.1 × 0.5 = 3.05 (base 10, 2 digits)

    even $\boxed{3.0}$ 3.1 (base 10, 1 digit)
- Property #2
  - Includes special values (NaN, ∞, −∞)
  - Examples:

    sqrt(-0.5) = NaN, *f*(NaN) = NaN [*check this!*]

    1/0 = ∞, 1/∞ = 0

[3] Markovic Brodersen, DSP Architecture Design Essentials.

Floating-Point Standard: IEEE 754

- Property #3
  - Uses denormals to represent the result < $1.0 \times e^{Emin}$

$$\begin{tikzcd} & \text{Flush to 0} \\ Emin = \text{min exponent} \arrow[ur] \arrow[dr] & \\ & \text{Use significand} < 1.0 \text{ and } Emin \text{ (``gradual underflow'')} \end{tikzcd}$$

Example:

base 10, 4 significant digits, $x = 1.234 \times 10^{Emin}$

denormals: $x/10 \rightarrow 0.123 \times 10^{Emin}$

$x/1{,}000 \rightarrow 0.001 \times 10^{Emin}$

$x/10{,}000 \rightarrow 0$

$x = y \Leftrightarrow x - y = 0$

flush-to-0: $x = 1.256 \times 10^{Emin}$, $y = 1.234 \times 10^{Emin}$

$x - y = \underbrace{0.022 \times 10^{Emin}}_{\text{denormal number (exact computation)}} = 0$ (although $x \neq y$)

Floating-Point Standard: IEEE 754

- Property #4
  - Rounding modes
    - Nearest (default)
    - Toward 0
    - Toward ∞
    - Toward −∞

Floating-Point Numbers

- Single precision: 32 bits
  - Sign: 1 bit
  - Exponent: 8 bits
  - Fraction: 23 bits
    - *Fraction* < 1 $\Rightarrow$ *Significand* = 1 + *Fraction*
  - Bias = 127

Example:

1   10000001 0100…0   (significand = 1.25)

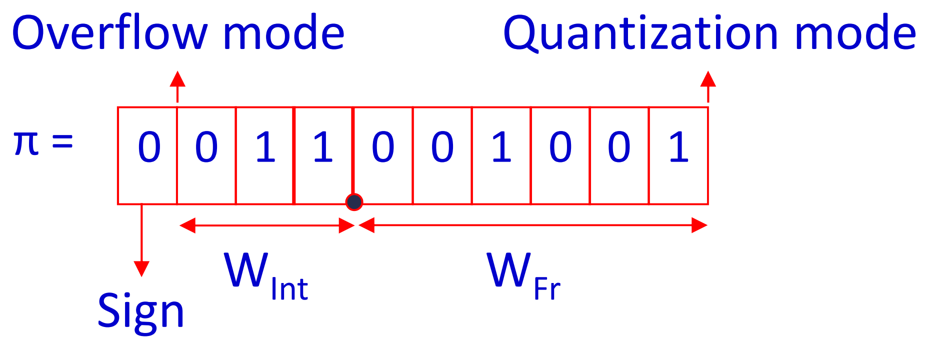
sign exponent   fraction

129 – 127 $0.01_2 = 0.25$

$-1.25 \times 2^2 = -5$

Fixed Point: 2's Complement

Chisel type: Fix



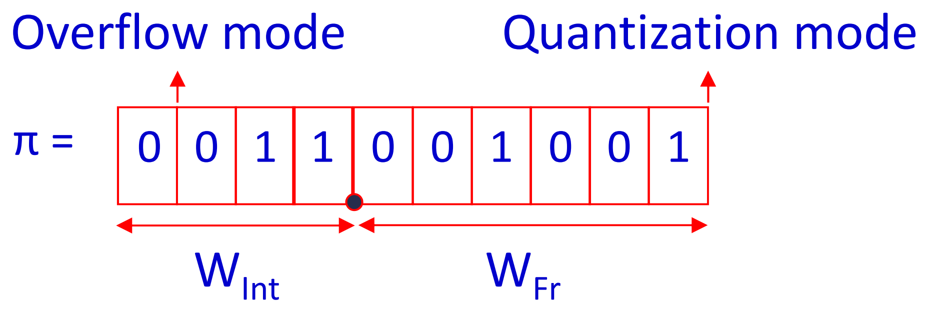
$$= 0\times2^{3} + 0\times2^{2} + 1\times2^{1} + 1\times2^{0} + \overbrace{0\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 0\times2^{-5} + 1\times2^{-6}}^{\text{fractional}}$$

$$= 3.140625$$

- W_{Int} and W_{Fr} suitable for predictable dynamic range
  - o-mode (overflow, wrap-around)
  - q-mode (trunc, roundoff)

- Compact implementation

Fixed Point: Unsigned Magnitude



- Useful built-in MATLAB functions:
  - fix, round, ceil, floor, dec2bin, bin2dec, etc.
- Find them in Scala!
  - round, ceil floor, Scala Breeze

Fixed-Point Representations

Chisel types: Uint, SInt

- Sign magnitude
- 2's complement
  - $x + (-x) = 2^n$ (complement each bit, add 1)
  - Most widely used (signed arithmetic easy to do)
- 1's complement
  - $x + (-x) = 2^n - 1$ (complement each bit)
- Biased $\rightarrow$ add bias, encode as ordinary unsigned number
  - $k + \text{bias} \geq 0$, bias $= 2^{n-1}$ (typically)

Fixed-Point Representations: Example

Example: n = 4 bits, k = 3, $-k$ = ?

- **Sign-magnitude: $k = 0011_2 \rightarrow -k = 1011_2$**
- 2's complement: $k + 1011 = 2^n$

$$\begin{array}{rr} -k = & \mathbf{1100} \\ + & \mathbf{1} \\ \hline & \mathbf{1101_2} \end{array} \qquad \begin{array}{r} 0011 \\ \mathbf{+1101} \\ \hline \mathbf{10000} \end{array}$$

Procedure:
- Bit-wise inversion
- Add "1"

- **1's complement: $-k = 1100_2$ $k + (-k) = 2^n - 1$**
- **Biased: $k + \text{bias} = 1011_2$ $-k + \text{bias} = 0101_2 = 5 \geq 0$**
  $2^{n-1} = 8 = 1000_2$

Overflow

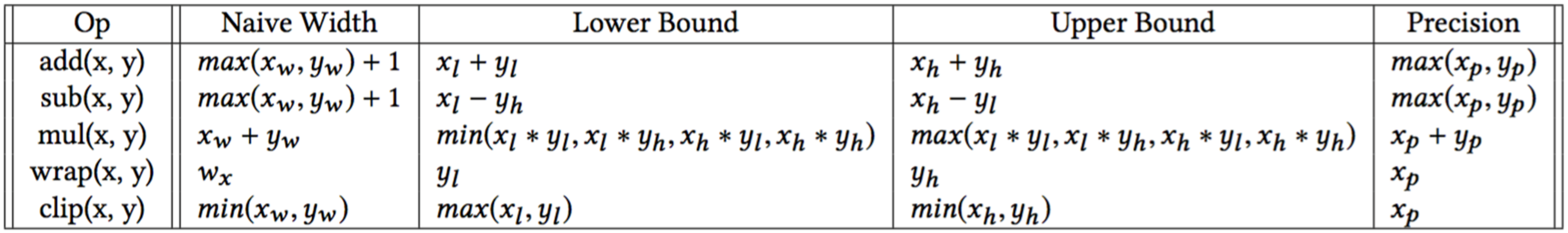
- Example: unsigned 4-bit addition

$$\begin{aligned} 6 &= 0110_2 \\ +11 &= 1011_2 \\ = 17 &= \underbrace{1}_{\text{extra bit}}0001_2 \text{ (5 bits!)} \end{aligned}$$

- **Property of 2's complement**
  - Negation = bit-by-bit complement + 1 → C_{in} = 1, result: a − b

Interval Arithmetic

- Don't think of numbers as *bits* with *bitwidths*!
- Think of them as ***numbers*** (Interval types), with ***ranges***!

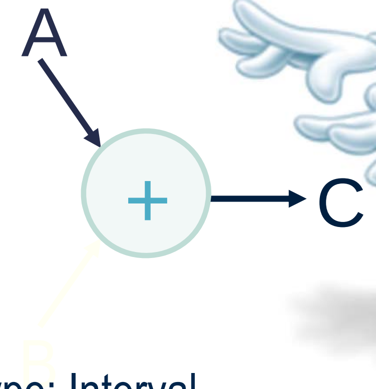


Op	Naive Width	Lower Bound	Upper Bound	Precision
add(x, y)	$max(x_w, y_w) + 1$	$x_l + y_l$	$x_h + y_h$	$max(x_p, y_p)$
sub(x, y)	$max(x_w, y_w) + 1$	$x_l - y_h$	$x_h - y_l$	$max(x_p, y_p)$
mul(x, y)	$x_w + y_w$	$min(x_l * y_l, x_l * y_h, x_h * y_l, x_h * y_h)$	$max(x_l * y_l, x_l * y_h, x_h * y_l, x_h * y_h)$	$x_p + y_p$
wrap(x, y)	w_x	y_l	y_h	x_p
clip(x, y)	$min(x_w, y_w)$	$max(x_l, y_l)$	$min(x_h, y_h)$	x_p

- + Propagate smart inference!
  - Reduce # of intermediate bits ☺
  - Guarantee correct result ☺

$$\underset{\text{4-bit}}{15} + \underset{\text{4-bit}}{15} = \underset{\text{5-bit}}{30}$$

$$\underset{\substack{\text{2-bit}\\ [0,2]}}{2} \times \underset{\substack{\text{2-bit}\\ [0,2]}}{2} = \underset{\substack{\text{3-bit}\\ [0,4]}}{4}$$



Chisel type: Interval

Don't sweat the details!

Angie Wang, Ph.D. Dissertation, UC Berkeley, 2018.

Other Number Representations

- Affine arithmetic
  - Intervals with correlations
- Logarithmic number systems (LNS)
- Redundant number systems
  - Carry-free arithmetic
  - Will revisit later
- Residue number system

Next Lecture

- Quantization, finite precision
- CORDIC algorithm