NUMBER REPRESENTATIONS

Lecture-12

Binary numbers

Binary

`0` and `1`

Integers

Place values

$$\underline{0101_b} = 0 \times \underline{2^3} + 1 \times \underline{2^2} + 0 \times \underline{2^1} + 1 \times \underline{2^0} = 5_d$$

Negative Numbers

- No `-` sign: compensate using convention
  - Signed magnitude: 1st bit indicates sign
  - Two's complement

Negative Numbers

- No `-` sign: compensate using convention
  - Signed magnitude: 1st bit indicates sign
  - Two's complement

2's complement

$$0101_b = 0 \times (-2^3) + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5_d$$

$$1101_b = 1 \times (-2^3) + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = -3_d$$

> Subtraction is now almost identical to addition

Dynamic range

- Ratio of largest magnitude to smallest (non-zero) magnitude

Dynamic range

> Ratio of largest magnitude to smallest (non-zero) magnitude

eg Decimal 4 digits (+ve):

- Largest: `9999`
- Smallest: `1`
- Dynamic range = 9999
    - Usually expressed in dB: $20 \underset{\downarrow \, \log_{10}}{\log} D \approx \underline{\underline{80dB}}$

Dynamic range

> Ratio of largest magnitude to smallest (non-zero) magnitude

eg Decimal 4 digits (+ve):

- Largest: `9999`
- Smallest: `1`
- Dynamic range = 9999
  - Usually expressed in dB: $20 \log D \approx 80dB$
- Binary:
  - Each bit doubles range: $20 \log 2 \approx 6dB$
  - 16 bits $\approx 96dB$

(no 2's c)

eg. 5-bit +ve :

$11111 \rightarrow 31$

$00001 \rightarrow 1$

$DR = \frac{31}{1} = 31$

$20 \log_{10} 31$

Real-valued numbers

> Convention: Choose location of *decimal* or *binary* point

Scaling

Choice of scaling factor determines resolution

Real-valued numbers

> Convention: Choose location of *decimal* or *binary* point

Scaling

Choice of scaling factor determines resolution

- Decimal:
  - xxxxx => smallest increment = 1
  - xxx.xx => smallest increment = 0.01
  - x.xxxx => smallest increment = 0.0001

Real-valued numbers

> Convention: Choose location of *decimal* or *binary* point

Scaling

Choice of scaling factor determines resolution

- Decimal:
  - xxxxx => smallest increment = 1
  - xxx.xx => smallest increment = 0.01
  - x.xxxx => smallest increment = 0.0001

> Dynamic range is the same for all:
> $99999/1 = 999.99/0.01 = 9.9999/0.00001 \approx 100dB$

Arithmetic: Addition

Scaling factor.

```
  0101.01     = 5.25
+   01.0001   = 1.0625
-----------
  0110.0101   = 6.3125
```

$25/4$

$17/2^4 = 17/16$

- Align the binary points

Arithmetic: Addition

```
      6
  0101.01     = 5.25
+   01.0001   = 1.0625
------------ 6
  0110.0101   = 6.3125
           8 bits
```

> Align the binary points

- Truncate output if needed?
- Overflow possible, same as with integers

Arithmetic: Multiplication

```
 0101.01        -> Multiplicand
   01.0001      -> Multiplier
  -----------
      .010101   x 2^-4
     0.00000    x 2^-3
    00.0000     x 2^-2
   000.000      x 2^-1
  0101.01       x 2^ 0
 00000.0        x 2^ 1
 ------------
 00101.100101
```

Partial product.

Product.

Scaling factors:

Significand /Mantissa.

$$5.25_d = 0101.01_b = (010101)_b \times 2^{-2}$$

- 6-bit *mantissa* = 21 and scale factor = 2^{-2}

Scaling factors:

$$5.25_d = 0101.01_b = (010101)_b \times 2^{-2}$$

> 6-bit *mantissa* = 21 and scale factor = 2^{-2}

$$1.0625_d = 01.0001_b = (010001)_b \times 2^{-4}$$

> 6-bit *mantissa* = 17 and scale factor = 2^{-4}

VIVADO HLS

ap_fixed<>

ap_fixed

- Arbitrary precision fixed point
- `ap_fixed<W,I,Q,O,N>`

<table>
<tr><th>Identifier</th><th colspan="3">Description</th></tr>
<tr><td>W</td><td colspan="3">Word length in bits</td></tr>
<tr><td>I</td><td colspan="3">The number of bits used to represent the integer value (the number of bits above the binary point)</td></tr>
<tr><td rowspan="10">Q</td><td colspan="3">Quantization mode
This dictates the behavior when greater precision is generated than can be defined by smallest fractional bit in the variable used to store the result.</td></tr>
<tr><td>SystemC Types</td><td>ap_fixed Types</td><td>Description</td></tr>
<tr><td>SC_RND</td><td>AP_RND</td><td>Round to plus infinity</td></tr>
<tr><td>SC_RND_ZERO</td><td>AP_RND_ZERO</td><td>Round to zero</td></tr>
<tr><td>SC_RND_MIN_INF</td><td>AP_RND_MIN_INF</td><td>Round to minus infinity</td></tr>
<tr><td>SC_RND_INF</td><td>AP_RND_INF</td><td>Round to infinity</td></tr>
<tr><td>SC_RND_CONV</td><td>AP_RND_CONV</td><td>Convergent rounding</td></tr>
<tr><td>SC_TRN</td><td>AP_TRN</td><td>Truncation to minus infinity (default)</td></tr>
<tr><td>SC_TRN_ZERO</td><td>AP_TRN_ZERO</td><td>Truncation to zero</td></tr>
</table>

<table>
<tr><th>Identifier</th><th colspan="3">Description</th></tr>
<tr><td rowspan="7">O</td><td colspan="3">Overflow mode.
This dictates the behavior when the result of an operation exceeds the maximum (or minimum in the case of negative numbers) value which can be stored in the result variable.</td></tr>
<tr><td>SystemC Types</td><td>ap_fixed Types</td><td>Description</td></tr>
<tr><td>SC_SAT</td><td>AP_SAT</td><td>Saturation</td></tr>
<tr><td>SC_SAT_ZERO</td><td>AP_SAT_ZERO</td><td>Saturation to zero</td></tr>
<tr><td>SC_SAT_SYM</td><td>AP_SAT_SYM</td><td>Symmetrical saturation</td></tr>
<tr><td>SC_WRAP</td><td>AP_WRAP</td><td>Wrap around (default)</td></tr>
<tr><td>SC_WRAP_SM</td><td>AP_WRAP_SM</td><td>Sign magnitude wrap around</td></tr>
<tr><td>N</td><td colspan="3">This defines the number of saturation bits in the overflow wrap modes.</td></tr>
</table>

Scaling factors:

$5.25_d = 0101.01_b = (010101)_b \times 2^{-2}$

ap-fixed<6, 4>

- 6-bit *mantissa* = 21 and scale factor = 2^{-2}

$1.0625_d = 01.0001_b = (010001)_b \times 2^{-4}$

ap-fixed<6, 2>

- 6-bit *mantissa* = 17 and scale factor = 2^{-4}

ap-fixed<12, 6>

$5.578125_d = 00101.100101_b = (00101100101)_b \times 2^{-6}$

- Product: 12-bit mantissa = 21 × 17 = 357 and scale factor = $2^{-2} \times 2^{-4}$

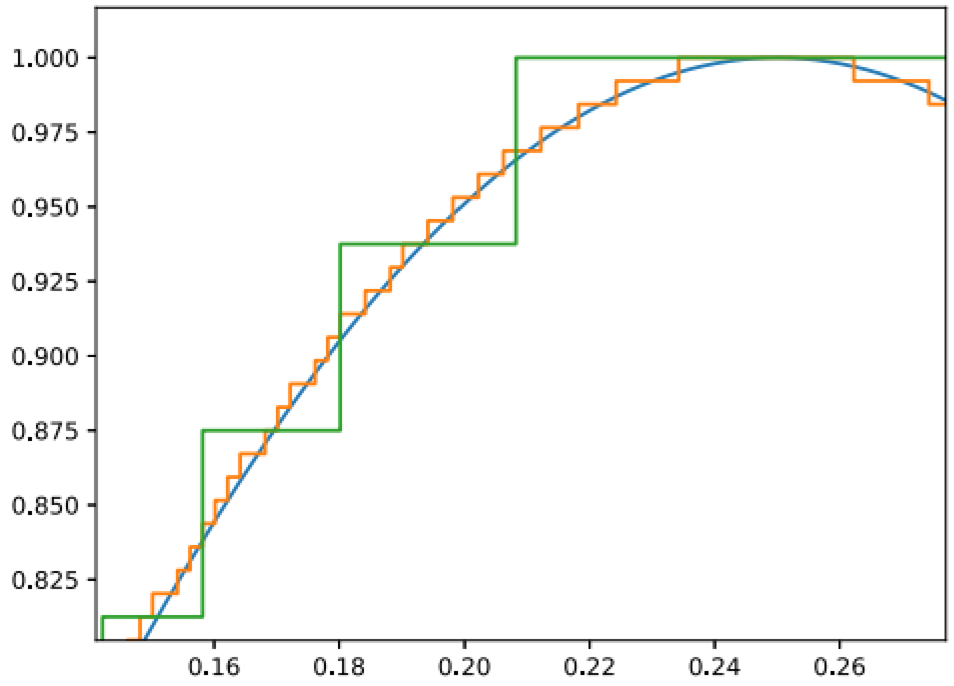
QUANTIZATION AND ERROR

Lecture 12

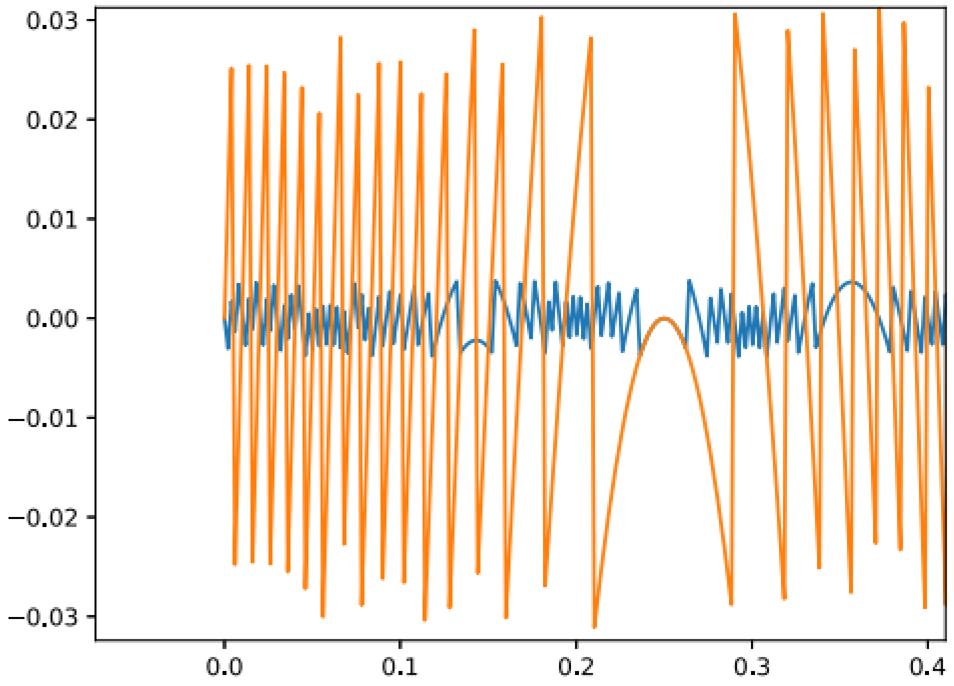
Quantization

- Magnitude
  - Largest value to be represented: integer
- Resolution
  - Smallest *delta* to be resolved: fraction

Quantization effects



Quantization Error



Error

- Assume uniformly distributed:
  - Model as additive noise
- * Signal to Noise Ratio (SNR)
- Roughly 6dB per bit of additional precision

FLOATING POINT AND OTHER NUMBER REPRESENTATIONS

Lecture 12

The scaling factor problem

- Example: SPICE circuit simulation

The scaling factor problem

Example: SPICE circuit simulation

- Typical ranges of values (for low-power electronic circuits) ~ DR: 1000s.
  - Voltages: up to about 5.0V max, resolution in mV (for references etc.)
  - Currents: nA to mA (leakage currents to bias currents) ~ 10^6
  - Impedance: $k\Omega$ to $M\Omega$ → 10^3

The scaling factor problem

> Example: SPICE circuit simulation

- Typical ranges of values (for low-power electronic circuits)
  - Voltages: up to about 5.0V max, resolution in mV (for references etc.)
  - Currents: nA to mA (leakage currents to bias currents) 10^6 10^{-9}
  - Impedance: $k\Omega$ to $M\Omega$ - 10^6

- Dynamic range
  - with individual units (μA), $k\Omega$ etc.) $\approx 10^6$ - 20 bits
  - SI units: nA to $M\Omega$ - 10^{15} - 45 bits

Efficient use of bits

Efficient use of bits

- 0.000123
- 12.3
- 123000000

Efficient use of bits

Significand

Scaling factors.

- 0.000123
- 12.3
- 123000000

- 1.23×10^{-4}
- 1.23×10^{1}
- 1.23×10^{8}

Efficient use of bits

- 0.000123
- 12.3
- 123000000

- 1.23×10^{-4}
- 1.23×10^{1}
- 1.23×10^{8}

Significant bits / digits

Exponent

Efficient use of bits

digit before point is not 0.

- 0.000123
- 12.3
- 123000000

- 1.23×10^{-4}
- 1.23×10^{1}
- 1.23×10^{8}

Normalized scientific notation

Significant bits / digits

Exponent

Options

- Each block has own exponent
- Each number has own exponent

Self-contained representation

- Mantissa: Significant bits - numerical value (1.23)
- Exponent: Scaling factor ← Fixed pt : implicit.
- **Sign**: Use a separate bit to represent this

Self-contained representation

- Mantissa: Significant bits - numerical value (1.23)
- Exponent: Scaling factor
- **Sign**: Use a separate bit to represent this

Example: IEEE 754 SP

Single Precision

S	Exponent	Mantissa			
0/1	e = 1 to 254 offset 127	1.mmm..mmm	Max: $2.0 - 2^{-23}$	Min: 1.0	MinRes: 2^{-23}

32b

1   8   e

Implicit   1 0101110 23 1

$\underbrace{1.0101110\ldots1}_{\text{24 bits.}} \times 2^{e}$

Dynamic range

- Largest positive value:
  - $1.111...1 \times 2^{+127} \approx 3.4 \times 10^{38}$
- Smallest positive value:
  - $1.000...0 \times 2^{-126} \approx 1.2 \times 10^{-38}$
- DR: $\approx 10^{77}$

Dynamic range

- Largest positive value:
  - $1.111...1 \times 2^{+127} \approx 3.4 \times 10^{38}$
- Smallest positive value:
  - $1.000...0 \times 2^{-126} \approx 1.2 \times 10^{-38}$
- DR: $\approx 10^{77}$

- Contrast 32-bit integer:
  - DR: $\frac{2^{31}}{1} \approx 2.2 \times 10^{9}$

Dynamic range

- Largest positive value:
  - $1.111...1 \times 2^{+127} \approx 3.4 \times 10^{38}$
- Smallest positive value:
  - $1.000...0 \times 2^{-126} \approx 1.2 \times 10^{-38}$
- DR: $\approx 10^{77}$

- Contrast 32-bit integer:
  - DR: $\frac{2^{31}}{1} \approx 2.2 \times 10^{9}$

Problem?

- Representing 9 digit integers?
  - $123456789 \rightarrow 123456792$

Special cases

- exponent = 255
  - mantissa = all zeros: $\pm\infty$ Infinity.
  - mantissa non-zero: NaN (not-a-number: *eg* $0/0$, $\infty - \infty$ etc.

Special cases

- exponent = 255
    - mantissa = all zeros: $\pm\infty$
    - mantissa non-zero: NaN (not-a-number: *eg* $0/0, \infty - \infty$ etc.

Addition

```
1.23 x 10^10
3.45 x 10^4
------------
   ? x 10^?
```

Addition

```
1.23 x 10^10
3.45 x 10^4
------------
   ? x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas

Addition

```
1.23     x 10^10
0.000345 x 10^10
----------------
   ?     x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas

Addition

```
1.23     x 10^10
0.000345 x 10^10
----------------
   ?     x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas

Addition

```
1.23     x 10^10
0.000345 x 10^10
----------------
1.230345 x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas

Addition

```
1.23      x 10^10
0.000345 x 10^10
----------------
1.230345 x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas
- Round

Addition

```
1.23     x 10^10
0.000345 x 10^10
----------------
1.23     x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas
- Round

Addition

```
1.23     x 10^10
0.000345 x 10^10
----------------
1.23     x 10^?
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas
- Round
- Update exponent of result

Addition

```
1.23      x 10^10
0.000345 x 10^10
----------------
1.23      x 10^10
```

- Convert to same exponent
  - Compare exponents, shift mantissas
- Add mantissas
- Round
- Update exponent of result

Extra steps compared to integer/fixed point

Addition - exponent change

```
 1.23 x 10^10
 9.99 x 10^10
-------------
    ? x 10^?
```

Addition - exponent change

```
 1.23 x 10^10
 9.99 x 10^10
-------------
    ? x 10^?
```

- Already same exponent
- Add mantissas

Addition - exponent change

```
 1.23 x 10^10
 9.99 x 10^10
-------------
11.22 x 10^?
```

- Already same exponent
- Add mantissas
  - overflow

Addition - exponent change

```
 1.23 x 10^10
 9.99 x 10^10
-------------
1.122 x 10^?
```

- Already same exponent
- Add mantissas
    - overflow
- Shift mantissa

Addition - exponent change

```
1.23 x 10^10
9.99 x 10^10
-------------
1.12 x 10^?
```

- Already same exponent
- Add mantissas
  - overflow
- Shift mantissa
- Round

Addition - exponent change

```
 1.23 x 10^10
 9.99 x 10^10
-------------
 1.12 x 10^11
```

- Already same exponent
- Add mantissas
  - overflow
- Shift mantissa
- Round
- Update exponent

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
       ? x 10^?
```

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
       ? x 10^?
```

- Exponents aligned
- Add mantissas (**note**: subtraction)

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 0.00001 x 10^?
```

- Exponents aligned
- Add mantissas (**note**: subtraction)

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 0.00001 x 10^?
```

- Exponents aligned
- Add mantissas (**note**: subtraction)
- Normalize!!!

Leading One Detection

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 1.00000 x 10^-5 x 10^?
```

- Exponents aligned
- Add mantissas (**note**: subtraction)
- Normalize!!!

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 1.00000 x 10^-5 x 10^?
```

- Exponents aligned
- Add mantissas (**note**: subtraction)
- Normalize!!!
- Update exponent

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 1.00000 x 10^-5 x 10^10
```

- Exponents aligned
- Add mantissas (**note**: subtraction)
- Normalize!!!
- Update exponent

Addition - LOD

```
 1.23456 x 10^10
-1.23455 x 10^10
----------------
 1.00000 x 10^5
```

- Exponents aligned
- Add mantissas (**note**: subtraction)
- Normalize!!!
- Update exponent
- Final exponent

Variants

- Double precision
  - 1 sign, 11 exponent, 52 mantissa: 64 bits
  - DR: $\approx 10^{600}$

Variants

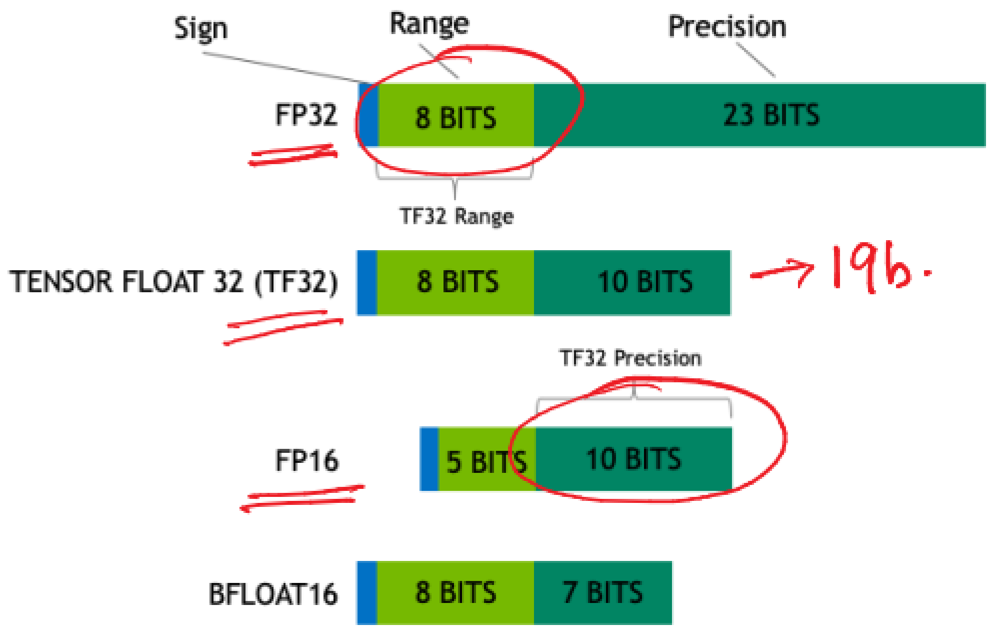
- Double precision
  - 1 sign, 11 exponent, 52 mantissa: 64 bits
  - DR: $\approx 10^{600}$
- Half precision
  - 1 sign, 5 exponent, 10 mantissa: 16 bits
  - DR: $\approx 10^{9}$ → DR of 32b int.

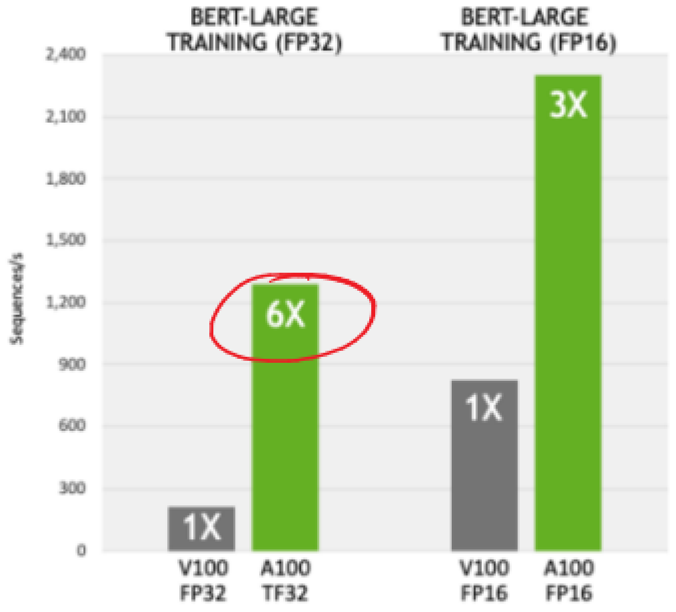
Variants

- Double precision
  - 1 sign, 11 exponent, 52 mantissa: 64 bits
  - DR: $\approx 10^{600}$
- Half precision
  - 1 sign, 5 exponent, 10 mantissa: 16 bits
  - DR: $\approx 10^{9}$
- Bfloat16
  - 1 sign, 8 exponent, 7 mantissa: 16 bits
  - DR: $\approx 10^{77}$! → 32b fp.
  - Easy conversion from single-precision
  - Much lower *precision* (*eg* $\pi \to 3.140625$)

Example: TensorFloat-32

Src: nvidia.com -- NVIDIA -- May 2020





BERT Large Training (FP32 & FP16) measures Pre-Training phase, uses PyTorch including (2/3) Phase1 with Seq Len 128 and (1/3) Phase 2 with Seq Len 512, V100 is DGX1 Server with 8xV100, A100 is DGX A100 Server with 8xA100, A100 uses TF32 Tensor Core for FP32 training

Hardware complexity

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

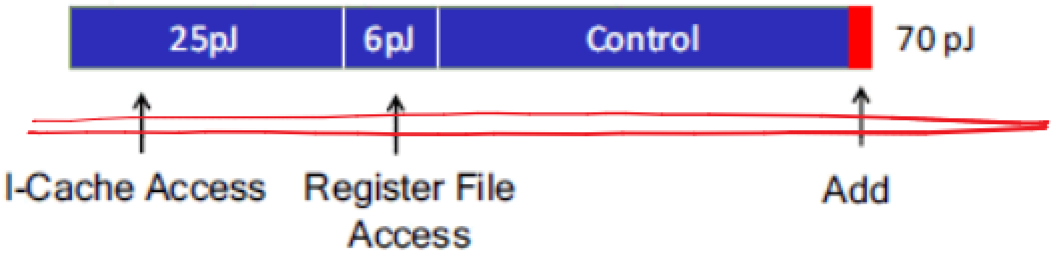
30x

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

9x

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

Instruction Energy Breakdown



Src: Horowitz, ISSCC 2014

Any Question…

Thank you