# HW/SW Co-Design Lab

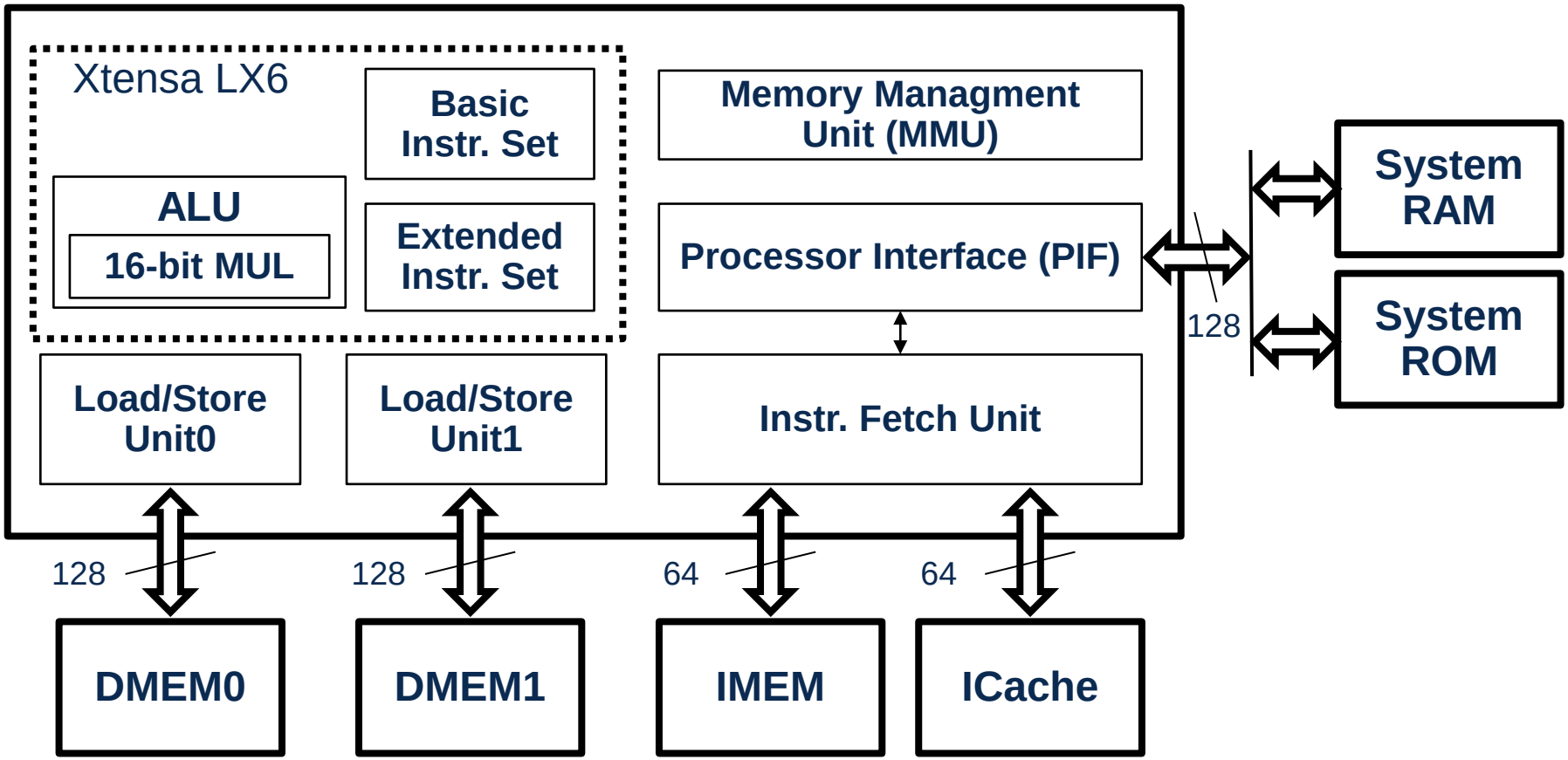
Seminar 2

SS 2024

# CORE FEATURES

# coreLX6_hwswcd





# coreLX6_hwswcd Memories



| Memory | Size | Start address |
|---|---|---|
| Instruction RAM (IMEM) | 64 kB | 0x5FFF0000 |
| Data RAM 0 (DMEM0) | 32 kB | 0x5FFE0000 |
| Data RAM 1 (DMEM1) | 32 kB | 0x5FFE8000 |
| Instruction Cache (ICache) | 16 kB | -- |
| System RAM | 4 MB | 0x60000000 |
| System ROM | 128 kB | 0x40000000 |

→ For parallel access, place important global variables/arrays in different DMEMs, e.g.:

```
int data_array[8]__attribute__((section(".dram0.data")));
```

or

```
int *data_array = (*int)0x5FFE8200;
```

→ Otherwise the compiler automatically places the variables in System RAM.

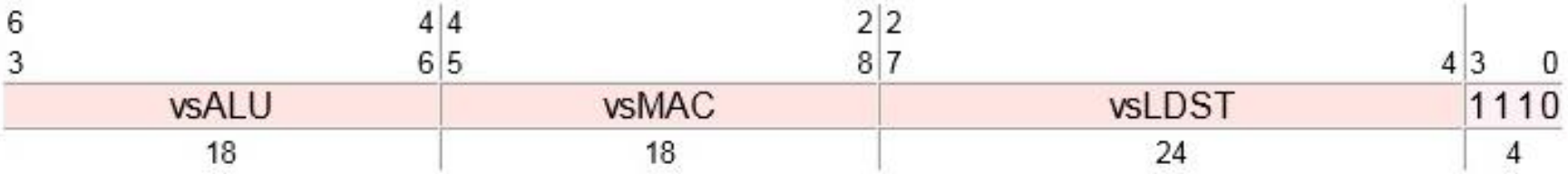
Basic and most used TIE language constructs:

- Immediate Range
- Table
- State
- Register File
- Operation
- Schedule
- Instruction Format
- Slot Opcodes
- TIE Built-in Modules
- Function

→ **Documentation TIE-Doc.pdf**

- FLIX: Flexible Length Instruction Extension
  - Combine operations into one instruction word of flexible width
    - ➔ VLIW: Very Large Instruction Word
  - Allowing the processor to execute several operations simultaneously

Format flix_x64




TIE code

```
format flix_x64 64 {slot_0, slot_1, slot2}
slot_opcodes slot_0 {vsALU}
slot_opcodes slot_1 {vsMAC}
slot_opcodes slot_2 {vsLDST}
```

*Requirements:*

- At least opt. level –O2
- Enable optimization alias restrict: –OPT:alias=restrict

# TASK 1: FIR FILTER

# FIR Filter



- FIR filter: $y_n = \sum_{i=0}^{N-1} b_i x_{n-i}$

```
void fir_C(short *in, int *out, int len) {
   int n, i;
   for (n = 0; n < len+7; n++) {
      for (i = 0; i < 8; i++)
         out[n] += in[n+7-i]*coeff[i];
   }
}
```

- Hot-Spots
  - Multiply/Accumulate Operation (MAC)
  - Load input values/coefficients
  - Store output values

- Different approaches:
  1) Performing MAC on one input element with one coefficient
  2) Performing MAC on multiple elements in parallel ➔ SIMD
  3) Separated instructions for Load, MAC and Store

# Approach 1



Performing MAC on one input element with one coefficient by applying Fusion

C code:

```
for (n = 0; n < len+7; n++)  {
    for (i = 0; i < 8; i++)
        FIR_MAC(out[n], in[n+7-i], coeff[i]);
}
```

TIE code:

```
operation FIR_MAC {inout AR acc, in AR a, in AR b} { } {
    wire [31:0] product = TIEmul(a[15:0], b[15:0], 1'b1);
    assign acc = acc + product;
}
```

FIR_MAC produces the output in one clock cycle. ➔ fused operations

# Approach 2



- Performing MAC on multiple elements in parallel ➔ SIMD
- FIR_MAC_SIMD produces output of the second for-loop in one clock cycle ➔ fused and parallel operations
- Recommended schedule to reduce c.p. ➔ Addition is done 1 cycle after multiplications

```
regfile VR_fir 128 8 vrf

operation FIR_MAC_SIMD {out AR acc, in VR_fir a, in VR_fir b} { } {
   // 8x signed multiplications
   wire [31:0] product0 = TIEmul(a[127:112], b[15:0], 1'b1);
   wire [31:0] product1 = TIEmul(a[111:96], b[31:16], 1'b1);
   wire [31:0] product2 = TIEmul(a[95:80], b[47:32], 1'b1);
   wire [31:0] product3 = TIEmul(a[79:64], b[63:48], 1'b1);
   wire [31:0] product4 = TIEmul(a[63:48], b[79:64], 1'b1);
   wire [31:0] product5 = TIEmul(a[47:32], b[95:80], 1'b1);
   wire [31:0] product6 = TIEmul(a[31:16], b[111:96], 1'b1);
   wire [31:0] product7 = TIEmul(a[15:0], b[127:112], 1'b1);

   // addition of the 8 products
   assign acc = TIEaddn(product0, product1, product2, product3,
                        product4,product5,product6,product7);
}

schedule FIR_MAC_SIMD_SCHED {FIR_MAC_SIMD} {
   def acc 2;
}
```

# Approach 3



- Separate Load, MAC, and Store operations into different TIE instructions
  - Load 8 elements: FIR_LD
  - Perform on 8 elements in parallel and store result in every clock cycle: FIR_CALC_ST
- Hold coefficients in Look-Up table ➔ access within 1 clock cycle
- Pipelining by using Schedule or additional registers

**C code**

```
FIR_LD();

for(n=0; n<((len+14)>>3)-2; n++) {
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();
   FIR_CALC_ST();

   FIR_LD();
}

FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
FIR_CALC_ST();
```

**TIE code**

```
operation FIR_LD {} {out VAddr, in MemDataIn128, inout ptr_in,
                     inout shad_in128, out in128_2} {
   assign VAddr = ptr_in;
   assign ptr_in = ptr_in + 16;
   assign shad_in128 = MemDataIn128;
   assign in128_2 = shad_in128;
}
```

# Approach 3



```
operation FIR_CALC_ST {} {out VAddr, out MemDataOut32, out StoreByteDisable, inout ptr_out,
                          inout in128_1, inout in128_2, inout st_cnt,
                          inout product0, inout product1, inout product2, inout product3,
                          inout product4, inout product5, inout product6, inout product7} {
  assign product0 = TIEmul(in128_1[127:112], coeff[0], 1'b1);
  assign product1 = TIEmul(in128_1[111: 96], coeff[1], 1'b1);
  assign product2 = TIEmul(in128_1[ 95: 80], coeff[2], 1'b1);
  assign product3 = TIEmul(in128_1[ 79: 64], coeff[3], 1'b1);
  assign product4 = TIEmul(in128_1[ 63: 48], coeff[4], 1'b1);
  assign product5 = TIEmul(in128_1[ 47: 32], coeff[5], 1'b1);
  assign product6 = TIEmul(in128_1[ 31: 16], coeff[6], 1'b1);
  assign product7 = TIEmul(in128_1[ 15:  0], coeff[7], 1'b1);

  assign in128_1 = {in128_2[15:0], in128_1[127:16]};
  assign in128_2 = {16'h0, in128_2[127:16]};

  assign VAddr = ptr_out;
  assign ptr_out = ptr_out + 4;
  assign ptr_out_kill = st_cnt;
  assign StoreByteDisable = {16{st_cnt}};
  assign st_cnt = 1'b0;

  // addition of the 8 products
  assign MemDataOut32 = TIEaddn(product0, product1, product2, product3,
                                product4, product5, product6, product7);
}
```

8-fold SIMD MULs

Shift/Prepare next values

Interface/Pointer increment

Write to memory

# TASK 2: FFT/IFFT

# Discrete Fourier Transformation DFT



- FIR filter:

$$y_n = \sum_{i=0}^{N-1} b_i x_{n-i}$$

- DFT:

$$X_m = \sum_{k=0}^{N-1} x_k e^{-j2\pi \frac{km}{N}}$$

- DFT is comparable with FIR filter:
  - Input values x
  - Coefficients become complex: $b_i$ ➔ $e^{-j2\pi\frac{km}{N}}$

# DFT ➔ FFT/DIT

- Cooley-Tukey algorithm for a Decimation in Time (DIT) radix-2 FFT

$$X_m = \sum_{k=0}^{N-1} x_k e^{-j2\pi \frac{km}{N}}$$

$$X_m = \sum_{k=0}^{\frac{N}{2}-1} x_{2k} e^{-j2\pi \frac{(2k)m}{N}} + \sum_{k=0}^{\frac{N}{2}-1} x_{2k+1} e^{-j2\pi \frac{(2k+1)m}{N}}$$

$$X_m = \sum_{k=0}^{n-1} x_{even,k} e^{-j2\pi \frac{km}{n}} + e^{-j\pi \frac{m}{n}} \sum_{k=0}^{n-1} x_{odd,k} e^{-j2\pi \frac{km}{n}}$$

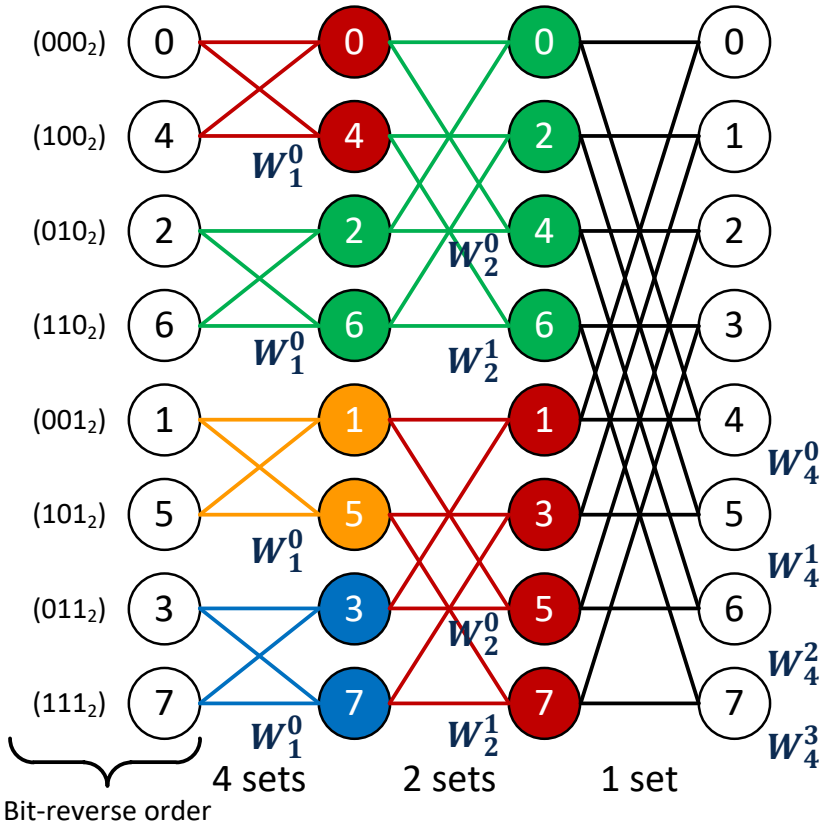
**N=2n, N is power of 2**

**Twiddle factor:** $W_n^m = e^{-j\pi \frac{m}{n}}$

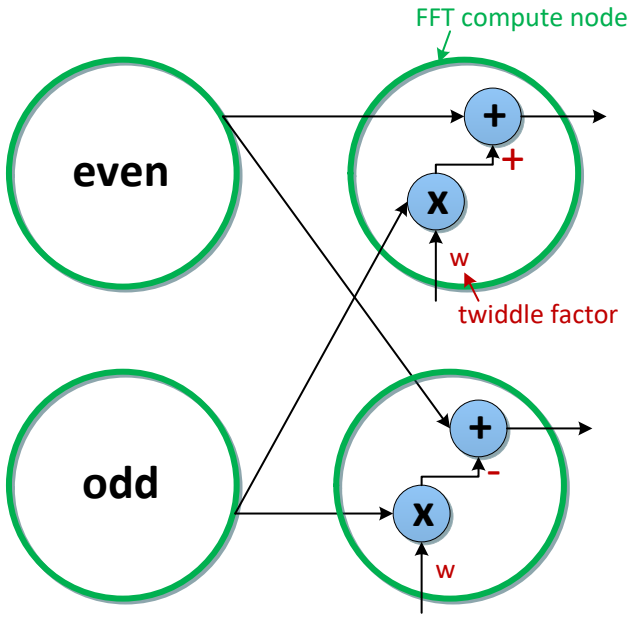
# Butterfly Network



## Decimation in Time




## DIT Compute Node



# Calculating Twiddle Factors



$$W_n^m = e^{-j\pi\frac{m}{n}}$$

n = 1

$$W_1^0 = 1$$

n = 2

$$W_2^0 = 1$$
$$W_2^1 = -j$$

n = 4

$$W_4^0 = 1$$
$$W_4^1 = \frac{1}{\sqrt{2}}(1 - j)$$
$$W_4^2 = j$$
$$W_4^3 = -\frac{1}{\sqrt{2}}(1 + j)$$

n … number of interleaved butterflies

- Sande-Tukey algorithm for a Decimation in Frequency (DIF) radix-2 FFT

$$X_m = \sum_{k=0}^{N-1} x_k e^{-j2\pi \frac{km}{N}}$$

$$X_m = \sum_{k=0}^{\frac{N}{2}-1} x_k e^{-j2\pi \frac{km}{N}} + \sum_{k=\frac{N}{2}}^{N-1} x_k e^{-j2\pi \frac{km}{N}}$$

$$X_m = \sum_{k=0}^{\frac{N}{2}-1} [x_k + x_{k+\frac{N}{2}}(-1)^m] e^{-j2\pi \frac{km}{N}}$$

# FFT DIF

- Even and odd indexed frequency values

$$X_{2m} = \sum_{k=0}^{n-1} [x_k + x_{k+n}] e^{-j2\pi \frac{km}{n}}$$

$$X_{2m+1} = \sum_{k=0}^{n-1} [x_k - x_{k+n}] e^{-j\pi \frac{k}{n}} e^{-j2\pi \frac{km}{n}}$$

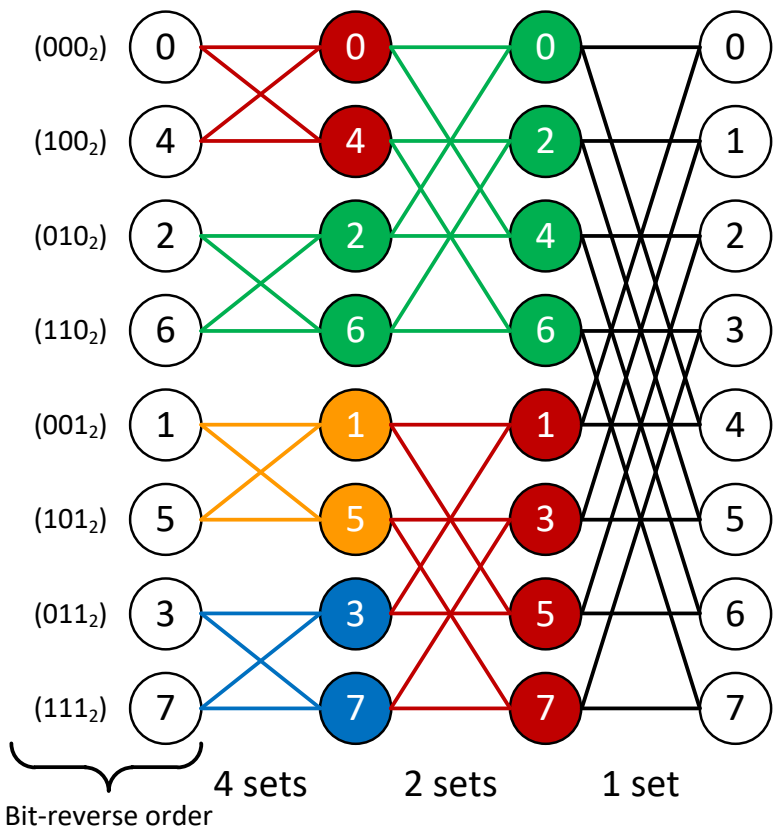
**N=2n, N is power of 2**

**Twiddle factor: W** $\boldsymbol{\frac{k}{n}} = e^{-j\pi \frac{k}{n}}$

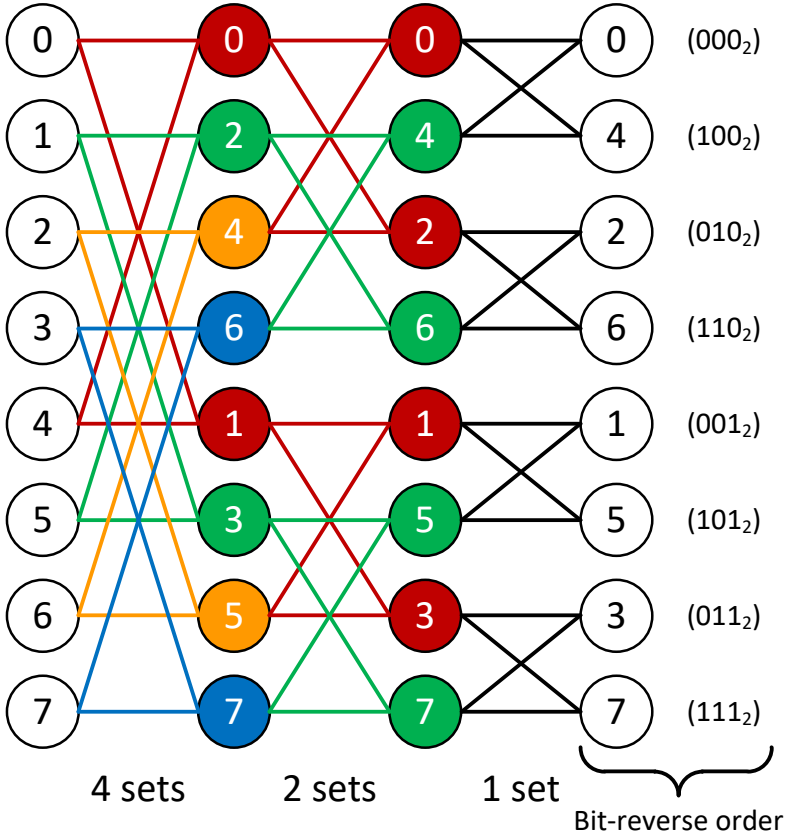
# DIT, DIF



## Decimation in Time



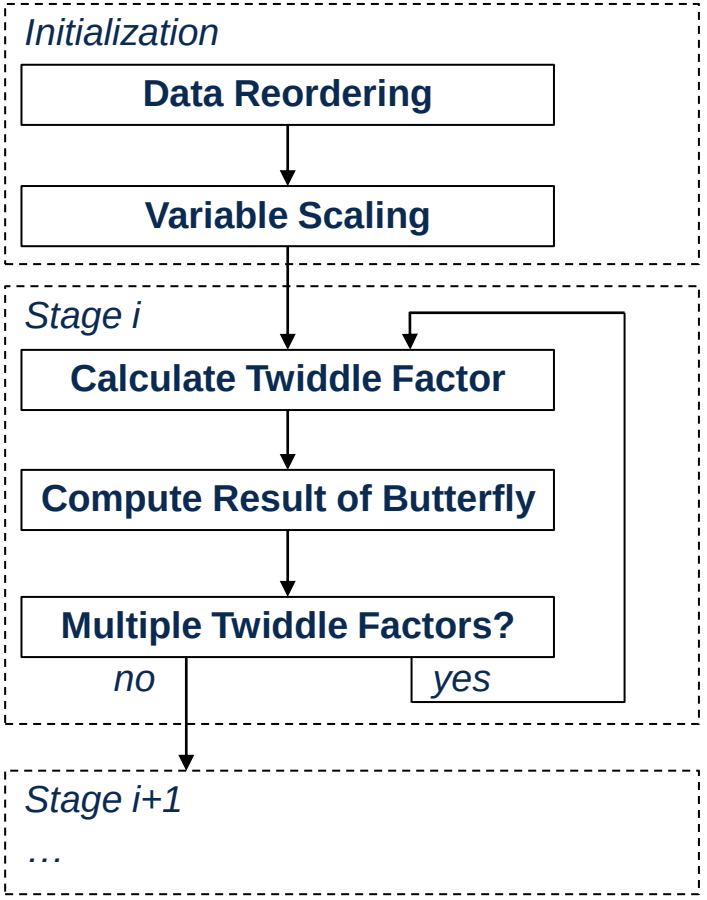

## Decimation in Frequency



# FFT: C code



- Given C code
  - Fixed integer arithmetic (16 bit real, 16 bit imaginary part)
  - Code for FFT/IFFT with any N with power of 2
  - Data Reordering
  - Variable scaling to prevent overflow



# SUMMARY

# Summary



- Tasks
  - Introduce TIE instructions in hot-spots of FFT/IFFT
  - Hold constants in TIE states or tables
  - Compare DIT/DIF
  - In the end, FFT should work with selected N with power of 2
- Parallelization
  - MAC operation: Butterfly Compute Node
  - SIMD: Butterflies of one stage can be computed independently
  - FLIX: Load input data simultaneously by using the two available DMEMs of coreLX6_hwswcd

➢ Submit **report** + **exported workspace** until September 30, 2024 via email to **viktor.razilov@tu-dresden.de**

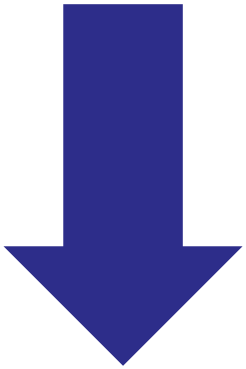
# Student Job Opportunity



Revision of this lab for SS 2025

- Research on (Open-Source?) RISC-V ASIP tools
- Implementation of ASIP
- Documentation of results

Contact: viktor.razilov@tu-dresden.de

More student opportunities can be found @ https://www.vodafone-chair.org/teaching+opportunities#opportunities