FFT Block Implementation

By-
Kothadiya Prince
Sri Aditya Deevi

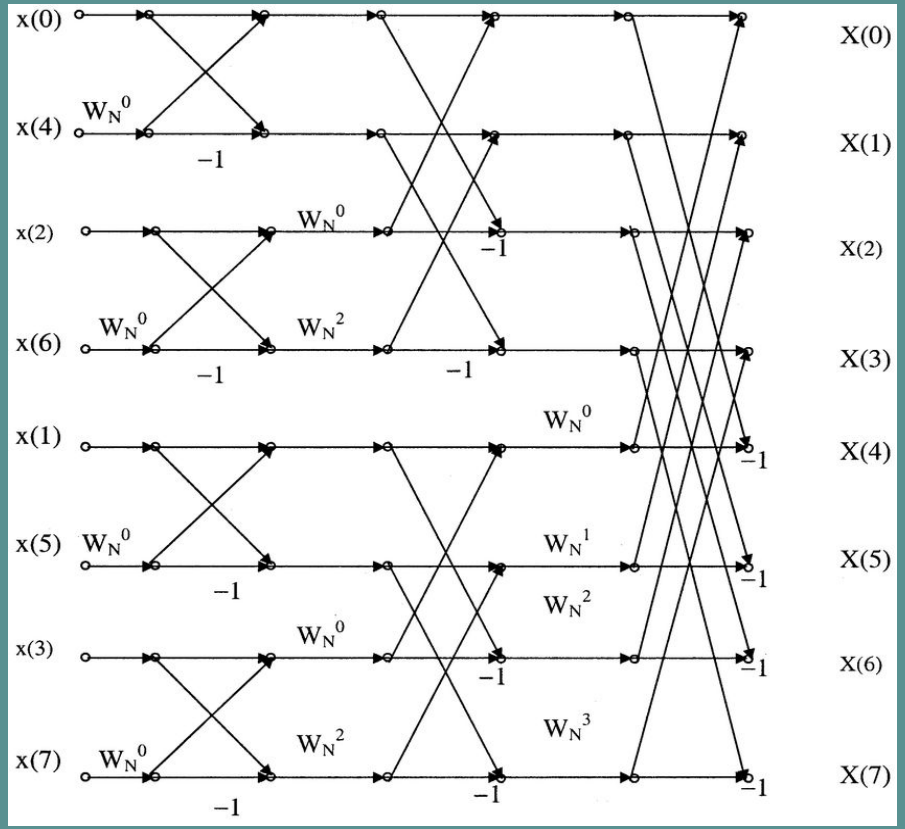
Discrete Fourier Transform (DFT)

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad W_N = e^{-j\frac{2\pi}{N}}$$

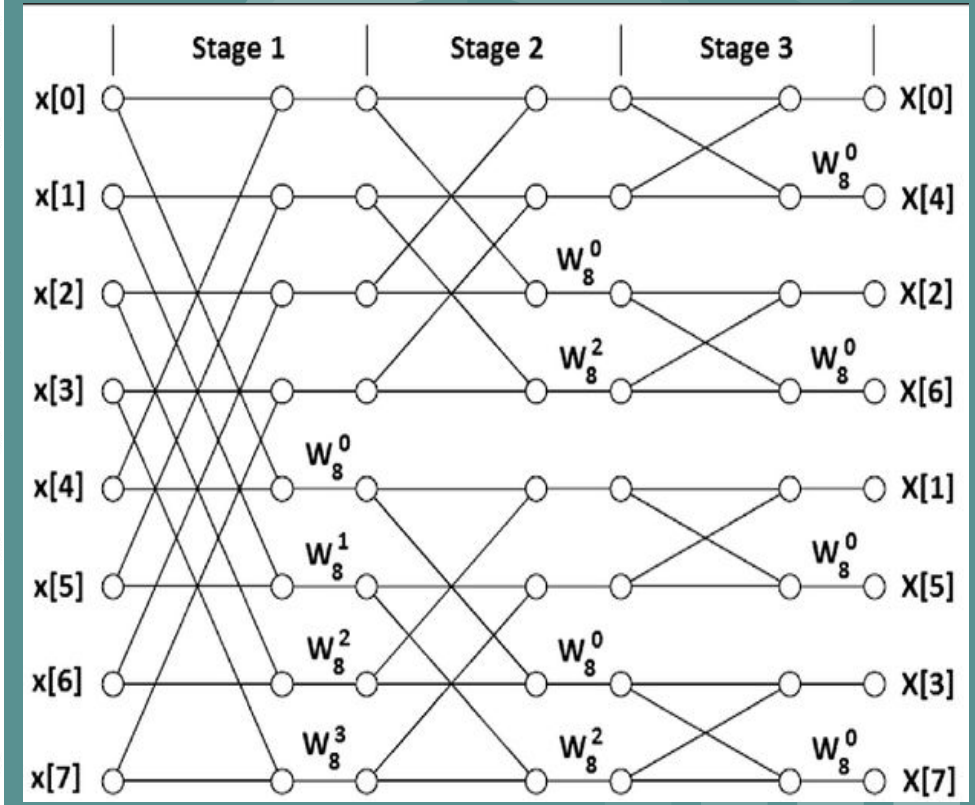
- Where, W_N = Twiddle Factor
- For efficient and speedy computation of DFT, FFT algorithm is used.
- $N = 2^v$

Number of Points, N	Complex Multiplications in Direct Computation, N^2	Complex Multiplications in FFT Algorithm, $(N/2)\log_2 N$	Speed Improvement Factor
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1,024	80	12.8
64	4,096	192	21.3
128	16,384	448	36.6
256	65,536	1,024	64.0
512	262,144	2,304	113.8
1,024	1,048,576	5,120	204.8

Decimation in Time (DIT)



Decimation in Frequency (DIF)



A Note on Implementation

- **Vivado® High-Level Synthesis (HLS)** software can be used to accelerate IP creation by enabling C, C++ and System C specifications to be directly targeted into Xilinx programmable devices without the need to manually create RTL (Co-simulated).

- Availability of FPU (Floating Point Unit) in Shakti enables us to implement the Floating point version of FFT Block, which gives us more accurate and precise results.

- Control Logic for AXI interface of Shakti can be implemented as an FSM.

STEPS INVOLVED IN DESIGN

1. FFT Code written in C++ (HL lang.)
2. Simulation and Functional verification
3. RTL (BSV) Generation, Co-Simulation and HDL Wrapper Creation (SoftCore IP)
4. Interfacing with AXI Bus of Shakti Processor as a Peripheral
5. API Creation in Shakti SDK

Review of FFT Code

```
void FFT0(int FFT_stage,int pass_check,int index_shift,int pass_shift,data_comp data_IN[N], data_comp data_OUT[N]){

        int butterfly_span=0,butterfly_pass=0;
        FFT_label1: for (int i = 0; i < N/2; i++) {
                int index = butterfly_span << index_shift;
                int Ulimit = butterfly_span + (butterfly_pass<<pass_shift);
                int Llimit = Ulimit + FFT_stage;
                data_comp Product = W[index] * data_IN[Llimit];//calculate the product
                data_OUT[Llimit] = data_IN[Ulimit]-Product;
                data_OUT[Ulimit] = data_IN[Ulimit]+Product;
                if (butterfly_span<FFT_stage-1){
                        butterfly_span++;
                } else if (butterfly_pass<pass_check-1) {
                        butterfly_span = 0;     butterfly_pass++;
                } else {
                        butterfly_span = 0;     butterfly_pass=0;
                }
        }
}
```

```
void FFT(data_comp data_IN[N], data_comp data_OUT[N]){
        static data_comp data_OUT0[N];
        static data_comp data_OUT1[N];
        static data_comp data_OUT2[N];
        static data_comp data_OUT3[N];
        static data_comp data_OUT4[N];

        static data_comp xin[N];
        static data_comp xout[N];

        for (int i=0; i<N; i++) xin[i] = data_IN[i];

        bitreverse(xin, data_OUT0);  //calculate bitreverse order

        FFT0(1,16,4,1,data_OUT0,data_OUT1); //calculate the FFT
        FFT0(2,8,3,2,data_OUT1,data_OUT2);
        FFT0(4,4,2,3,data_OUT2,data_OUT3);
        FFT0(8,2,1,4,data_OUT3,data_OUT4);
        FFT0(16,1,0,5,data_OUT4,xout);

        for (int i=0; i<N; i++) data_OUT[i] = xout[i];
}
```

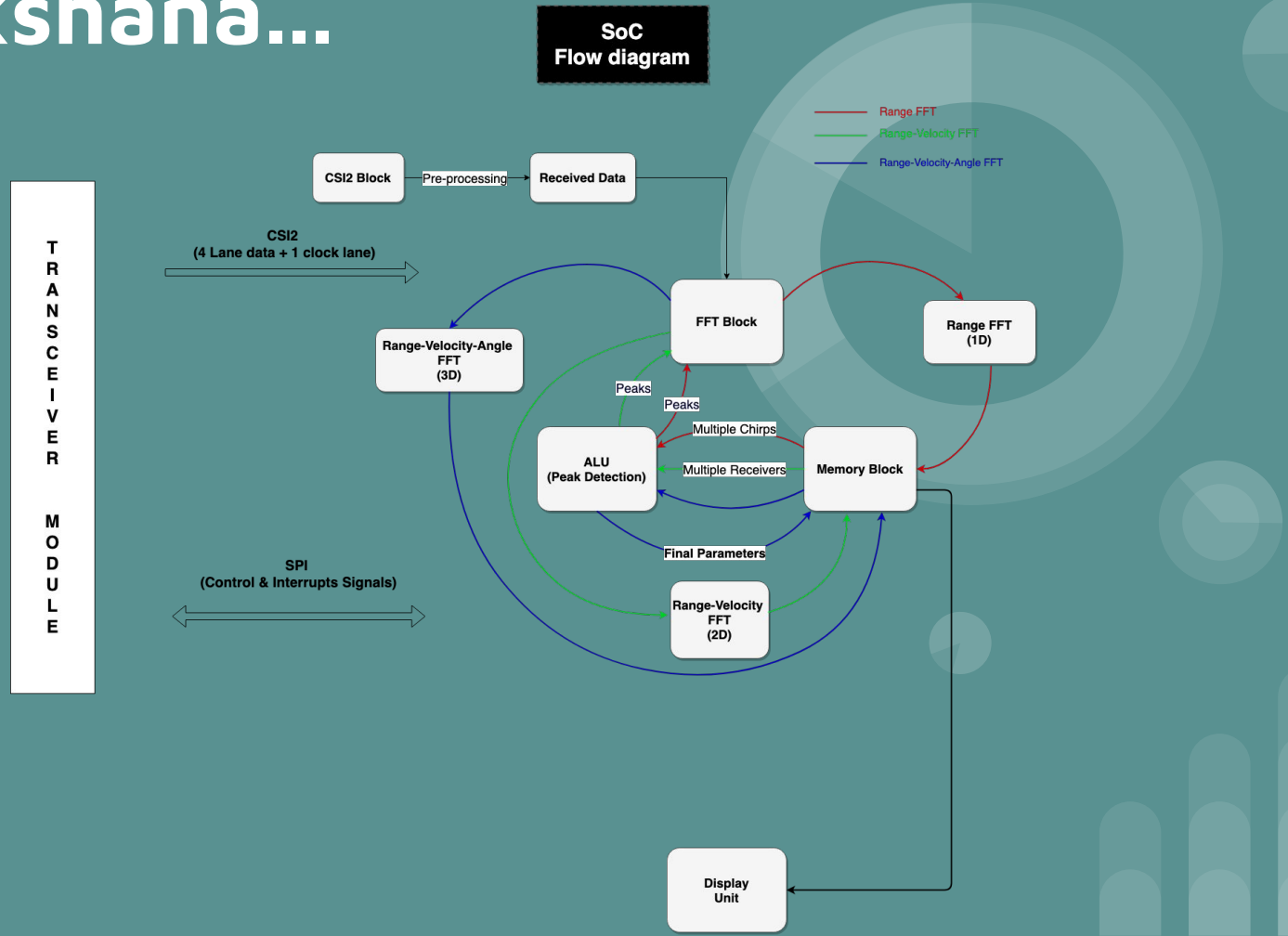
C++ Libraries

- <math.h>
- <complex>
- <ap_fixed.h>
- <iostream>
- <stdlib.h>
- <fstream>

Block Design

- IP of FFT block can be created using Vivado HLS tools.
- Importing IP's of different Blocks into Vivado HLS to make complete Block design
- Complete Block diagram can be used for synthesis, RTL code generation and FPGA implementation.

In Ikshana...


SoC
Flow diagram
Range FFT
Range-Velocity FFT
Range-Velocity-Angle FFT
TRANSCEIVER MODULE
CSI2 Block
Pre-processing
Received Data
CSI2
(4 Lane data + 1 clock lane)
FFT Block
Range FFT
(1D)
Range-Velocity-Angle
FFT
(3D)
Peaks
Peaks
Multiple Chirps
ALU
(Peak Detection)
Multiple Receivers
Memory Block
Final Parameters
SPI
(Control & Interrupts Signals)
Range-Velocity
FFT
(2D)
Display
Unit

THANK YOU!

References

- https://www.youtube.com/watch?v=LC5XByTjzzY
- https://www.youtube.com/watch?v=R-msBFn6r88
- https://gitlab.com/chandrachoodan/teach-fpga/-/blob/master/01-fft/vhls/float/fft.cpp FFT module on FPGA
- Implementation of Fast Fourier Transform (FFT) on FPGA using Verilog HDL