Assignment 7: Streaming Pipelined FFT

In this assignment, you will implement a hardware-pipelined, streaming Fast Fourier Transform (FFT) in SystemVerilog. You will also implement quantization to convert the floating-point C reference model to fixed-point hardware, and verify correctness using UVM.

Setup:

Compile and run the C-code to generate the input and output files for your simulation and constant declarations for your FFT:

- **gcc fft_quant.c -o fft_quant**
- **./fft_quant**

Implementation

- Build a **streaming pipelined FFT module** in SystemVerilog based on the provided C reference code. Requirements:
    - Read **real and imaginary inputs** from FIFOs continuously.
    - Implement an **N-point FFT** (e.g., N = 16, 32, etc).
    - Output **transformed real and imaginary values** to separate FIFOs.
    - Implement **stage-wise pipelining** using generate-for loops for butterfly operations. Each stage should compute a butterfly per clock cycle.
    - Include a **bit-reversal module** to reorder inputs before FFT stages.
    - Design the pipeline so that after initial fill latency, **one FFT sample is output per clock cycle**.
- Implement **fixed-point quantization** throughout the FFT:
    - Quantize inputs, intermediate stage values, and outputs (e.g., 14–16 bits).
    - Apply consistent scaling to avoid overflow.
- Provide **parameterized design** for:
    - Data width for real/imaginary values.
    - FFT size (N).
    - FIFO depth.
- Implement a **UVM verification environment**:
    - Generate fixed-point real/imaginary input sequences using UVM sequences.
    - Drive inputs into the FFT input FIFOs and capture outputs from output FIFOs.
    - Compare outputs against the software FFT reference to check quantization accuracy.
    - Include functional coverage for all FFT stages and butterfly operations.
- Measure **throughput and latency**:

    - Determine clock cycles for pipeline fill and for continuous sample output.
    - Compute throughput in **samples per second** assuming a 100 MHz clock (10 ns period).

Verification:

- In your UVM testbench:
    - Generate **streaming input vectors** (random or deterministic) for real and imaginary parts.
    - Feed inputs continuously into the FFT module's input FIFOs.
    - Capture streaming outputs and compare to the software reference.
    - Validate both **functional correctness** and **timing behavior** of the pipeline.
- Testbench parameters:
    - Data width: 32-bit (real and imaginary).
    - FIFO depth: 16 elements.
    - Clock frequency: 100 MHz (10 ns period).
- Create a simulation script (fft_sim.do) to compile SystemVerilog and UVM models, generate waveforms, and run the simulation.

Compare Results:

- Compare hardware FFT outputs vs. C reference:
    - Report **bit-true accuracy**.
    - Quantify **quantization error** and its effect on FFT magnitude and phase.
- Evaluate **throughput and latency**:
    - Clock cycles for N-point FFT computation.
    - Effective samples-per-second rate after pipeline fills.

Reporting:

Submit a PDF report including:

- **Simulation Results**
    - Clock cycles for N-point FFT computation
    - Bit-true comparison with software reference
    - Functional coverage for all stages
- **Synthesis Results** (if applicable)
    - Maximum clock frequency
    - Registers / LUTs / Logic Elements
    - Memory utilization
    - Multipliers / DSP usage
    - Worst path timing analysis
    - RTL schematic / architecture diagram

    - Throughput (samples per second)
- **Documentation**
    - Description of pipeline architecture and design choices
    - Input/output test vectors used for verification
    - Waveform screenshots showing streaming operation

Turn in your designs with the report. Your file should be zipped, and should include the SystemVerilog files, synthesis, simulation, and input/output files. **PLEASE MAKE SURE TO REMOVE ALL THE INTERMEDIATE WORK DIRECTORIES.**