HW 6: 1D-FFT Computation over a Serial Port
CMPE 415
UMBC

April 24, 2016

For this lab, you will implement serial-port communication on an FGPA to load a data vector from the computer, compute an FFT (a discrete-time version of a Fourier transform), and transfer the results back to the computer. You will need to use the Xilinx FFT core and serial UARTs that allow communication with a PC. You will need to design a state-machine to control these components.

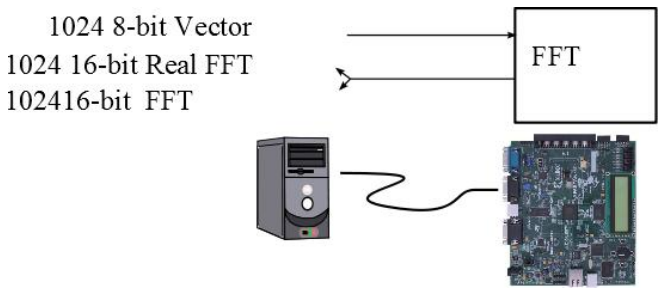


Figure 1: Hardware. Input data should be sent to the FPGA though the serial port. The FPGA should compute the FFT result and send the result (real and imaginary components) back to the computer through the serial port. For this part of the project, you can implement a 32-point computation to speed simulation. You will be required to implement the full size on the implementation later.

1 Description

I provided some in-class explanation of this project and the FFT computation and will not repeat everything here. The data should be sent in and read back through 1-bit serial interface.

A FFT is computed by performing an FFT transformation a series of data points. The data should be streamed in one data point at a time. You should alternate the real and imaginary data words when it is sent back to the computer: 16-bits real (high byte then low byte), 16 bits imaginary (high byte then low byte).

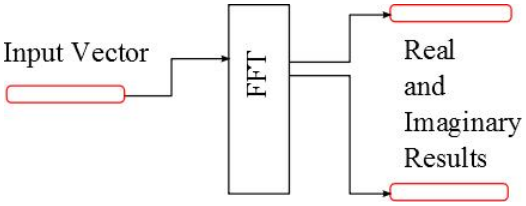


Figure 2: FFT computation. The FFT core accepts 16-bits at a time. Each of the values in a vector is sent in sequentially. Generally, an FFT computation computes a real and imaginary data results based on real and imaginary data. Since the input is real data, zero may be sent into the imaginary data input for this FFT.

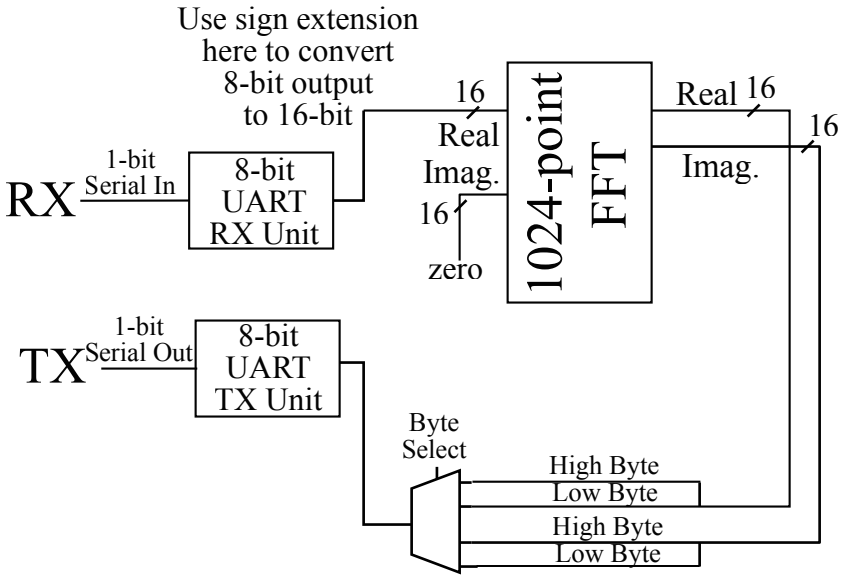


Figure 3: Register Data Flow.

2 Design Approach and Details

You will need to use the Xilinx IP Core Generator to generate an FFT modules.

Download a serial, Write a case-statement-based state machine to control the modules and data flow.

You will need to use the Xilinx IP Core Generator to generate two 1-D FFT modules and two memories. Download a serial, unbuffered UART from the Internet (cite source). Write a case-statement-based state machine to control the modules and data flow.

2.1 UART (universal asynchronous receiver-transmitter) for Serial Communication with computers and other devices

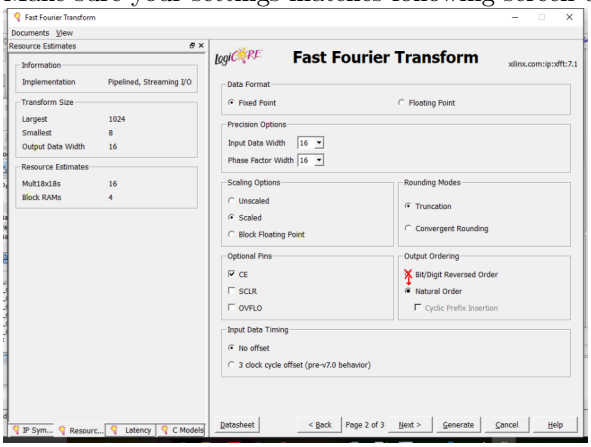
- Download uart.v,uart_loopback.v, uart_loopback_tb.v, and hw6.ucf file from course website as a starting point to create a project with serial communication. You should at least simulate it, though testing the serial communication in hardware before integrating other parts is recommended. (For those peeking at the code in the loopback, it does not use the 3-always-block style that was recommended, but the style is non-the-less allowed if done correctly).
- Note, the provided testbench reads data from a file. To use it, you'll need to provide a simple text file with multiple lines, each with two characters representing the hex representation of a byte. An example input file is provided.

2.2 FFT

2.2.1 TO generate FFT module using Xilinx's Core Generator, following the following steps.

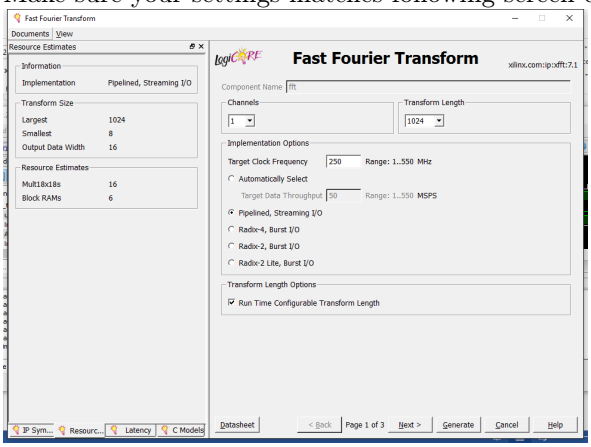
- In ISE, go to Menu → Project →New Source...
  - A window labeled **New Source Wizard** appears
  - From the list, select **IP (CORE Generator & Architecture Wizard)**
  - Enter a filename **my_fft.v**
  - Click **Next**
  - After new window appears select the tab **View by Name**
  - From the list select **Fast Fourier Transform**
  - Click **Next**
  - Finally click **Finish**
- You will proceed through the configuration wizard for generating your IP core.

– Make sure your settings matches following screen capture and hit next



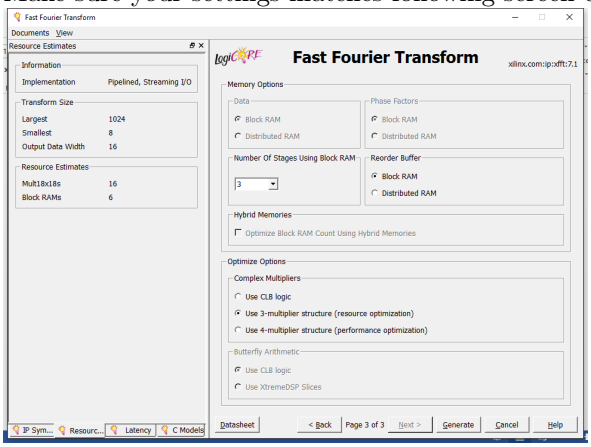
The CE (clock enable option) will allow your state machine to control the progression and data flow through this module

– Make sure your settings matches following screen capture and hit next



* Note the documentation can be accessed through the Datasheet button. You can access this datasheet again later by restarting the IP Core Generator process, which is done by opening the module from the design tree. It is also available at http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf
* (The Run-Time Configurable Transform Length allows you to perform operations on signals that are up to or much less than 1024-point, without having to change FPGA hardware.It also allows your to implement a smaller transform for shorter simulation.)

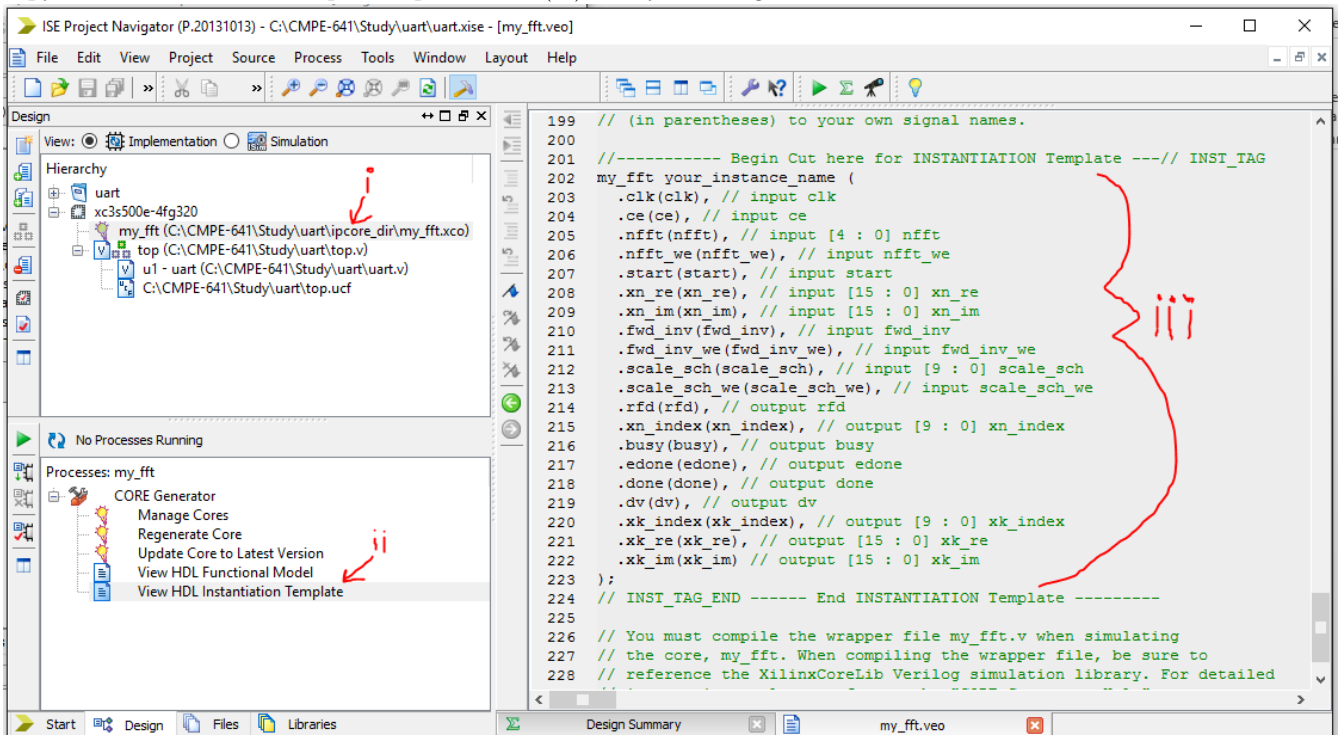
– Make sure your settings matches following screen capture and hit generate.



– Wait for core generator to generate FFT module, it may take several minutes depending on your computer.

2.2.2 Using the Xilinx core generators "FFT" module in your design

- You now may instantiate the generated core in your design. To do this, select (i) , then (ii) per the following figure. Then, copy the instantiation example template code (iii) into your design.



In	Out	Size	Desc
.clk		1	use a system clock
.ce		1	think of this as an active-low pause, you can use it to pause the FFT module for example while you are waiting for data to arrive from the uart, this is perhaps the most flexible signal to use for data flow control – set it high one cycle at a time to advance the FFT process in loading and unloading data, leave high other times
.start		1	use to start a transform cycle by setting it to high for one clock cycle, or leave high for a continous operation (see note for rfd)
	.rfd	1	when rfd is high the fft is loading input the input data words, to load data you can present data to .xk_re, wait for rfd and then immediately lower ce until the next byte is available from the input uart, then raise ce for one clk cycle, repeat until all data is loaded. If the start signal is not left high, rfd will goto zero when the all the input data required is load...the same time that the input index counter resets to zero. If the start signal is left high, rfd may stay high.
.xn_re		16	(real input) , sign extend the input byte (8 bits) to 16 bits
	.xn_index	10	indicates at what index the input value is loaded (goes from 0 to 2^{nfft}-1 and repeats)
	.edone	1	advance done signal goes high one clock cycle early to indicate the cacluation is finishing on this cycle, the next cycle busy goes low
	.xk_index	10	output index counter
	.xk_re	16	output data
	.xk_im	16	output data
.nfft		5	log2 of the fft size (e.g. 5 indicates a 32-point transform)
.nfft_we		1	must set high one clock cycle to load transform size setting nfft
Signals below are hardwired to values or ignored			
	.busy	1	indicates a calculation is in progress, which tells us enough input data has been loaded, data valid goes high when this goes low, may wait for this to go low to capture the first output
.fwd_inv		1	hardwire to 1'b1 to select a formward transform as opposed to an inverse
.fwd_inv_we		1	contols loading of a dynamic design, can hardware to 1'b0
.xn_im		16	just tie this to 16'b0
.scale_sch		10	set to 10'b0
.scale_sch_we		1	set to 1'b0
	.done	1	indicates a calculation is finished by going high for one cycle (can also use edone)
.unload		1	signals to unload results data may ignore or tie to any value
	.dv	1	specifies when some output data is ready to be unloaded, goes high when busy goes low

2.3 Control FSM and miscellaneous glue logic

- The center of control should be a case-statement-based state machine
  - Extra components or registers may be generated separately as needed/desired
  - The input data is 8 bits and should be sign-extended to match the 16-bit input of the FFT.
  - The output should be sent back by alternating real and imaginary words (16-bit words, high-byte-first).
  - Signals and Control for the FFT
    * You need to set nfft_we high for one cycle to load the transform size setting (this also reinitializes FFT core)
    * Collect the first bye from the UART and present it to the FFT module's input xn_re[15:0] using sign extension
    * To start a transform, you must set that start signal high
    * soon after, rfd will go high and xn_index will be 0 – during that cycle the data word is captured from xn_re[15:0]
    * You should immediately set ce low (such that rfd and ce overlap by only one clock cycle) to prevent loading the next word – whenever xn_index increases to N it means the previous data has been loaded at index N-1
    * Once the next word is available from the UART, you may set ce high for one clock cycle
    * Repeat the process of alternating process with setting CE high for one cycle untill all data is loaded into the FFT module
    * Set CE high again
    * wait for the signal edone to go high (only stays high for one cycle).
      · On the next cycle the first output data will arrive (xk_index will be 0) , you must set CE low to prevent the following next byte from arriving (and xk_index→ 1) before you are ready.
    * Send all four output bytes through the UART on at time by using a mux
    * Once you are ready for the next value, set CE high for one cycle
    * Repeat the process of alternating the sending process and setting CE high for one cycle until all data is unloaded and sent

- Testbenches
  - Generate a testbench to test at least one transform of at least size 32. Some examples are provided on the course website.
  - Input data should be read from disk and results should be written to disk. An example is provided for the UART Module that already reads from a file.
  - Matlab testing cod will be provided for anyone who wishes to test their design in hardware.

3 What to turn in

- Follow instructions posted on the website and Piazza. You will be expected to provide design files including a "top" module implemented in Verilog NOT SCHEMATIC.
- The top module should only have have pins rx, tx, clk_50mhz, and reset.
  - Adherence to this is required for the TA to test your design and give you points for correct implementation.
- Testbench and input files you used to test your design should be provided
- A Report including discussion of design overview, design choices, and verification will be requested

4 Bonus

- +5% Implement and demonstrate a 1024-point transform in hardware.