ĐỒ ÁN TỐT NGHIỆP

THIẾT KẾ BỘ FFT VÀ IFFT TRÊN NỀN CÔNG NGHỆ FPGA

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

AHDL	Altera Hardware Description Language
AS	Active Serial
ASIC	Application-specific Integrated Circuit
CPLD	Complex Programmable Logic Device
DE	Development and Education
DFT	Dicrete Fourier Transform
DSP	Digital Signal Processor
DTFT	Dicrete-Time Fourier Transform
EDA	Electronic Design Automation
EDIF	Electronic Design Interchange Format
FFT	Fast Fourier Transform
FPGA	Field-programmable gate array
HDL	Hardware Description Language
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
JTAG	Joint Test Action Group
LCD	Liquid crystal display
LED	Light Emitting Diode
O	Order
PAL	Programmable Array Logic
PLA	Programmable Logic Array
PLD	Programmable Logic Device
RAM	Random-access Memory
ROM	Read-only Memory
RTL	Register Transfer Level
SDF	Single-path Delay Feedback
SDRAM	Synchorous dynamic random-access Memory

SOPC	System On a Programmable Chip
SPLD	Simple Programmable Logic Device
SRAM	Static random-access Memory
SW	Switch
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

CHƯƠNG 1. TỔNG QUAN VỀ BỘ FFT/IFFT

1.1 Khái niệm DFT và FFT

1.1.1 DFT

DFT N – điểm của một tín hiệu dài – L được định nghĩa bằng DTFT được tính toán tại N tần số phân bố đều trên toàn khoảng Nyquist, $0 \leq \omega \leq 2\pi$.

$$\omega_k = \frac{2\pi k}{N},\ k = 0, 1, \ldots, N-1 \tag{1.1}$$

⇨ DFT:

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

$$\Leftrightarrow X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn} \quad ,\text{Với } k = 0, 1, 2, \ldots, N-1 \tag{1.2}$$

Trong đó:

- x(n) là ngõ vào ở miền liên tục thời gian, được phân tích tín hiệu với N điểm.
- X(k) là ngõ ra ở miền tần số.
- $W_N = e^{-j\left(\frac{2\pi}{N}\right)}$: là trọng số nhân.

DFT đảo (IDFT):

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \cdot W_N^{-kn} \quad ,\text{Với } n = 0, 1, 2, \ldots, N-1 \tag{1.3}$$

1.1.2 FFT & IFFT:

Biến đổi Fourier nhanh (FFT) là một thuật toán của biến đổi DFT của 1 chuỗi N điểm. Có rất nhiều phương pháp thực hiện khác nhau với các ưu nhược điểm riêng biệt cho từng phương pháp. Ví dụ: Radix-2 (cơ số 2), Radix-4 (cơ số 4), Radix-mix (bộ trộn cơ số),….

Khác biệt của FFT và DFT là độ phức tạp và tốc độ trong tính toán. DFT thực hiện với độ phức tạp O (N^2). Còn FFT thực hiện với độ phức tạp O ($N \log_m N$), với m là cơ số của FFT. Ví dụ: FFT cơ số 2 (Radix-2) có độ phức tạp O ($N \log_2 N$).

IFFT là bộ đảo của FFT. Ta sử dụng công thức của IDFT để thực hiện. Vẫn giống như FFT, IFFT sẽ giảm độ phức tạp so với IDFT tùy theo phương pháp được chọn.

1.1.3 Radix-2 (cơ số 2)

Radix-2 là cơ số cơ bản nhất trong thiết kế thuật toán FFT/IFFT. Thiết kế Radix-2 FFT/IFFT có phần tiết kiệm tài nguyên nhờ chính sự đơn giản của nó. Nhưng bù lại cho việc đó, thuật toán này sẽ không có độ chính xác cao và tốc độ kém hơn các thiết kế Radix-mix (trộn cơ số). Ý tưởng thiết kế Radix-2 FFT/IFFT dựa trên việc chia chuỗi N – điểm x(n) ra thành hai chuỗi N/2 điểm chẵn, được đặt là x(2n), và N/2 điểm lẻ, x(2n + 1), được tính độc lập với nhau.

1.1.3.1 Radix-2 FFT:

$$X(k) = \sum_{n=0}^{N-1} x(n) \,.\, e^{-j2\pi k\frac{n}{N}}$$

$$\Leftrightarrow \quad X(k) = \sum_{n=0}^{N-1} x(n) \,.\, W_N^{kn} \quad ,với\ \ k = 0, 1, \dots, N-1 \qquad (1.4)$$

- n được chia thành 2 phần: N/2 điểm chẵn và N/2 điểm lẻ.
- điểm chẵn : $x_0(n) = x(2n)$.
- điểm lẻ : $x_1(n) = x(2n+1)$.

⇨ **Điểm chẵn:**

$$X_0(k) = \sum_{n=0}^{\frac{N}{2}-1} x_0(n) \,.\, W_N^{k2n} \quad ,với\ \ k = 0, 1, \dots, \frac{N}{2}-1$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n) \,.\, W_N^{k2n} \quad ,với\ \ k = 0, 1, \dots, \frac{N}{2}-1 \qquad (1.5)$$

⇨ **Điểm lẻ:**

$$X_1(k) = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) \,.\, W_N^{k2n} \quad ,với\ \ k = 0, 1, \dots, \frac{N}{2}-1$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) \,.\, W_N^{k2n} \quad ,với\ \ k = 0, 1, \dots, \frac{N}{2}-1 \qquad (1.6)$$

Ta có: $$W_N^{k2n} = e^{-j2\pi k(\frac{2n}{N})} = e^{-j2\pi k(\frac{n}{\frac{N}{2}})} = W_{\frac{N}{2}}^{kn}$$

⇨ $$X_0(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) \,.\, W_{\frac{N}{2}}^{kn} \quad , với \;\; k = 0, 1, \dots, \frac{N}{2} - 1 \tag{1.7}$$

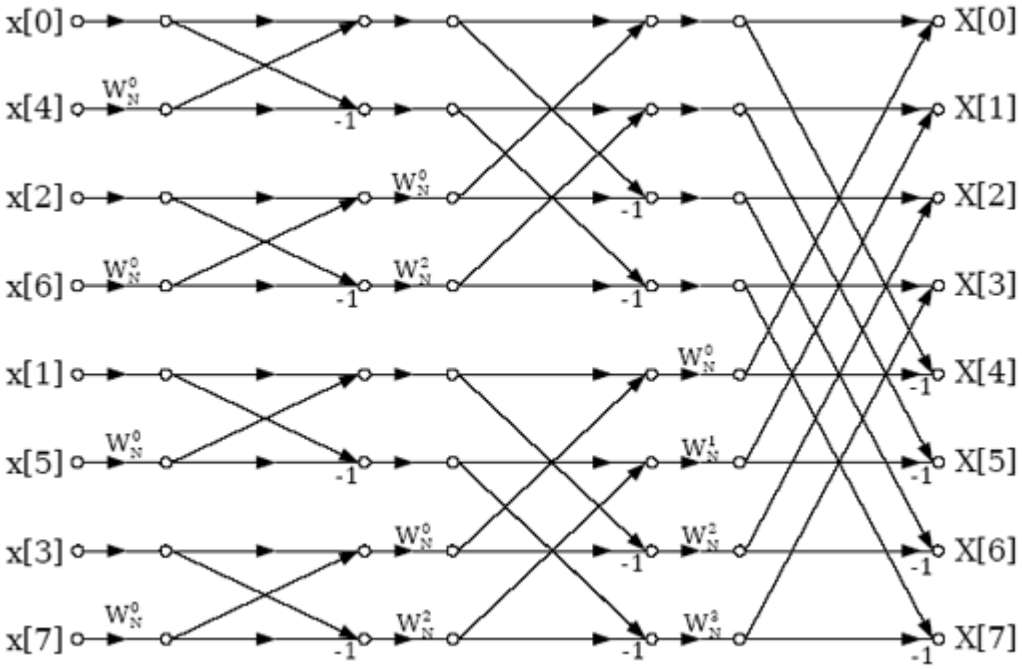
$$X_1(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) \,.\, W_{\frac{N}{2}}^{kn} \quad , với \;\; k = 0, 1, \dots, \frac{N}{2} - 1 \tag{1.8}$$

Từ đó, gộp lại và tính X(k) với k = 0, 1, …, N – 1:

⇨ $$X(k) = X_0(k) + X_1(k) \,.\, W_N^{kn} \quad , với \;\; k = 0, 1, \dots, \frac{N}{2} - 1 \tag{1.9}$$

$$X\left(k + \frac{N}{2}\right) = X_0(k) - X_1(k) \,.\, W_N^{kn}, \; với \;\; k = 0, 1, \dots, \frac{N}{2} - 1 \tag{1.10}$$

Bộ FFT cho 8 điểm:

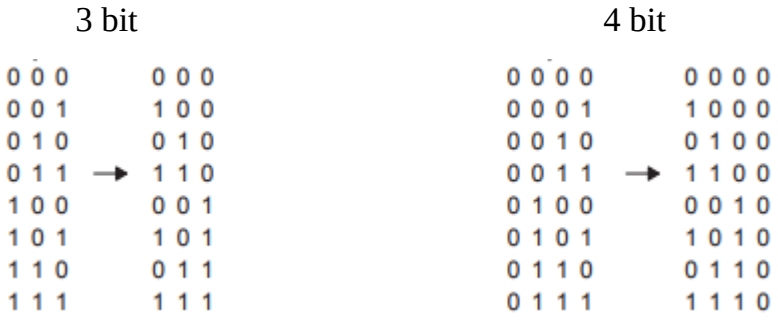


Hình 1-1: Mô hình FFT 8 điểm [2]

Lưu ý: Ở phần này thứ tự các x(n) sẽ có phần bị thay đổi. Ví dụ ở FFT 8 điểm, $8 = 2^3$ nên ta có 3 bit.

Với bit lẻ, bit giữa sẽ không đảo hay thay đổi với các bit còn lại. Ví dụ cho 3 bit, phần bit 0 và 2 sẽ đảo vị trí cho nhau. Phần bit 1 sẽ không đổi.

Với phần chẵn, ta sẽ không có bit giữa nên tất cả các bit đối xứng nhau sẽ bị đảo với nhau. Ví dụ cho 4 bit, bit 0 và 2 sẽ đảo cho nhau. Bit 1 và 3 cũng tương tự. Các bit được đảo theo như hình dưới:



Hình 1-2: Qui tắc biến đổi thứ tự n của x và y

1.1.3.2 Radix-2 IFFT:

IFFT là bộ đảo của FFT. Ta sử dụng công thức của IDFT để thực hiện. Vẫn giống như FFT. Ta chia nhỏ số điểm thành 2 phần chẵn và lẻ để thực hiện.

⇨ **Điểm chẵn:**

$$x_0(n) = \sum_{k=0}^{\frac{N}{2}-1} X_0(k) \,.\, W_N^{-k2n} \quad ,với\ n = 0, 1, \dots, \frac{N}{2} - 1$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X(2k) \,.\, W_N^{-k2n} \quad ,với\ n = 0, 1, \dots, \frac{N}{2} - 1 \qquad (1.11)$$

⇨ **Điểm lẻ:**

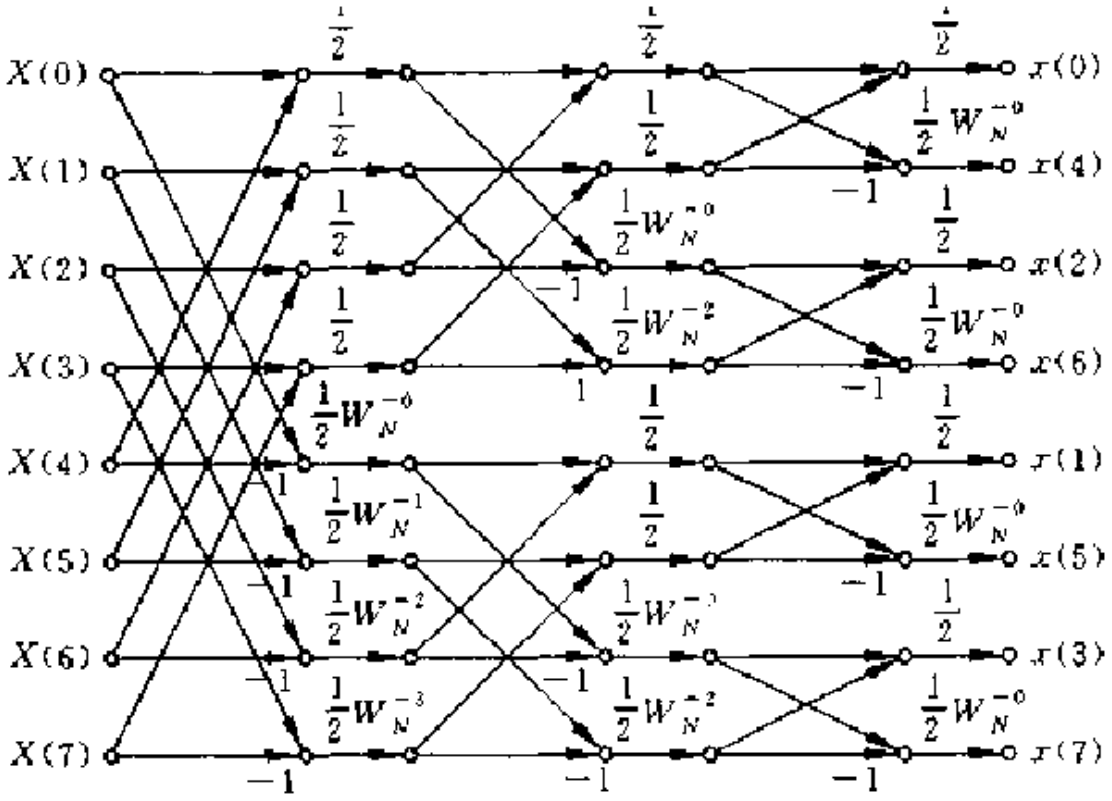
$$x_1(n) = \sum_{k=0}^{\frac{N}{2}-1} X_1(k) \,.\, W_N^{-k2n} \quad ,với\ n = 0, 1, \dots, \frac{N}{2} - 1$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X(2k+1) \,.\, W_N^{-k2n} \,,với\ n = 0, 1, \dots, \frac{N}{2} - 1 \qquad (1.12)$$

Từ đó, gộp lại và tính x(n) với n = 0, 1, …, N – 1:

⇨ $$x(n) = \frac{1}{N}[x_0(n) + x_1(n) \,.\, W_N^{-kn}] \quad ,\ với\ n = 0, 1, \dots, \frac{N}{2} - 1 \qquad (1.13)$$

$$x\left(n + \frac{N}{2}\right) = \frac{1}{N}[x_0(n) - x_1(n) \,.\, W_N^{-kn}] \quad ,\ với\ n = 0, 1, \dots, \frac{N}{2} - 1 \qquad (1.14)$$



Hình 1-3: Mô hình IFFT 8 điểm [2]

Lưu ý: Ở phần này thứ tự các X(k) sẽ có phần bị thay đổi tương tự như ở phần FFT.

Sau đây là các ví dụ về thuật toán Radix-2:

- 4-point (4 điểm): N = 4

  x = [1 2 3 4], tìm kết quả X(k) sau khi qua FFT

Dựa vào công thức (1.7) và (1.8):

$X_0(0) = x(0) + x(2) = 4$

$X_0(1) = x(0) + x(2) \cdot W^1_2 = -2$ (Với $W^1_2 = e^{-j\pi} = -1$)

$X_1(0) = x(1) + x(3) = 6$

$X_1(1) = x(1) + x(3) \cdot W^1_2 = -2$

Với k = 0, 1, 2, 3

Theo công thức (1.9) và (1.10):

⇨ X(0) = 4 + 6 = 4

X(1) = (–2) + (–2) . W^1_4 = –2 + 2j (Với $W^1_4 = e^{-j\pi/2} = -j$)

X(2) = 4 – 6 = –2

X(3) = (–2) – (–2) . W^1_4 = –2 – 2j

X = [10 –2 + 2j –2 –2 – 2j], tìm kết quả x(n) sau khi qua IFFT

Dựa vào công thức (1.11) và (1.12):

$x_0(0)$ = X(0) + X(2) = 10 – 2 = 8

$x_0(1)$ = X(0) + X(2) . W^1_2 = 10 + 2 = 12 (Với $W^1_2 = e^{-j\pi} = -1$)

$x_1(0)$ = X(1) + X(3) = (–2) + 2j + (–2) – 2j = -4

$x_1(1)$ = X(1) + X(3) . W^1_2 = (–2) + 2j – (–2) + 2j = 4j

Với n = 0, 1, 2, 3

Theo công thức (1.13) và (1.14):

⇨ $X(0) = \frac{1}{N}(x_0(0) + x_1(0) . W^0_4) = (8 – 4)/4 = 1$

$X(1) = \frac{1}{N}(x_0(1) + x_1(1) . W^{-1}_4) = (12 + 4j . j)/4 = 2$ (Với $W^{-1}_4 = e^{j\pi/2} = j$)

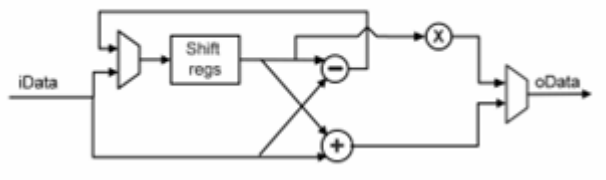
$X(2) = \frac{1}{N}(x_0(0) – x_1(0) . W^0_4) = (8 + 4)/4 = 3$

$X(3) = \frac{1}{N}(x_0(1) – x_1(1) . W^{-1}_4) = (12 – 4j . j)/4 = 4$

Như vậy, ta ra được kết quả x = [1 2 3 4]

Do đó, ta có thể nâng cao lên đến 8,16,…, 1024 điểm với phương pháp trên bằng cách chia nhỏ từng tầng.

1.2 Kiến trúc SDF (Single–path Delay Feedback)



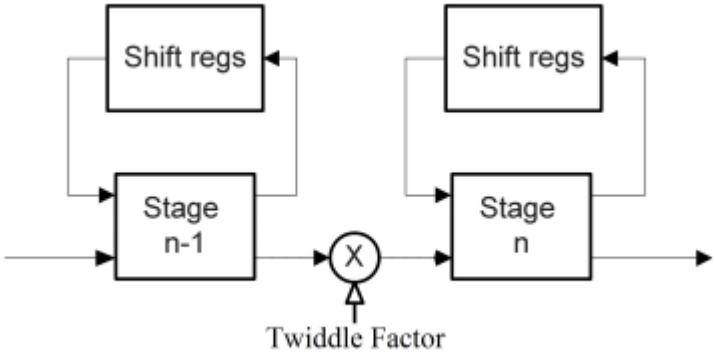
Hình 1-4: Kiến trúc SDF Radix-2 [2]

Sau đây là các bước thực hiện của 1 tầng SDF:

Bước 1: N/2 dữ liệu đầu ở ngõ vào đi vào bộ mux và được ghi vào thanh ghi dịch (shift regs). Không có dữ liệu ở ngõ ra.

Bước 2: N/2 dữ liệu đầu ở thanh ghi dịch sẽ thực hiện bộ cộng (+), trừ (–) với dữ liệu sau ở ngõ vào. Sau đó, dữ liệu ở bộ cộng (+) sẽ xuất ở ngõ ra. Dữ liệu ở bộ trừ (–) được hồi tiếp trở lại và vào thanh ghi dịch.

Bước 3: N/2 dữ liệu ở thanh ghi dịch sau bước 2 sẽ được xuất ở ngõ ra và kết thúc quá trình.



Hình 1-5: Sơ đồ khối 2 tầng kề nhau [2]

Mỗi tầng SDF sẽ có một bộ đệm dữ liệu hồi tiếp. Dung lượng bộ đệm của tầng sau sẽ bằng nửa dung lượng bộ đệm của tầng trước. Bộ nhân số phức W sẽ nằm trên đường đi dữ liệu giữa hai tầng kế tiếp. Việc thay đổi cơ số Radix chỉ làm thay đổi cấu trúc và cách điều khiển bộ nhân W, chứ không làm thay đổi cấu trúc của một tầng trong SDF.

CHƯƠNG 2. TỔNG QUAN VỀ FPGA

2.1 Giới thiệu về FPGA

FPGA là một vi mạch tích hợp dạng lớn dùng cấu trúc mảng logic mà người sử dụng có thể lập trình được.

Các bộ phận cấu thành một vi mạch FPGA:

- Các khối logic cơ bản có thể lập trình (Logic Block).
- Hệ thống các mạch liên kết có thể lập trình.
- Các khối xuất/nhập (I/O Pads).
- Các phần tử được thiết kế sẵn như RAM, ROM, nhân vi xử lý, DSP Slice,….

2.1.1 Lịch sử

FPGA được thiết kế đầu tiên bởi nhà sáng lập công ty Xilinx, Ross Freeman, vào năm 1984, kiến trúc mới của FPGA đã cho phép tích hợp một lượng rất lớn các phần tử bán dẫn vào 1 vi mạch so với CPLD. Số lượng cổng logic mà FPGA có khả năng chứa là từ 100.000 đến hàng tỷ cổng. Trong khi đó, con số mà CPLD có là 10.000 - 100.000 cổng logic. Ngoài ra PAL, PLA còn thấp hơn nữa với chỉ từ thấp hơn 10.000 cổng.

CPLD được cấu trúc từ một lượng cố định các khối SPLD (thuật ngữ chỉ chung cho PLA, PAL). SPLD là một mảng logic AND/OR lập trình được có kích thước xác định và chứa một lượng hạn chế các phần tử nhớ đồng bộ (clocked register).

Kiến trúc FPGA là kiến trúc mảng các khối logic, nó nhỏ hơn nhiều so với một khối SPLD, ưu điểm này đã giúp FPGA có thể chứa các phần tử logic nhiều hơn và khả năng lập trình của các phần tử logic và hệ thống mạch kết nối được phát huy tối đa, kiến trúc của FPGA phức tạp hơn nhiều so với CPLD để có thể đạt được các mục tiêu trên.

2.1.2 Sự khác biệt của FPGA với ASIC

ASIC là một vi mạch IC được thiết kế cho một ứng dụng riêng biệt và không thể thay đổi ứng dụng đó trong cùng một vi mạch.

Cũng giống như ASIC, FPGA cũng được xem như một loại vi mạch bán dẫn chuyên dụng. Nhưng FPGA không thể giống các trường hợp ASIC đặc chế hoàn toàn hay ASIC thiết kế trên thư viện logic. Khuyết điểm của FPGA so với ASIC là không thể đạt mức độ tối ưu, và hạn chế trong việc thực hiện các chức năng với những tác vụ đặc biệt phức tạp.

Tuy nhiên, FPGA lại có ưu thế hơn là có thể cấu trúc và tổng hợp lại khi đang sử dụng, chi phí giảm do công đoạn thiết kế đơn giản hơn, nhờ vậy mà thời gian đưa sản phẩm vào cũng được rút ngắn.

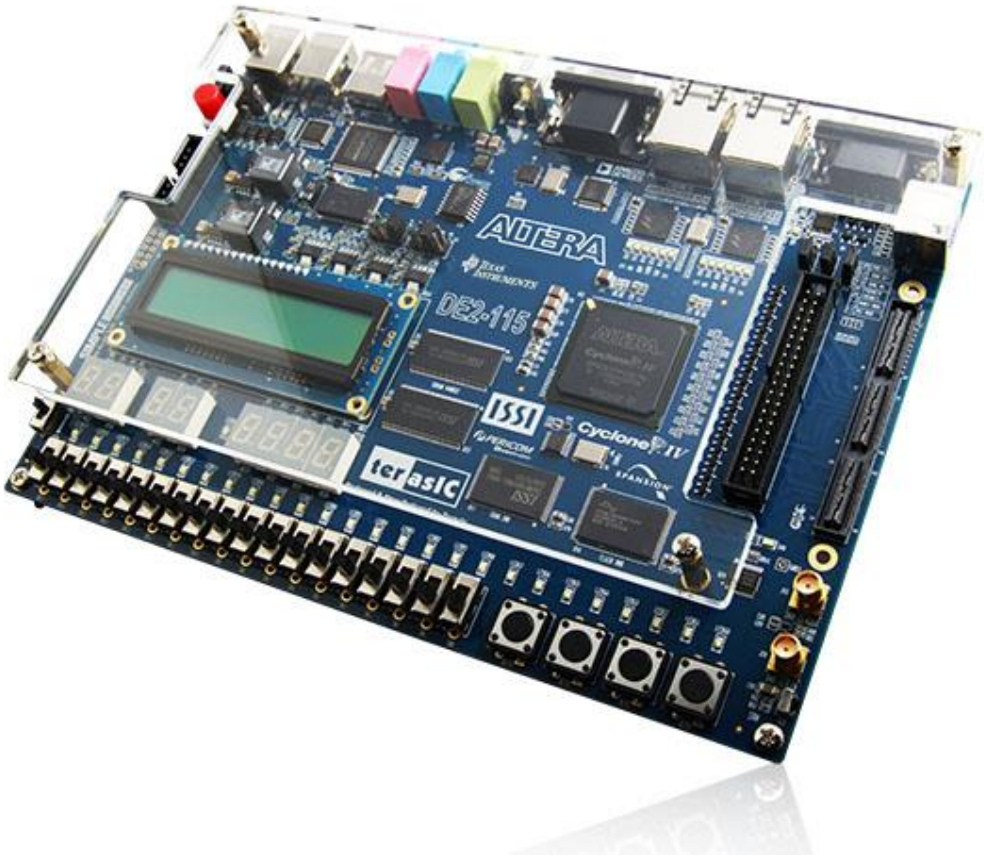
2.1.3 Ứng dụng

FPGA có nhiều ứng dụng như: các xử lý số DSP, tiền thiết kế cho các mẫu ASIC, phân tích và nhận dạng âm thanh, hình ảnh, mô hình phần cứng máy tính, mật mã học,….

Các bài toán phức tạp cũng được FPGA giải quyết nhanh mà trước kia chỉ có thể thực hiện trên phần mềm máy tính bởi tính linh động cao trong quá trình thiết kế của nó.

Ngoài ra, nhờ số lượng mật độ các cổng logic trong FPGA tương đối lớn có thể thực hiện các bài toán với độ phức tạp cao cần khối lượng bước tính toán lớn và dùng trong các thực hiện trong thời gian thực.

2.2 Tổng quan về kit DE2-115



Hình 2-1: Kit DE2-115 [3]

Kit học tập và nghiên cứu phát triển DE2-115 của hãng Altera được thiết kế bởi các nhà phát triển, với mục đích phục vụ cho việc nghiên cứu và học tập. Là một công cụ thông minh dành cho việc học tập về logic số, FPGA. Đặc trưng cho dòng Cyclone IV 4CE115 FPGA, kit DE2-115 được sử dụng phổ biến trong phòng thí nghiệm tại các trường đại học, cao đẳng, phù hợp cho các bài thực hành, nghiên cứu khoa học về logic số, từ những ứng dụng cơ bản đến thiết kế nâng cao.

Dòng DE2 luôn đi đầu trong việc phát triển giáo dục bởi sự khác biệt của bản thân nó với một giao diện phong phú để đáp ứng nhiều ứng dụng thiết yếu. Mở rộng việc dẫn đầu

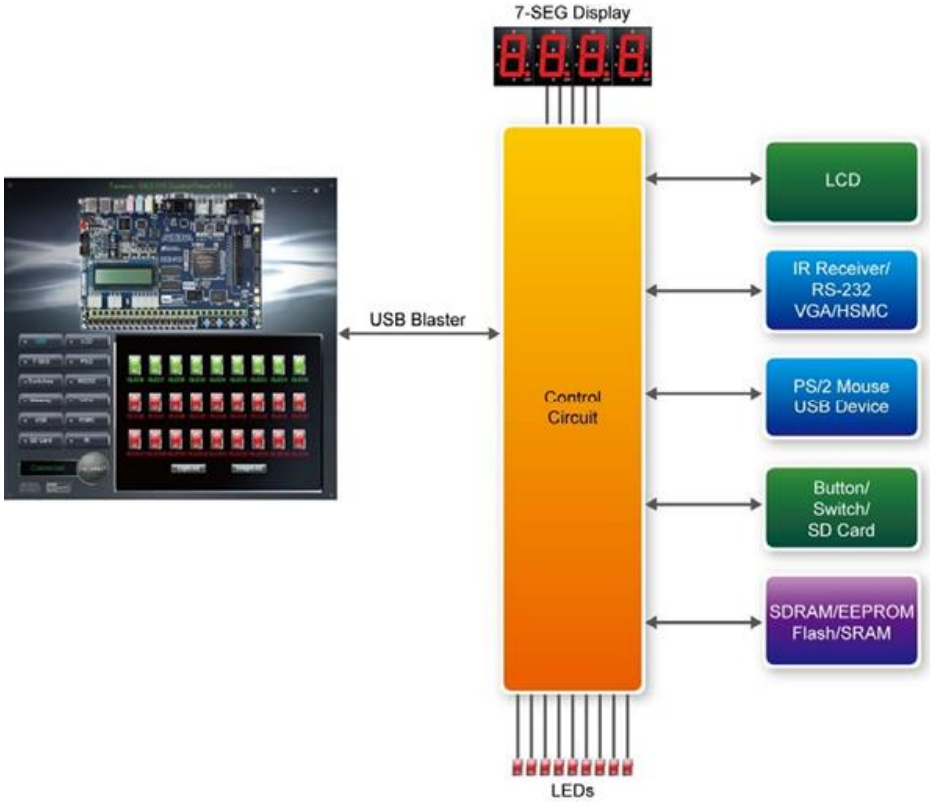
và thành công, Terasic thông báo rằng DE2-115 là sản phẩm cuối cùng với sự phục vụ của thiết bị Cyclone IV E.

2.2.1 Cấu tạo DE2-115

Kit DE2-115 có rất nhiều tính năng nổi bậc cho phép người dùng có thể đa dạng về thiết kế, cấu trúc các mạch, từ các mạnh đơn giản đến nhiều dự án đa phương tiện. Sau đây là phần cứng của kit DE2-115:

- Thiết bị Altera Cyclone IV 4CE115
- Thiết bị cấu hình Serial Altera – EPCS64
- USB Blaster (trong mạch) cho việc lập trình; cả JTAG và Active Serial (AS) các chế độ lập trình cũng được hỗ trợ.
- 2MB SRAM.
- Hai bộ 64MB SDRAM.
- 8MB Flash memory.
- SD Card socket.
- 4 nút bấm.
- 18 công tắc.
- 18 đèn LED đỏ.
- 9 đèn LED xanh lá.
- Bộ dao động cho các nguồn xung clock 50MHz trong và ngoài.
- 24-bit CD-quality audio CODEC với ngõ vào, ngõ ra, và các ổ cắm microphone.
- VGA DAC (8-bit high-speed triple DACs) với cổng kết nối ngoài của VGA.
- Bộ giải mã TV (NTSC/PAL/SECAM) và cổng kết nối ngõ vào TV.
- Gigabit Ethernet PHY với các cổng kết nối RJ45.
- USB bộ điều khiển Host/Slave với cổng nối USB loại A và loại B.
- Bộ thu phát RS-232 and 9 cổng nối.
- PS/2 cổng nối chuột/bàn phím.

- Bộ thu IR.
- Cổng nối ngõ vào, ra của xung clock ngoài của SMA.
- Một cổng nối High Speed Mezzanine Card (HSMC).
- 16x2 LCD module.



Hình 2-2: Sơ đồ chức năng DE2-115 [3]

2.2.2 Sơ đồ chân

Sau đây là sơ đồ các chân kit DE2-115:

Bảng 2-1: Bảng vị trí các chân công tắc (Switches) [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
SW[0]	Chân AB28	Công tắc đóng/mở	Phụ thuộc JP7
SW[1]	Chân AC28	Công tắc đóng/mở	Phụ thuộc JP7
SW[2]	Chân AC27	Công tắc đóng/mở	Phụ thuộc JP7
SW[3]	Chân AD27	Công tắc đóng/mở	Phụ thuộc JP7

SW[4]	**Chân AB27**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[5]	**Chân AC26**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[6]	**Chân AD26**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[7]	**Chân AB26**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[8]	**Chân AC25**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[9]	**Chân AB25**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[10]	**Chân AC24**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[11]	**Chân AB24**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[12]	**Chân AB23**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[13]	**Chân AA24**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[14]	**Chân AA23**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[15]	**Chân AA22**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[16]	**Chân Y24**	**Công tắc đóng/mở**	**Phụ thuộc JP7**
SW[17]	**Chân Y23**	**Công tắc đóng/mở**	**Phụ thuộc JP7**

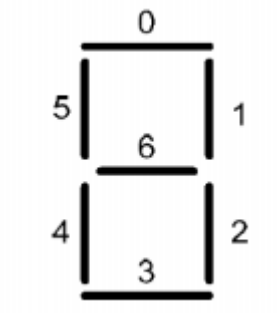
Bảng 2-2: Bảng vị trí các chân nút bấm (Button) [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
KEY[0]	**Chân M23**	**Công tắc nút**	**Phụ thuộc JP7**
KEY[1]	**Chân M21**	**Công tắc nút**	**Phụ thuộc JP7**
KEY[2]	**Chân N21**	**Công tắc nút**	**Phụ thuộc JP7**
KEY[3]	**Chân R24**	**Công tắc nút**	**Phụ thuộc JP7**

Bảng 2-3: Bảng vị trí các chân LED [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
LEDR[0]	**Chân G19**	**LED đỏ [0]**	**2.5V**
LEDR[1]	**Chân F19**	**LED đỏ [1]**	**2.5V**
LEDR[2]	**Chân E19**	**LED đỏ [2]**	**2.5V**
LEDR[3]	**Chân F21**	**LED đỏ [3]**	**2.5V**
LEDR[4]	**Chân F18**	**LED đỏ [4]**	**2.5V**
LEDR[5]	**Chân E18**	**LED đỏ [5]**	**2.5V**
LEDR[6]	**Chân J19**	**LED đỏ [6]**	**2.5V**
LEDR[7]	**Chân H19**	**LED đỏ [7]**	**2.5V**
LEDR[8]	**Chân J17**	**LED đỏ [8]**	**2.5V**
LEDR[9]	**Chân G17**	**LED đỏ [9]**	**2.5V**
LEDR[10]	**Chân J15**	**LED đỏ [10]**	**2.5V**
LEDR[11]	**Chân H16**	**LED đỏ [11]**	**2.5V**
LEDR[12]	**Chân J16**	**LED đỏ [12]**	**2.5V**

LEDR[13]	**Chân H17**	**LED đỏ [13]**	**2.5V**
LEDR[14]	**Chân F15**	**LED đỏ [14]**	**2.5V**
LEDR[15]	**Chân G15**	**LED đỏ [15]**	**2.5V**
LEDR[16]	**Chân G16**	**LED đỏ [16]**	**2.5V**
LEDR[17]	**Chân H15**	**LED đỏ [17]**	**2.5V**
LEDG[0]	**Chân E21**	**LED xanh [0]**	**2.5V**
LEDG[1]	**Chân E22**	**LED xanh [1]**	**2.5V**
LEDG[2]	**Chân E25**	**LED xanh [2]**	**2.5V**
LEDG[3]	**Chân E24**	**LED xanh [3]**	**2.5V**
LEDG[4]	**Chân H21**	**LED xanh [4]**	**2.5V**
LEDG[5]	**Chân G20**	**LED xanh [5]**	**2.5V**
LEDG[6]	**Chân G22**	**LED xanh [6]**	**2.5V**
LEDG[7]	**Chân G21**	**LED xanh [7]**	**2.5V**
LEDG[8]	**Chân F17**	**LED xanh [8]**	**2.5V**



Hình 2-3: Thứ tự các chân LED 7 đoạn [2]

LED 7 đoạn trong kit DE2-115 là LED Anode nên các đoạn trên LED 7 đoạn sẽ sáng lên khi được tích hợp bit 0 (mức thấp).

Bảng 2-4: Vị trí các chân LED 7 đoạn [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
HEX0[0]	**Chân G18**	**LED 7 đoạn 0[0]**	**2.5V**
HEX0[1]	**Chân F22**	**LED 7 đoạn 0[1]**	**2.5V**
HEX0[2]	**Chân E17**	**LED 7 đoạn 0[2]**	**2.5V**
HEX0[3]	**Chân L26**	**LED 7 đoạn 0[3]**	**Phụ thuộc JP7**
HEX0[4]	**Chân L25**	**LED 7 đoạn 0[4]**	**Phụ thuộc JP7**
HEX0[5]	**Chân J22**	**LED 7 đoạn 0[5]**	**Phụ thuộc JP7**
HEX0[6]	**Chân H22**	**LED 7 đoạn 0[6]**	**Phụ thuộc JP7**
HEX1[0]	**Chân M24**	**LED 7 đoạn 1[0]**	**Phụ thuộc JP7**

HEX1[1]	**Chân Y22**	**LED 7 đoạn 1[1]**	**Phụ thuộc JP7**
HEX1[2]	**Chân W21**	**LED 7 đoạn 1[2]**	**Phụ thuộc JP7**
HEX1[3]	**Chân W22**	**LED 7 đoạn 1[3]**	**Phụ thuộc JP7**
HEX1[4]	**Chân W25**	**LED 7 đoạn 1[4]**	**Phụ thuộc JP7**
HEX1[5]	**Chân U23**	**LED 7 đoạn 1[5]**	**Phụ thuộc JP7**
HEX1[6]	**Chân U24**	**LED 7 đoạn 1[6]**	**Phụ thuộc JP7**
HEX2[0]	**Chân AA25**	**LED 7 đoạn 2[0]**	**Phụ thuộc JP7**
HEX2[1]	**Chân AA26**	**LED 7 đoạn 2[1]**	**Phụ thuộc JP7**
HEX2[2]	**Chân Y25**	**LED 7 đoạn 2[2]**	**Phụ thuộc JP7**
HEX2[3]	**Chân W26**	**LED 7 đoạn 2[3]**	**Phụ thuộc JP7**
HEX2[4]	**Chân Y26**	**LED 7 đoạn 2[4]**	**Phụ thuộc JP7**
HEX2[5]	**Chân W27**	**LED 7 đoạn 2[5]**	**Phụ thuộc JP7**
HEX2[6]	**Chân W28**	**LED 7 đoạn 2[6]**	**Phụ thuộc JP7**
HEX3[0]	**Chân V21**	**LED 7 đoạn 3[0]**	**Phụ thuộc JP7**
HEX3[1]	**Chân U21**	**LED 7 đoạn 3[1]**	**Phụ thuộc JP7**
HEX3[2]	**Chân AB20**	**LED 7 đoạn 3[2]**	**Phụ thuộc JP6**
HEX3[3]	**Chân AA21**	**LED 7 đoạn 3[3]**	**Phụ thuộc JP6**
HEX3[4]	**Chân AD24**	**LED 7 đoạn 3[4]**	**Phụ thuộc JP6**
HEX3[5]	**Chân AF23**	**LED 7 đoạn 3[5]**	**Phụ thuộc JP6**
HEX3[6]	**Chân Y19**	**LED 7 đoạn 3[6]**	**Phụ thuộc JP6**
HEX4[0]	**Chân AB19**	**LED 7 đoạn 4[0]**	**Phụ thuộc JP6**
HEX4[1]	**Chân AA19**	**LED 7 đoạn 4[1]**	**Phụ thuộc JP6**
HEX4[2]	**Chân AG21**	**LED 7 đoạn 4[2]**	**Phụ thuộc JP6**
HEX4[3]	**Chân AH21**	**LED 7 đoạn 4[3]**	**Phụ thuộc JP6**
HEX4[4]	**Chân AE19**	**LED 7 đoạn 4[4]**	**Phụ thuộc JP6**
HEX4[5]	**Chân AF19**	**LED 7 đoạn 4[5]**	**Phụ thuộc JP6**
HEX4[6]	**Chân AE18**	**LED 7 đoạn 4[6]**	**Phụ thuộc JP6**
HEX5[0]	**Chân AD18**	**LED 7 đoạn 5[0]**	**Phụ thuộc JP6**
HEX5[1]	**Chân AC18**	**LED 7 đoạn 5[1]**	**Phụ thuộc JP6**
HEX5[2]	**Chân AB18**	**LED 7 đoạn 5[2]**	**Phụ thuộc JP6**
HEX5[3]	**Chân AH19**	**LED 7 đoạn 5[3]**	**Phụ thuộc JP6**
HEX5[4]	**Chân AG19**	**LED 7 đoạn 5[4]**	**Phụ thuộc JP6**
HEX5[5]	**Chân AF18**	**LED 7 đoạn 5[5]**	**Phụ thuộc JP6**
HEX5[6]	**Chân AH18**	**LED 7 đoạn 5[6]**	**Phụ thuộc JP6**
HEX6[0]	**Chân AA17**	**LED 7 đoạn 6[0]**	**Phụ thuộc JP6**
HEX6[1]	**Chân AB16**	**LED 7 đoạn 6[1]**	**Phụ thuộc JP6**
HEX6[2]	**Chân AA16**	**LED 7 đoạn 6[2]**	**Phụ thuộc JP6**
HEX6[3]	**Chân AB17**	**LED 7 đoạn 6[3]**	**Phụ thuộc JP6**

HEX6[4]	**Chân AB15**	**LED 7 đoạn 6[4]**	**Phụ thuộc JP6**
HEX6[5]	**Chân AA15**	**LED 7 đoạn 6[5]**	**Phụ thuộc JP6**
HEX6[6]	**Chân AC17**	**LED 7 đoạn 6[6]**	**Phụ thuộc JP6**
HEX7[0]	**Chân AD17**	**LED 7 đoạn 7[0]**	**Phụ thuộc JP6**
HEX7[1]	**Chân AE17**	**LED 7 đoạn 7[1]**	**Phụ thuộc JP6**
HEX7[2]	**Chân AG17**	**LED 7 đoạn 7[2]**	**Phụ thuộc JP6**
HEX7[3]	**Chân AH17**	**LED 7 đoạn 7[3]**	**Phụ thuộc JP6**
HEX7[4]	**Chân AF17**	**LED 7 đoạn 7[4]**	**Phụ thuộc JP6**
HEX7[5]	**Chân AG18**	**LED 7 đoạn 7[5]**	**Phụ thuộc JP6**
HEX7[6]	**Chân AA14**	**LED 7 đoạn 7[6]**	**3.3V**

Bảng 2-5: Vị trí các chân xung clock [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
CLOCK_50	**Chân Y2**	**Ngõ vào clock 50MHz**	**3.3V**
CLOCK2_50	**Chân AG14**	**Ngõ vào clock 50MHz**	**3.3V**
CLOCK3_50	**Chân AG15**	**Ngõ vào clock 50MHz**	**Phụ thuộc JP6**
SMA_CLKOUT	**Chân AE23**	**Ngõ ra clock ngoài**	**Phụ thuộc JP6**
SMA_CLKIN	**Chân AH14**	**Ngõ vào clock ngoài**	**3.3V**

Bảng 2-6: Vị trí các chân giao tiếp LCD [7]

Tên Tín Hiệu	Vị Trí Chân Trong FPGA	Mô Tả	Tiêu Chuẩn Ngõ Vào/Ra
LCD_DATA[7]	**Chân M5**	**Dữ liệu LCD (7)**	**3.3V**
LCD_DATA[6]	**Chân M3**	**Dữ liệu LCD (6)**	**3.3V**
LCD_DATA[5]	**Chân K2**	**Dữ liệu LCD (5)**	**3.3V**
LCD_DATA[4]	**Chân K1**	**Dữ liệu LCD (4)**	**3.3V**
LCD_DATA[3]	**Chân K7**	**Dữ liệu LCD (3)**	**3.3V**
LCD_DATA[2]	**Chân L2**	**Dữ liệu LCD (2)**	**3.3V**
LCD_DATA[1]	**Chân L1**	**Dữ liệu LCD (1)**	**3.3V**
LCD_DATA[0]	**Chân L3**	**Dữ liệu LCD (0)**	**3.3V**
LCD_EN	**Chân L4**	**LCD Enable**	**3.3V**
LCD_RW	**Chân M1**	**Chế độ Đọc/Viết, '0' = viết, '1' = đọc**	**3.3V**
LCD_RS	**Chân M2**	**Lệnh/ Dữ liệu, '0' = Lệnh, '1' = dữ liệu**	**3.3V**
LCD_ON	**Chân L5**	**LCD mở/ngắt nguồn**	**3.3V**
LCD_BLON	**Chân L6**	**LCD mở/ ngắt back light**	**3.3V**

2.3 Phần mềm thiết kế Quartus II

2.3.1 Giới thiệu

Altera Quartus là phần mềm thiết kế và lập trình thiết bị logic được sản xuất bởi Altera, trước khi Altera được mua lại bởi Intel và được đổi tên thành Intel Quartus Prime. Quartus II cho phép các nhà phân tích và tổng hợp các thiết kế HDL, cho phép các nhà phát triển biên soạn các tài liệu của họ, phân tích thời gian thực hiện, kiểm tra sơ đồ RTL, mô phỏng phản ứng của 1 thiết kế, và cấu hình cho thiết bị với người lập trình. Quartus II bao gồm việc triển khai VHDL và Verilog để mô tả phần cứng. Chỉnh sửa các mạch logic, và mô phỏng dạng sóng.

Các thiết kế logic được Quartus II cung cấp:

- Thiết kế Logiclock.
- Công cụ mạnh cho việc tổng hợp logic.
- Có khả năng nhận biết và mô tả linh kiện.
- Tự đinh vị lỗi và cảnh báo.
- Sử dụng một số ngôn ngữ lập trình để thiết kế sơ đồ khối, bản vẽ như: AHDL, VHDL, Verilog HDL.
- Mô phỏng chức năng và thời gian của thiết kế.

2.3.2 Ngôn ngữ lập trình VHDL

Giống với ngôn ngữ lập trình Verilog HDL, VHDL được xem như 1 module: Bên trong đó bao gồm 3 phần: phần khai báo thư viện (Library declarations), phần mô tả các thực thể trong chương trình (Entity) và cuối cùng là phần mô tả kiến trúc chương trình (Architecture).

2.3.2.1 Thư viện (Library)

Đúng như tên gọi, thư viện là nơi để người lập trình chứa các thông tin của một project. Trong thư viện có thể lưu trữ kiểu dữ liệu, hàm, tín hiệu, khai báo thành phần mà các có thể sử dụng bởi các mô hình VHDL khác nhau.

Cú pháp:

Library [tên_thư_viện];

Use [tên_thư_viện].[tên_gói].[thành_phần_gói];

Một số thư viện đã được nhà sản xuất tích hợp sẵn:

- IEEE:
  - Gói std_logic_1164: bao gồm các dữ liệu chuẩn.
  - Gói std_logic_arith: định nghĩa các chức năng số học, các tín hiệu signed, unsigned, integer, std_logic.
- STD.TEXTIO: chứa các hàm READ/WRITE để đọc dữ liệu từ FILE.
- IEEE.math_real, IEEE.math_complex: cung cấp các công thức tính số thực hay số phức như hàm COSINE, SINE và nhiều hàm tính toán.

2.3.2.2 Entity

Entity là phần khai báo thực thể. Các biến ngõ vào ra cấp cao. Trong một thiết kế sẽ có một hoặc nhiều entity liên kết với nhau tùy thuộc vào độ phức tạp của chương trình.

Cú pháp:

Entity [tên_entity] **is**;

Port (tên thực thể : [chế độ] [kiểu tín hiêu];

tên thực thể : [chế độ] [kiểu tín hiêu]);

End [tên_entity];

2.3.2.3 Architecture

Architecture là phần mô tả kiến trúc của chương trình VHDL. Là cách mô tả để chương trình thực hiện theo cách thức hoạt động, chức năng mà người dùng đã lập trình.

Một kiến trúc bao gồm khai báo tín hiệu, hằng số, loại tín hiệu. Sau đó là sử dụng các cổng logic hay hàm đã được lập trình và khai báo ở thư viện để thực hiện.

Cú pháp:

Architecture [tên_architecture] **of** [tên_entity] **is**

--khai báo hằng số, tín hiệu , kiểu tín hiệu, thành phần, hàm--

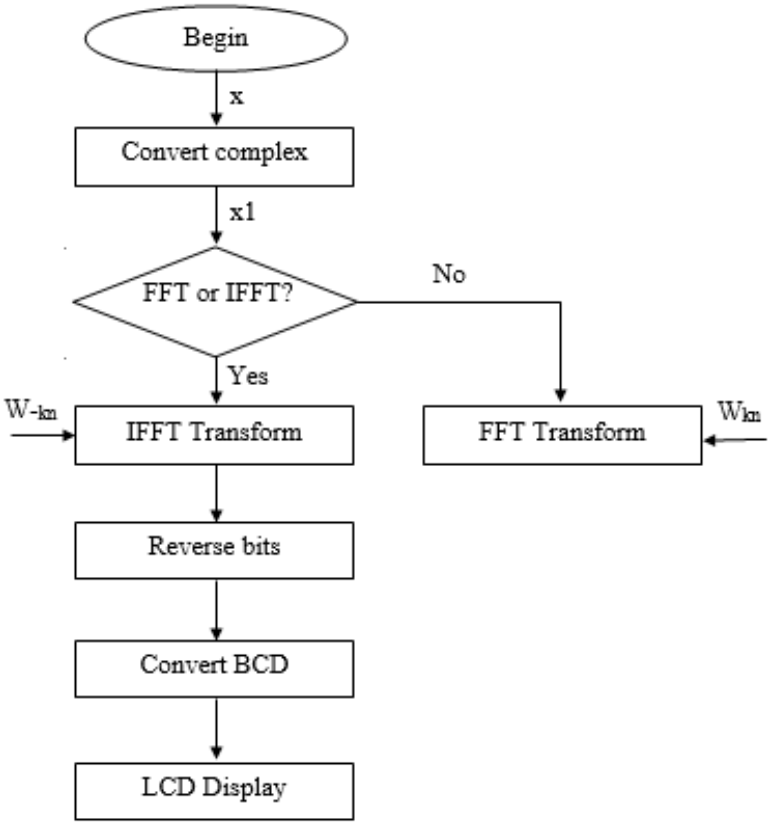
Begin

--- viết code---

End [tên_architecture];

CHƯƠNG 3. QUÁ TRÌNH THỰC HIỆN

3.1 Thiết lập sơ đồ hoạt động

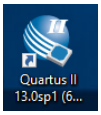


Hình 3-1: Sơ đồ hoạt động của bộ FFT/IFFT

Đầu tiên, ngõ vào x là một chuỗi số nguyên (nhị phân). Sau đó, dữ liệu được biến đổi thành số phức (nhị phân) để có thể thực hiện việc tính toán. Tiếp theo, dữ liệu được đưa vào bộ FFT/IFFT. Người dùng sẽ có 2 chế độ: FFT nếu bit điều khiển là '0' và IFFT nếu bit điều khiển là '1'. Sau khi đã thực hiện tính toán, dữ liệu được đảo bit nhị phân của số thứ tự n trong ngõ y. Tiếp đến, bộ sẽ chuyển đổi sang dạng BCD để có thể hiển thị lên LCD.

3.2 Các bước thực hiện

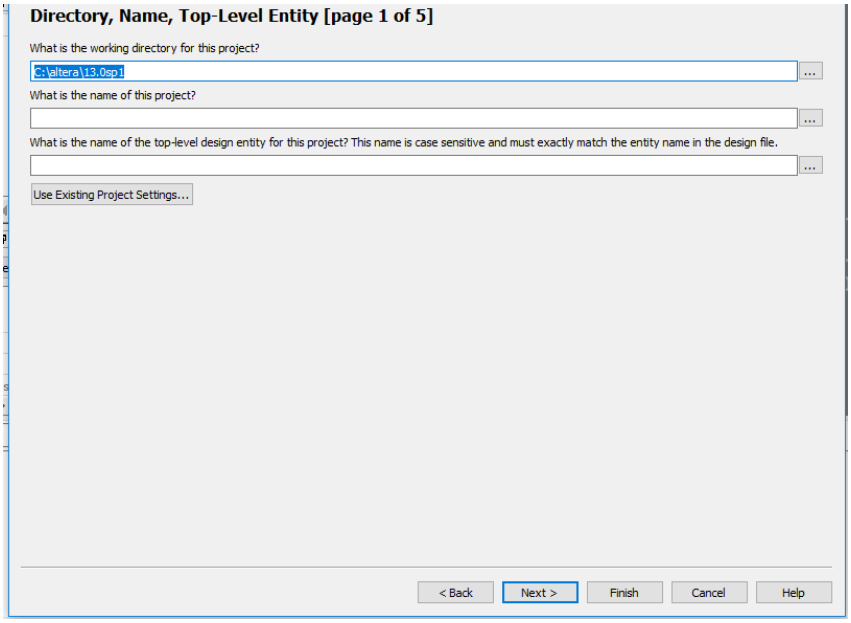
3.2.1 Tạo Project

Bước 1: Chọn biểu tượng 

để khởi động chương trình.

Bước 2: Khi chương trình đã được khởi động, chọn File → New Project Wizard.

Bước 3: Màn hình hiện lên bảng tạo project.

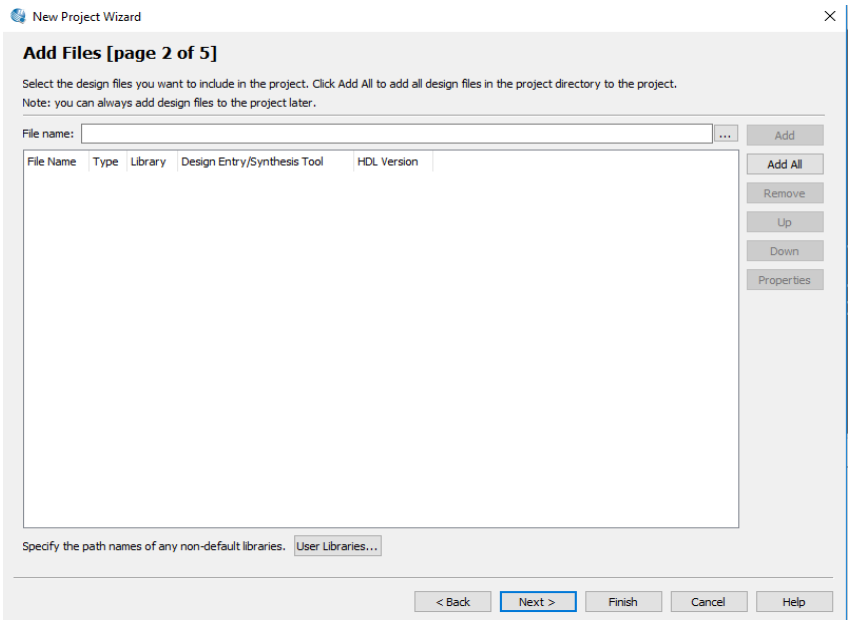


Hình 3-2: Giao diện đặt tên project

- ✓ Dòng đầu tiên là chọn vị trí đặt file project.
- ✓ Dòng thứ 2 là đặt tên cho project.
- ✓ Dòng thứ 3 là đặt tên thiết kế chính của project.

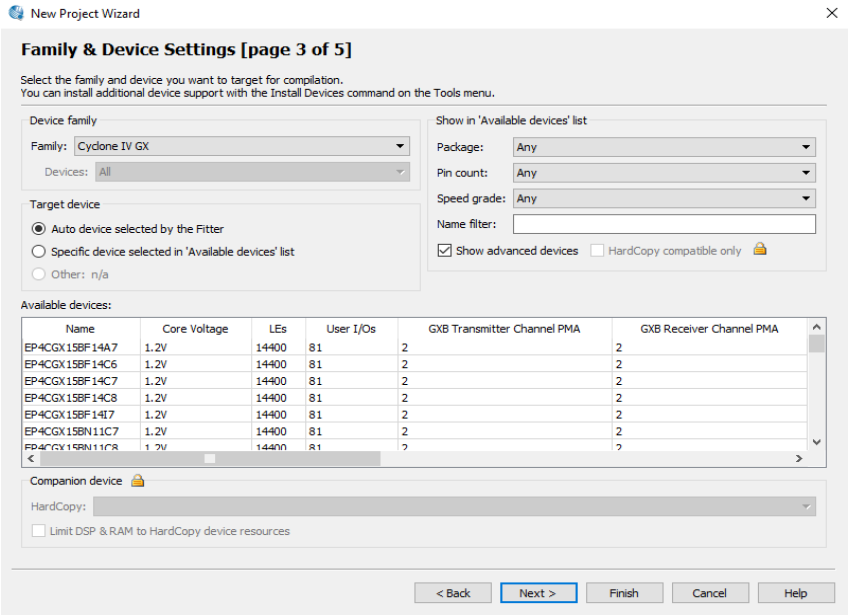
Bấm next sau khi đã xong phần đặt tên.

Bước 4: Trang tiếp theo cho phép ta thêm các file code đã có sẵn.



Hình 3-3: Giao diện thêm thư viên,file có sẵn cho project

Bước 5: Trang tiếp theo cho phép ta chọn dòng thiết bị thực hiện.



Hình 3-4: Giao diện chọn thiết bị cho project

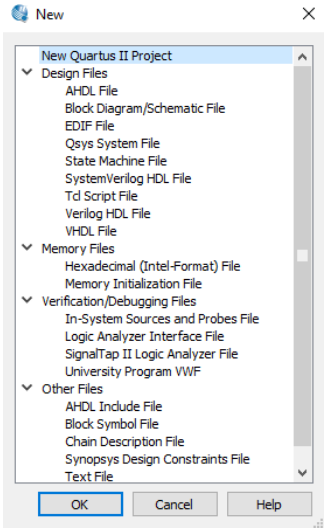
Bước 6: Trang thiết lập cài đặt EDA tool.

Bước 7: chọn Finish để hoàn thành việc tạo project.

3.2.2 Viết code và compile code

Bước 1: Sau khi đã tạo project, ta tạo file code.

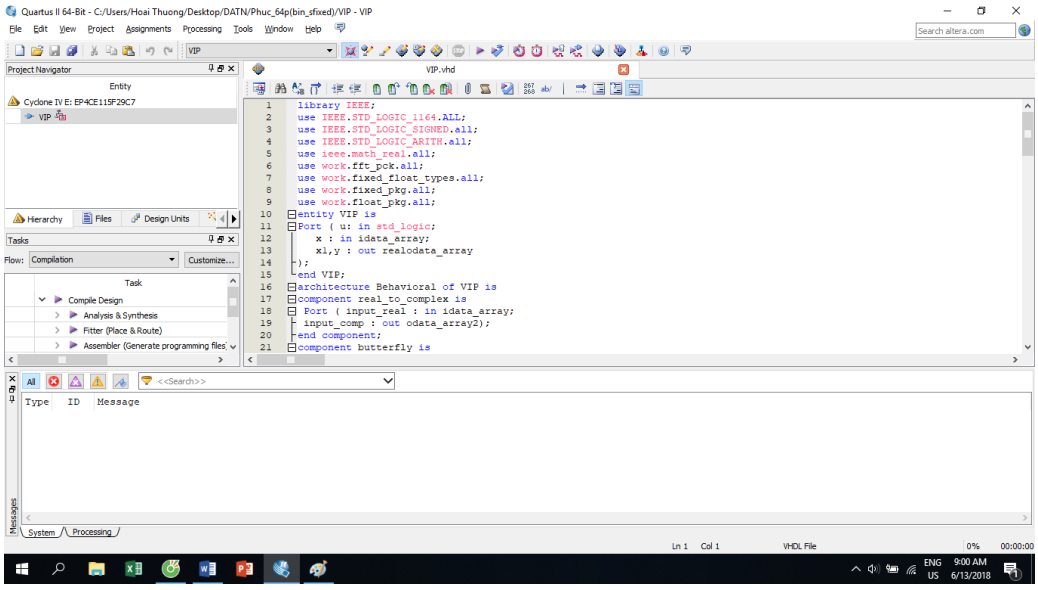
Bao gồm file AHDL, VHDL, Verilog HDL và một số các thiết kế khác.



Hình 3-5: Tạo file mới cho project

Bước 2: Ta chọn VHDL File để viết và lập trình.

Bước 3: Thực hiện việc viết code để thiết kế toàn mạch hoặc thành phần.



Hình 3-6: Trang để viết code VHDL

Bước 3: Chọn Processing → Start Compilation hoặc chọn  trên giao diện để thực hiện.

3.2.3 Mô phỏng ModelSim

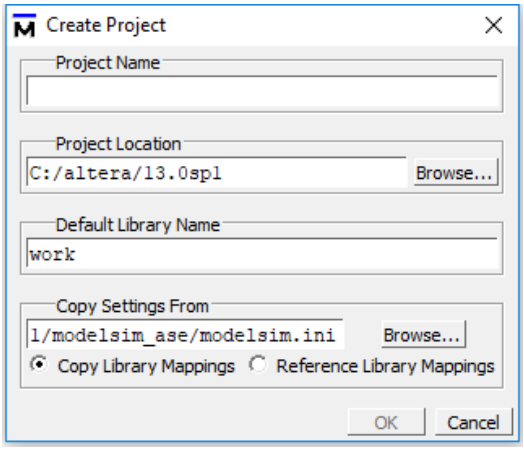
ModelSim là phần mềm cũng thuộc Altera cùng với Quartus II. ModelSim là phần mềm mô phỏng hoàn thiện hơn so với mô phỏng dạng sóng trong Quartus II. ModelSim có thể mô phỏng tín hiệu dạng chuỗi số phức mà Quartus II không thực hiện được.

Sau đây là các bước thực hiện mô phỏng với ModelSim:

Bước 1: chọn biểu tượng  để khởi động chương trình.

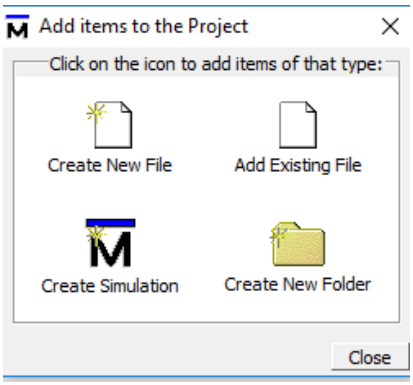
Bước 2: Chọn File → New → Project.

Bước 3: Cửa sổ hiện ra. Đặt tên project và chọn vị trí của project.



Hình 3-7: Giao diện đặt tên project (ModelSim)

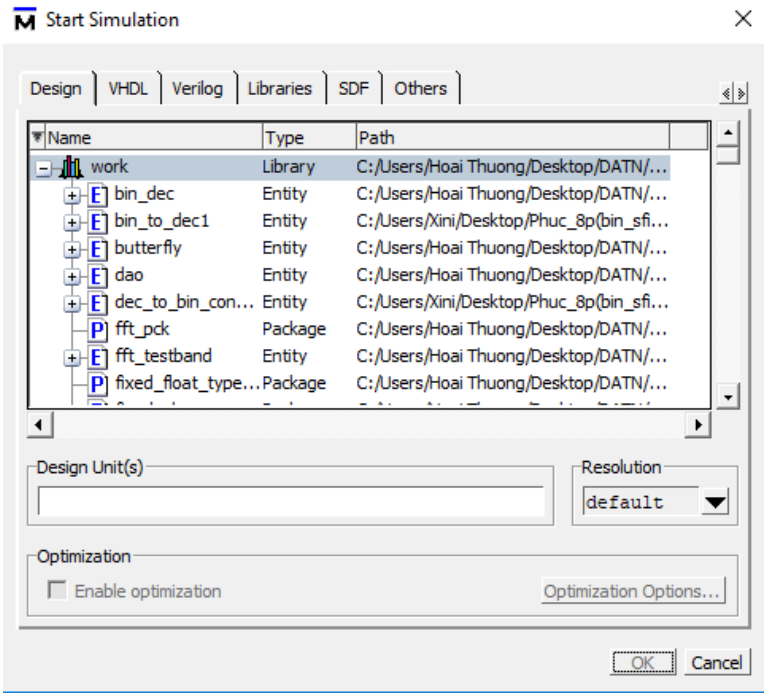
Bước 4: Nếu có file sẵn (VHDL, Verilog) thì chọn Add Existing File và thêm File. Nếu chưa có, chọn Create New File để thực hiện việc viết code.



Hình 3-8: Giao diện thêm file có sẵn cho project

Bước 5: Sau khi đã thêm code và tạo project, chọn Compile → Compile All.

Bước 6: Chọn Simulate → Start simulation → Chọn File code mô phỏng.



Hình 3-9: Chọn file bắt đầu mô phỏng

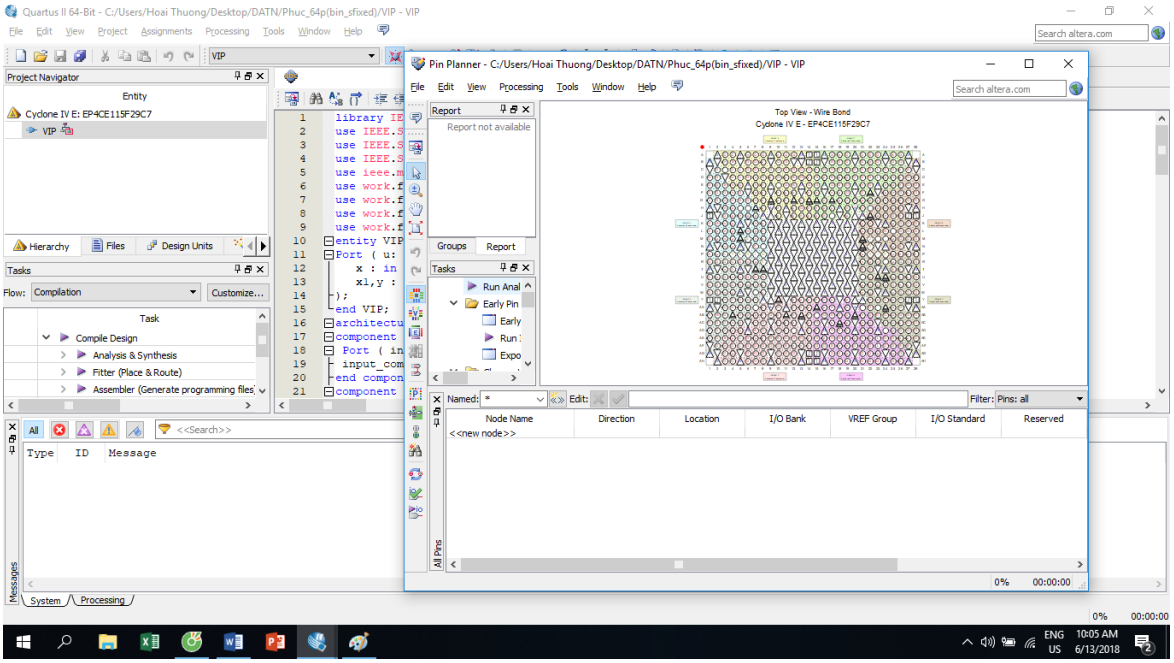
Bước 7: Chọn Add → To Wave. Màn hình sẽ xuất kết quả mô phỏng.

3.2.4 Xác định và nạp vị trí các chân

Bước 1: Chọn Assignment → Pin Planner hoặc chọn  trên giao diện để thiết lập vị trí chân.

Lưu ý: Nên compile trước 1 lần các chân trong trang thiết lập chân xuất hiện.

Bước 2: Màn hình hiện ra trang thiết lập chân.



Hình 3-10: Giao diện gán chân cho kit

Bước 3: Bắt đầu gán chânn cho các biến được tạo trên code.

Bước 4: Compile lần nữa để hoàn thành việc thiết lập.

CHƯƠNG 4. KẾT QUẢ VÀ SO SÁNH

4.1 FFT 64-điểm:

Ví dụ: Cho ngõ vào là biến x có giá trị x = [1, 3, 4, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Kết quả ngõ ra y được thể hiện ở mô phỏng MATLAB và mô phỏng Model-Sim, Quartus.

4.1.1 MATLAB

```
y =

  Columns 1 through 7

  73.0000 + 0.0000i   8.7605 - 1.9424i   8.0591 - 3.7605i   6.9459 - 5.3393i   5.4998 - 6.5822i   3.8226 - 7.4179i   2.0306 - 7.8059i

  Columns 8 through 14

   0.2457 - 7.7388i  -1.4142 - 7.2426i  -2.8442 - 6.3740i  -3.9601 - 5.2149i  -4.7047 - 3.8662i  -5.0515 - 2.4383i  -5.0059 - 1.0430i

  Columns 15 through 21

  -4.6037 + 0.2163i  -3.9075 + 1.2521i  -3.0000 + 2.0000i  -1.9773 + 2.4227i  -0.9395 + 2.5124i   0.0171 + 2.2904i   0.8088 + 1.8043i

  Columns 22 through 28

   1.3712 + 1.1226i   1.6640 + 0.3283i   1.6736 - 0.4893i   1.4142 - 1.2426i   0.9248 - 1.8541i   0.2655 - 2.2626i  -0.4892 - 2.4291i

  Columns 29 through 35

  -1.2572 - 2.3396i  -1.9570 - 2.0059i  -2.5158 - 1.4644i  -2.8758 - 0.7719i  -3.0000 + 0.0000i  -2.8758 + 0.7719i  -2.5158 + 1.4644i

  Columns 36 through 42

  -1.9570 + 2.0059i  -1.2572 + 2.3396i  -0.4892 + 2.4291i   0.2655 + 2.2626i   0.9248 + 1.8541i   1.4142 + 1.2426i   1.6736 + 0.4893i

  Columns 43 through 49

   1.6640 - 0.3283i   1.3712 - 1.1226i   0.8088 - 1.8043i   0.0171 - 2.2904i  -0.9395 - 2.5124i  -1.9773 - 2.4227i  -3.0000 - 2.0000i

  Columns 50 through 56

  -3.9075 - 1.2521i  -4.6037 - 0.2163i  -5.0059 + 1.0430i  -5.0515 + 2.4383i  -4.7047 + 3.8662i  -3.9601 + 5.2149i  -2.8442 + 6.3740i

  Columns 57 through 63

  -1.4142 + 7.2426i   0.2457 + 7.7388i   2.0306 + 7.8059i   3.8226 + 7.4179i   5.4998 + 6.5822i   6.9459 + 5.3393i   8.0591 + 3.7605i

  Column 64

   8.7605 + 1.9424i
```

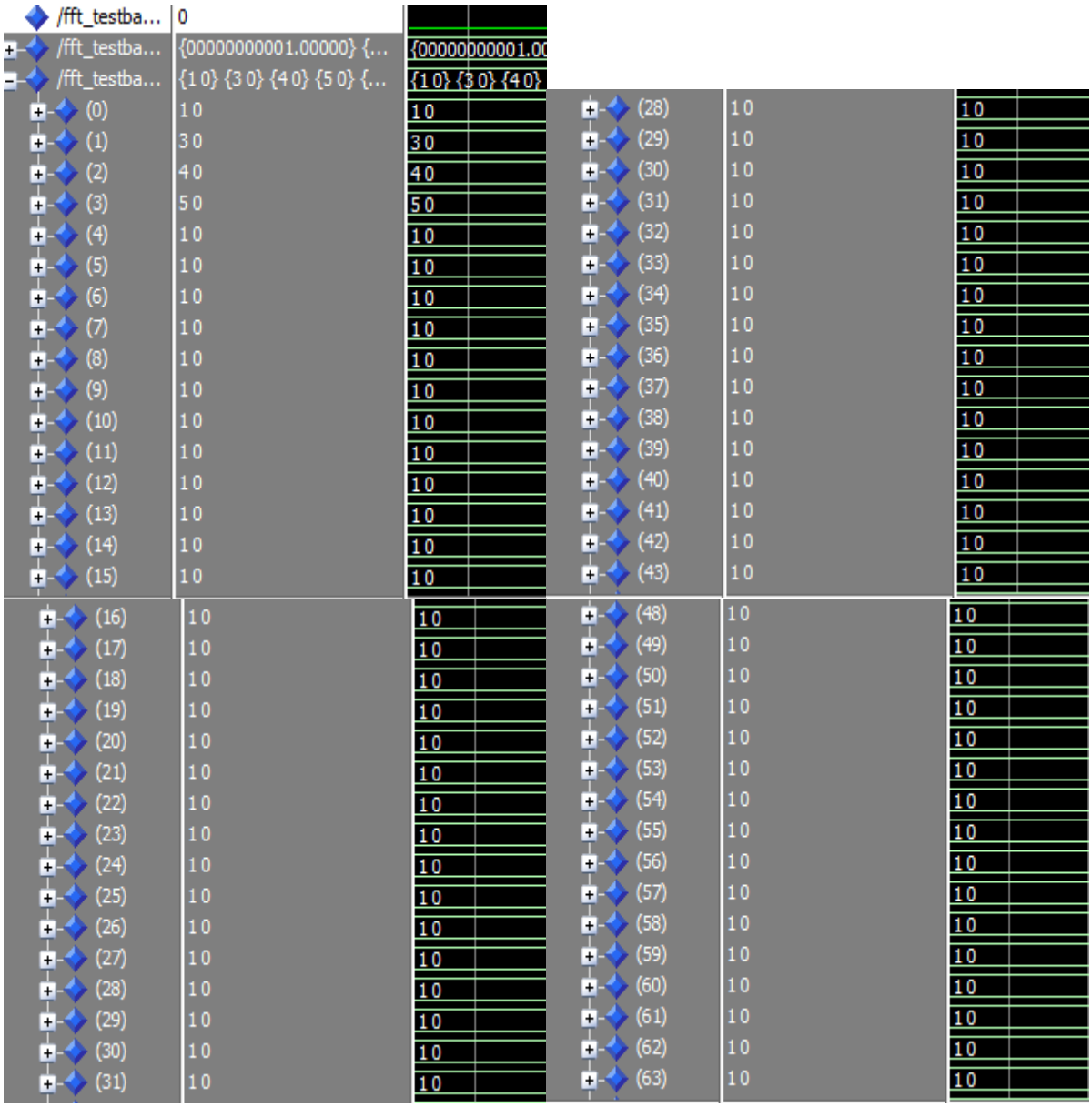
Hình 4-1: Kết quả FFT 64-điểm (MATLAB)

4.1.2 Quartus (ModelSim)

Với giá trị u là bit nhị phân được mô tả như sau:

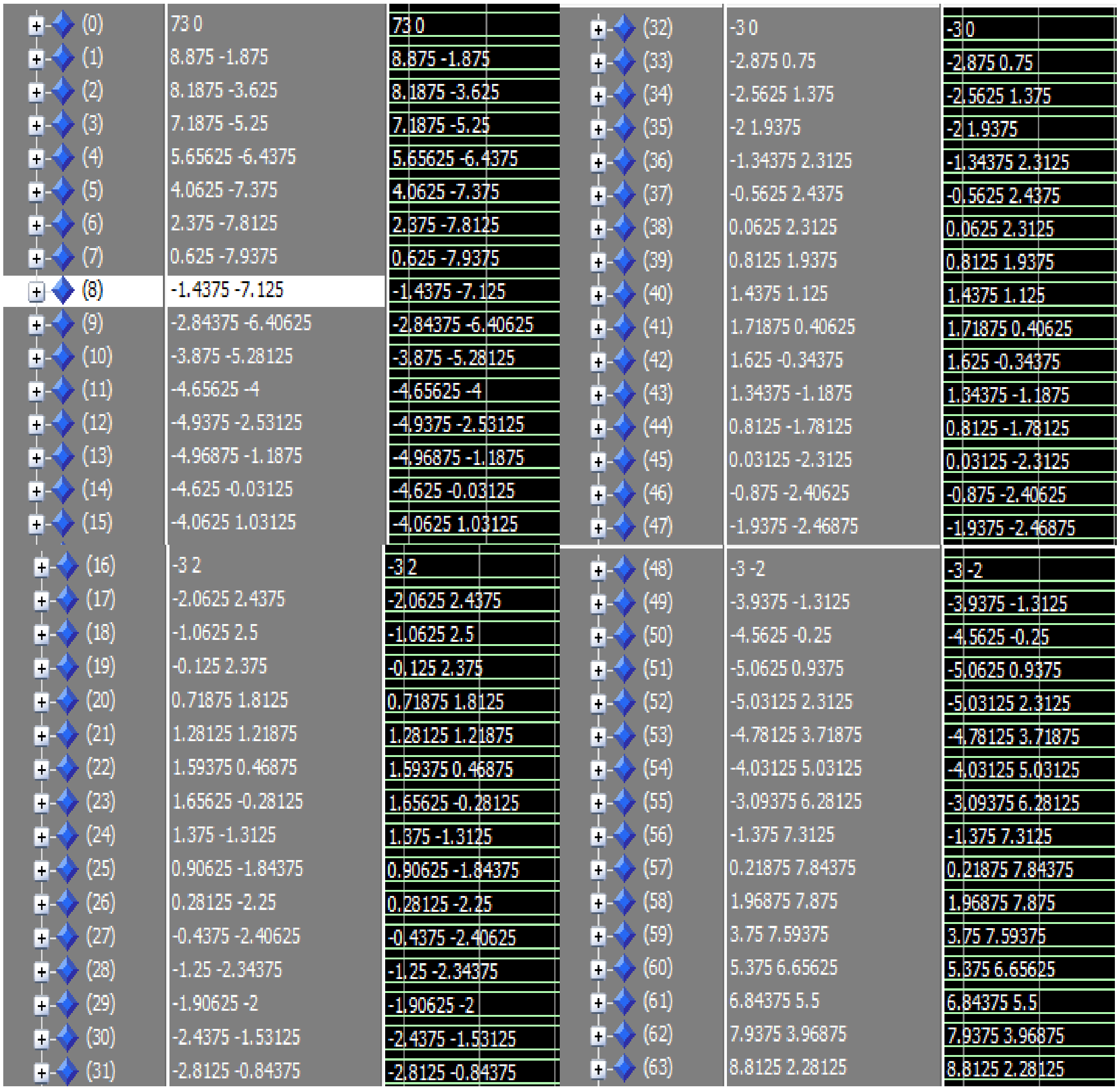
Khi u = ‘0’, bộ sẽ thực hiện chế độ FFT cho chuỗi số ngõ vào x để xuất chuỗi số ngõ ra y.

Lưu ý: Phần thực và phần ảo sẽ cách nhau bằng dấu cách.



Hình 4-2: Ngõ vào x và bit điều khiển u = '0'

Lúc này, chương trình sẽ thực hiện bộ FFT vì u = '0'.



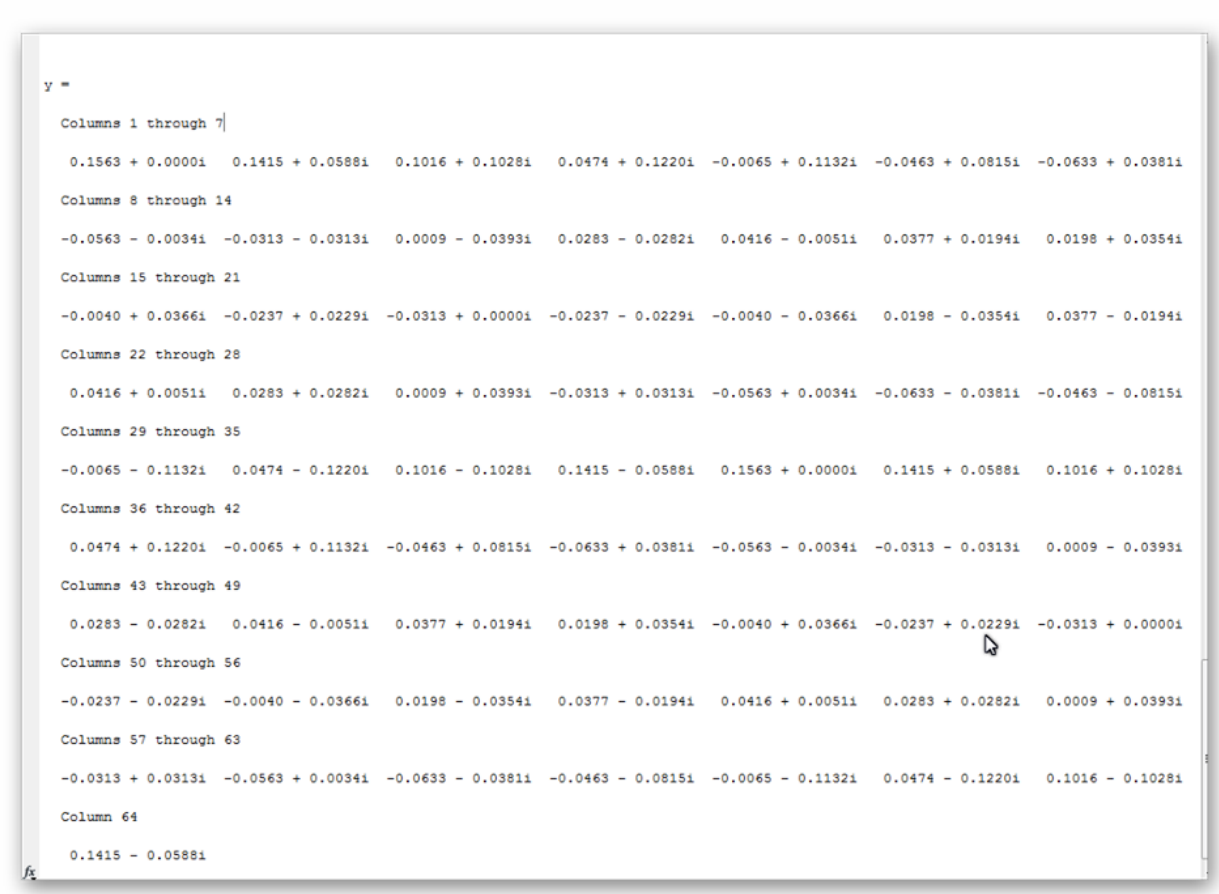
Hình 4-3: Kết quả FFT 64 điểm (ModelSim)

4.2 IFFT 64-điểm:

Ví dụ: cho ngõ vào là biến x có giá trị x = [1, 0, 2, 0, 3, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Kết quả ngõ ra y được thể hiện ở mô phỏng MATLAB và mô phỏng Model-Sim, Quartus:

4.2.1 MATLAB

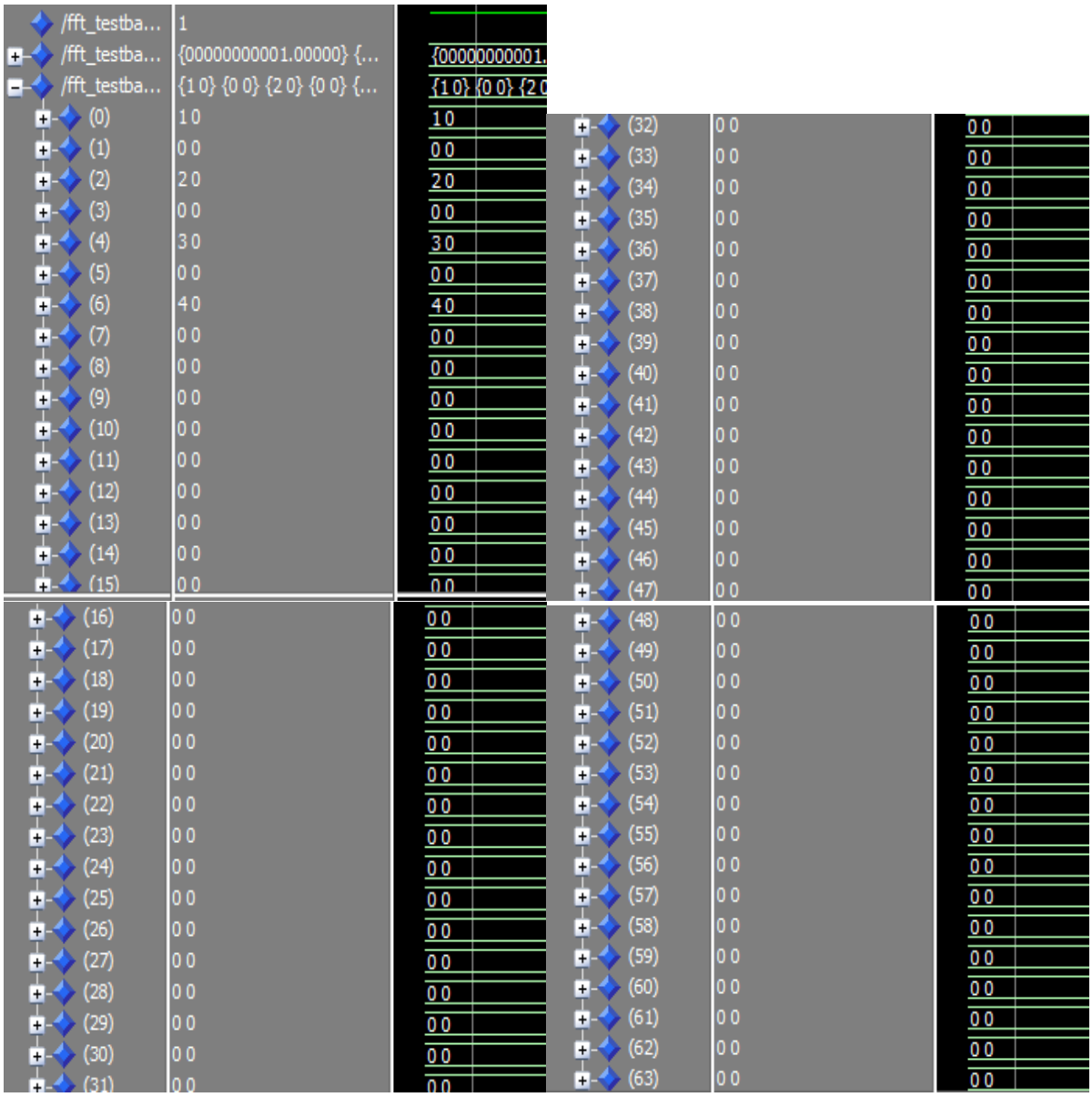


Hình 4-4: Kết quả IFFT 64-điểm (MATLAB)

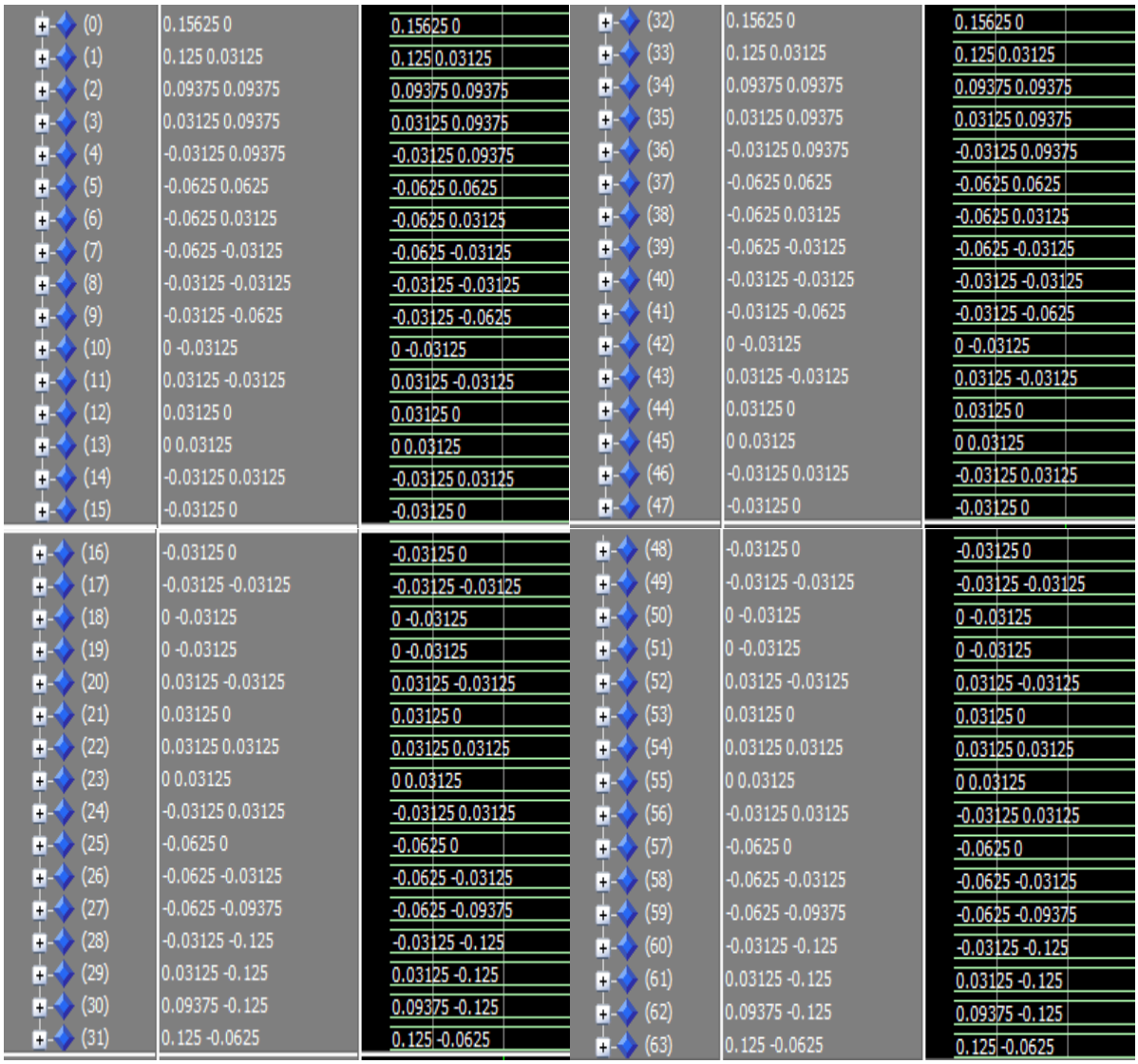
4.2.2 Quartus (ModelSim)

Với giá trị u là bit nhị phân được mô tả như sau:

Khi u = ‘1’, bộ sẽ thực hiện IFFT cho chuỗi số ngõ vào x để xuất chuỗi số ngõ ra y.



Hình 4-5: Ngõ vào x và bit điều khiển u = ’1’



(0)	0.15625 0	0.15625 0	(32)	0.15625 0	0.15625 0
(1)	0.125 0.03125	0.125 0.03125	(33)	0.125 0.03125	0.125 0.03125
(2)	0.09375 0.09375	0.09375 0.09375	(34)	0.09375 0.09375	0.09375 0.09375
(3)	0.03125 0.09375	0.03125 0.09375	(35)	0.03125 0.09375	0.03125 0.09375
(4)	-0.03125 0.09375	-0.03125 0.09375	(36)	-0.03125 0.09375	-0.03125 0.09375
(5)	-0.0625 0.0625	-0.0625 0.0625	(37)	-0.0625 0.0625	-0.0625 0.0625
(6)	-0.0625 0.03125	-0.0625 0.03125	(38)	-0.0625 0.03125	-0.0625 0.03125
(7)	-0.0625 -0.03125	-0.0625 -0.03125	(39)	-0.0625 -0.03125	-0.0625 -0.03125
(8)	-0.03125 -0.03125	-0.03125 -0.03125	(40)	-0.03125 -0.03125	-0.03125 -0.03125
(9)	-0.03125 -0.0625	-0.03125 -0.0625	(41)	-0.03125 -0.0625	-0.03125 -0.0625
(10)	0 -0.03125	0 -0.03125	(42)	0 -0.03125	0 -0.03125
(11)	0.03125 -0.03125	0.03125 -0.03125	(43)	0.03125 -0.03125	0.03125 -0.03125
(12)	0.03125 0	0.03125 0	(44)	0.03125 0	0.03125 0
(13)	0 0.03125	0 0.03125	(45)	0 0.03125	0 0.03125
(14)	-0.03125 0.03125	-0.03125 0.03125	(46)	-0.03125 0.03125	-0.03125 0.03125
(15)	-0.03125 0	-0.03125 0	(47)	-0.03125 0	-0.03125 0
(16)	-0.03125 0	-0.03125 0	(48)	-0.03125 0	-0.03125 0
(17)	-0.03125 -0.03125	-0.03125 -0.03125	(49)	-0.03125 -0.03125	-0.03125 -0.03125
(18)	0 -0.03125	0 -0.03125	(50)	0 -0.03125	0 -0.03125
(19)	0 -0.03125	0 -0.03125	(51)	0 -0.03125	0 -0.03125
(20)	0.03125 -0.03125	0.03125 -0.03125	(52)	0.03125 -0.03125	0.03125 -0.03125
(21)	0.03125 0	0.03125 0	(53)	0.03125 0	0.03125 0
(22)	0.03125 0.03125	0.03125 0.03125	(54)	0.03125 0.03125	0.03125 0.03125
(23)	0 0.03125	0 0.03125	(55)	0 0.03125	0 0.03125
(24)	-0.03125 0.03125	-0.03125 0.03125	(56)	-0.03125 0.03125	-0.03125 0.03125
(25)	-0.0625 0	-0.0625 0	(57)	-0.0625 0	-0.0625 0
(26)	-0.0625 -0.03125	-0.0625 -0.03125	(58)	-0.0625 -0.03125	-0.0625 -0.03125
(27)	-0.0625 -0.09375	-0.0625 -0.09375	(59)	-0.0625 -0.09375	-0.0625 -0.09375
(28)	-0.03125 -0.125	-0.03125 -0.125	(60)	-0.03125 -0.125	-0.03125 -0.125
(29)	0.03125 -0.125	0.03125 -0.125	(61)	0.03125 -0.125	0.03125 -0.125
(30)	0.09375 -0.125	0.09375 -0.125	(62)	0.09375 -0.125	0.09375 -0.125
(31)	0.125 -0.0625	0.125 -0.0625	(63)	0.125 -0.0625	0.125 -0.0625

Hình 4-6: Kết quả IFFT 64 điểm (ModelSim)

CHƯƠNG 5. KẾT QUẢ TRÊN KIT DE2-115

5.1 Quá trình thực hiện kết nối và nhập dữ liệu

5.1.1 Gán chân cho kit

Bước 1: Vào Assignment → Pin Planner.

Bước 2: Chọn kết nối chân dựa trên bảng danh sách các chân kết nối DE2-115.

Bước 3: Compile chương trình.

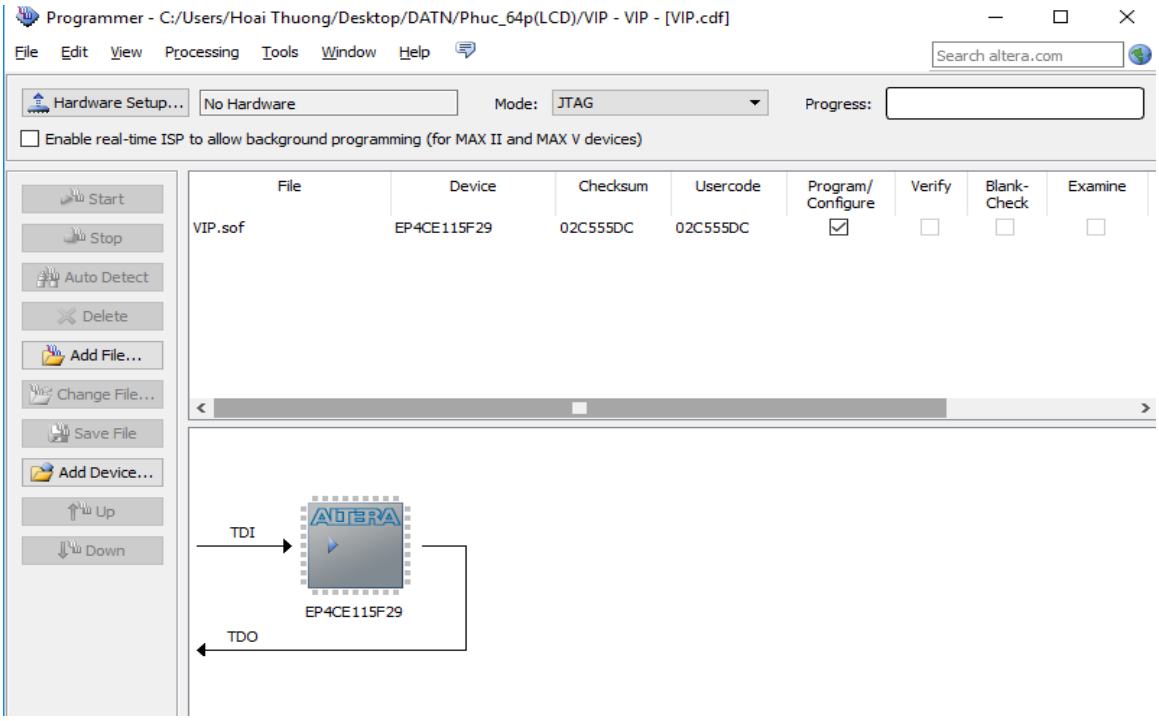
5.1.2 Nạp code

Sau khi đã Compile chương trình, bắt đầu quá trình nạp code vào kit.

Bước 1: Chọn Tools → Programmable.

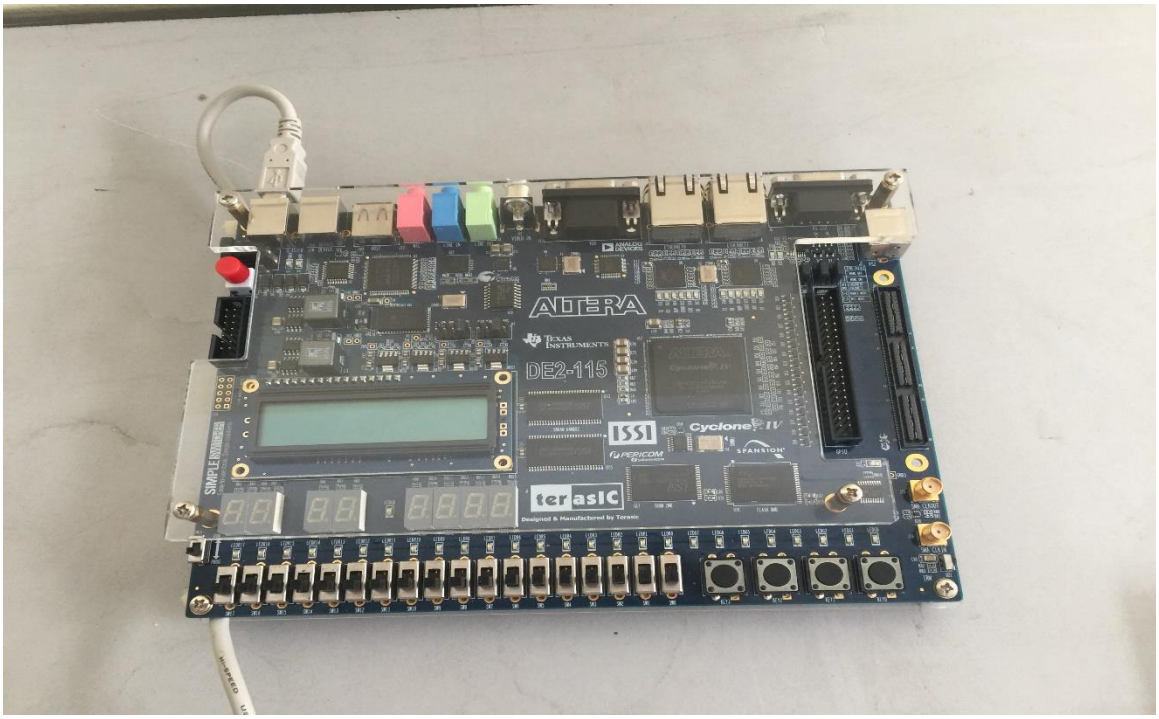
Bước 2: Bấm Start để bắt đầu quá trình nạp code.

Lưu ý: Nếu không thể bắt đầu, ta kiểm tra máy tính đã có driver của Quartus II hay chưa. Nếu chưa, ta thực hiện việc dò driver của chương trình.



Hình 5-1: Quá trình nạp code

Vì dữ liệu ngõ vào của bộ là nhị phân, người sử dụng sẽ nhập dữ liệu ngõ vào x bằng 11 công tắc từ SW0 đến SW11. Để thay đổi dữ liệu thứ n của x, người sử dụng bấm key 1 để tăng số n của x, tùy thuộc và số muốn tăng sẽ nhấn bấy nhiêu lần và thay đổi dữ liệu ngõ vào tùy ý. Ví dụ, để thay đổi x(0) đến x(1) để nhập dữ liệu ngõ vào tiếp theo, người dùng nhấn 1 lần key0.



Hình 5-2: Kit DE2-115 và chức năng

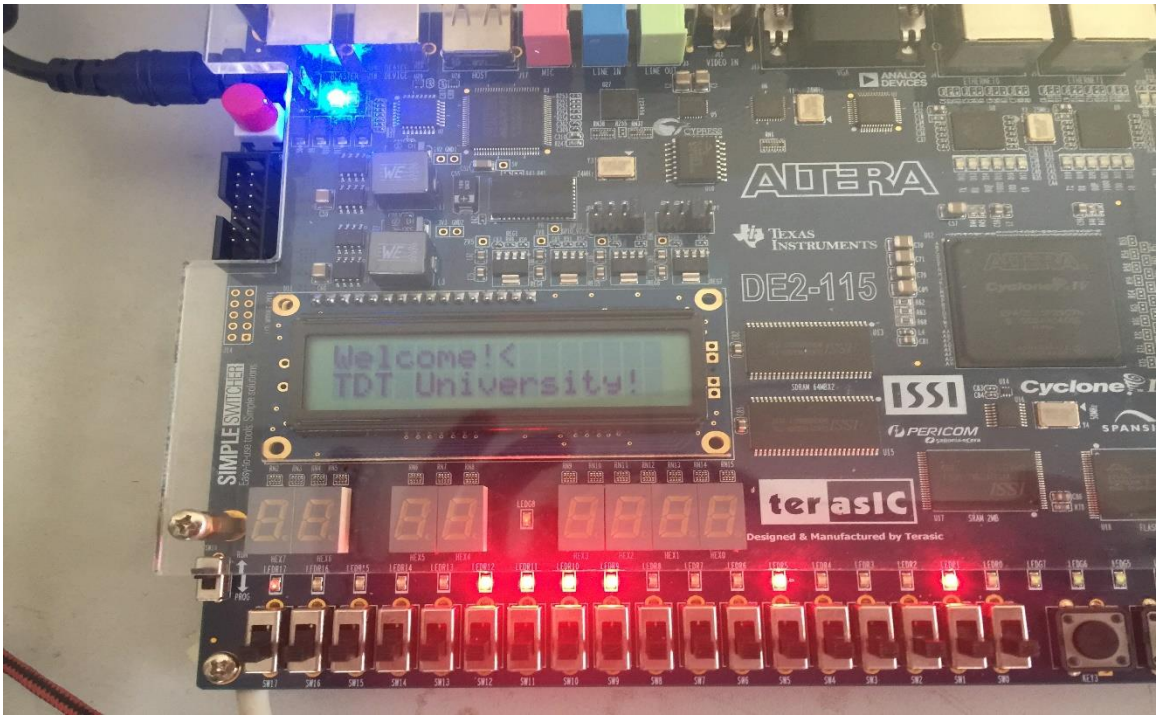
Chức năng:

- SW 0 đến 11: Dữ liệu ngõ vào x.
- SW 16: Điều chỉnh tăng giảm số n. Bit 1 tăng và bit 0 giảm.
- SW 17: Điều chỉnh bộ FFT/IFFT. Bit 0 là FFT và bit 1 là IFFT.
- Key 0: Tăng và giảm ngõ vào x thứ n.
- Key 1: Tăng và giảm ngõ ra y thứ n.
- Key 2: Điều chỉnh hiển thị của LCD.
- Key 3: Reset toàn bộ dữ liệu.

- LED Red 0 → 11: Hiển thị dữ liệu ngõ vào (nhị phân).
- LED Red 16: Hiển thị sự tăng và giảm thứ n. Nếu bit '1' sẽ tăng, bit '0' sẽ giảm.
- LED Red 17: Hiển thị chế độ FFT & IFFT. Nếu bit '0' là chế độ FFT, bit '1' là chế độ IFFT.
- LED Green 0 → 5: Hiển thị số thứ n của ngõ vào x (nhị phân).
- Hex0 → 3: Hiển thị số thứ n của ngõ vào x (thập phân).

5.2 Kết quả

Sau khi quá trình nạp kết quả thành công, LCD của kit hiển thị dòng chữ sau:



Hình 5-3: Kit hiển thị dòng chữ ban đầu của chương trình

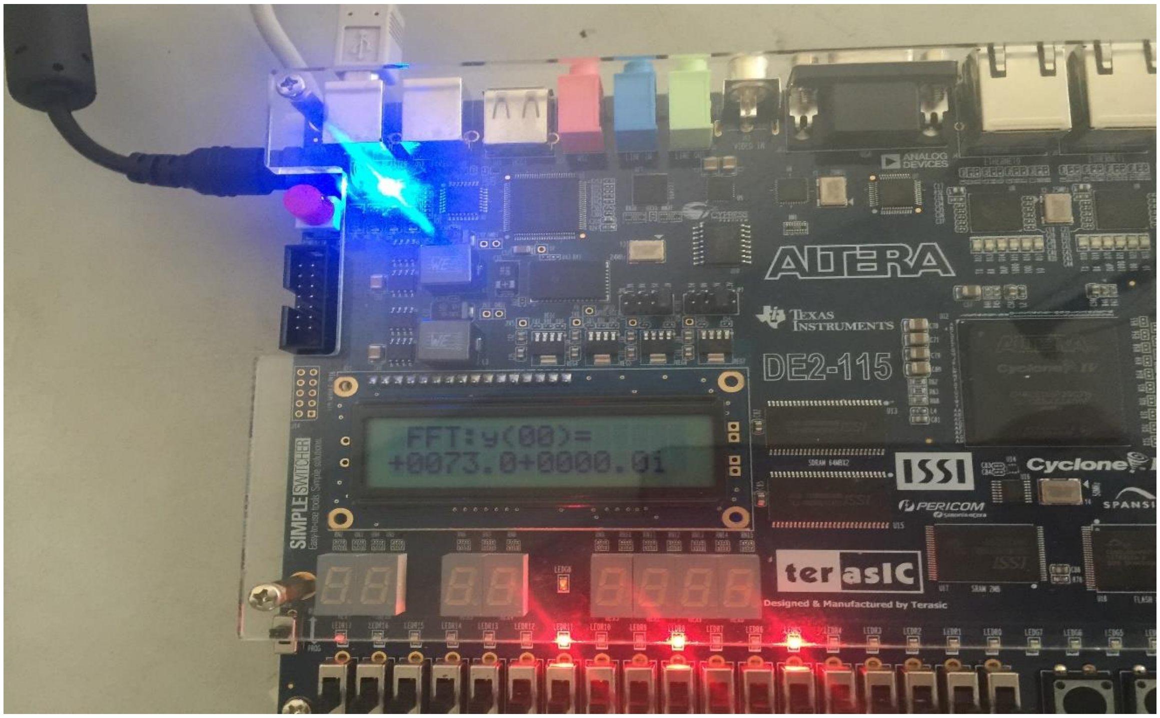
Sau khi nhấn Key2, kit sẽ chuyển sang chế độ làm việc theo bộ FFT/IFFT.

5.2.1 Bộ biến đổi FFT:

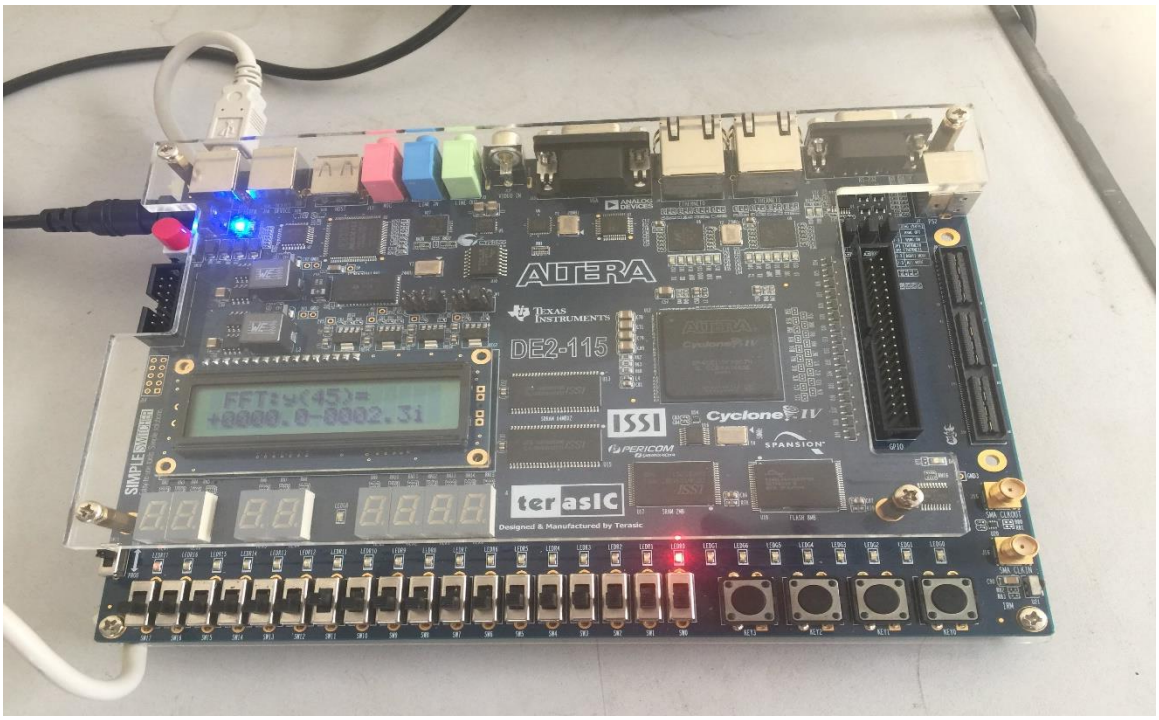
Sau khi đã bật chế độ FFT và điều chỉnh ngõ vào x với dữ liệu:

x = [1, 3, 4, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

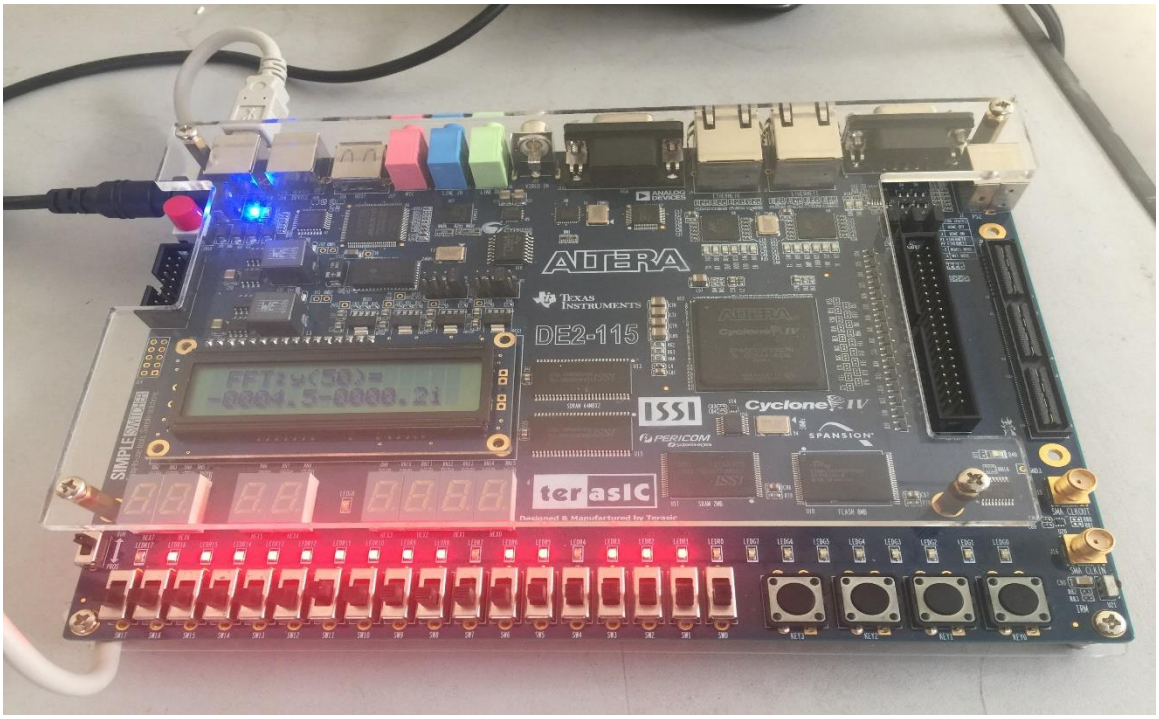
Kết quả ngõ ra FFT như sau:



Hình 5-4: FFT ngõ ra y(0)



Hình 5-5: FFT ngõ ra y(45)



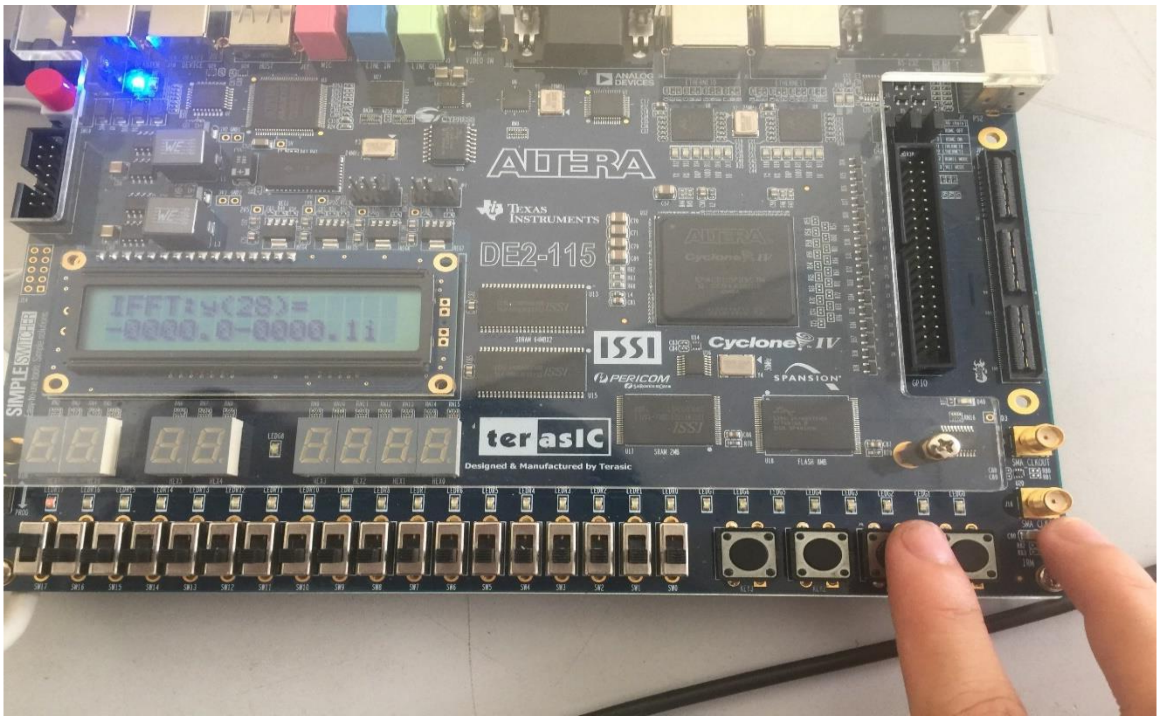
Hình 5-6: FFT ngõ ra y(50)

5.2.2 Bộ biến đổi IFFT

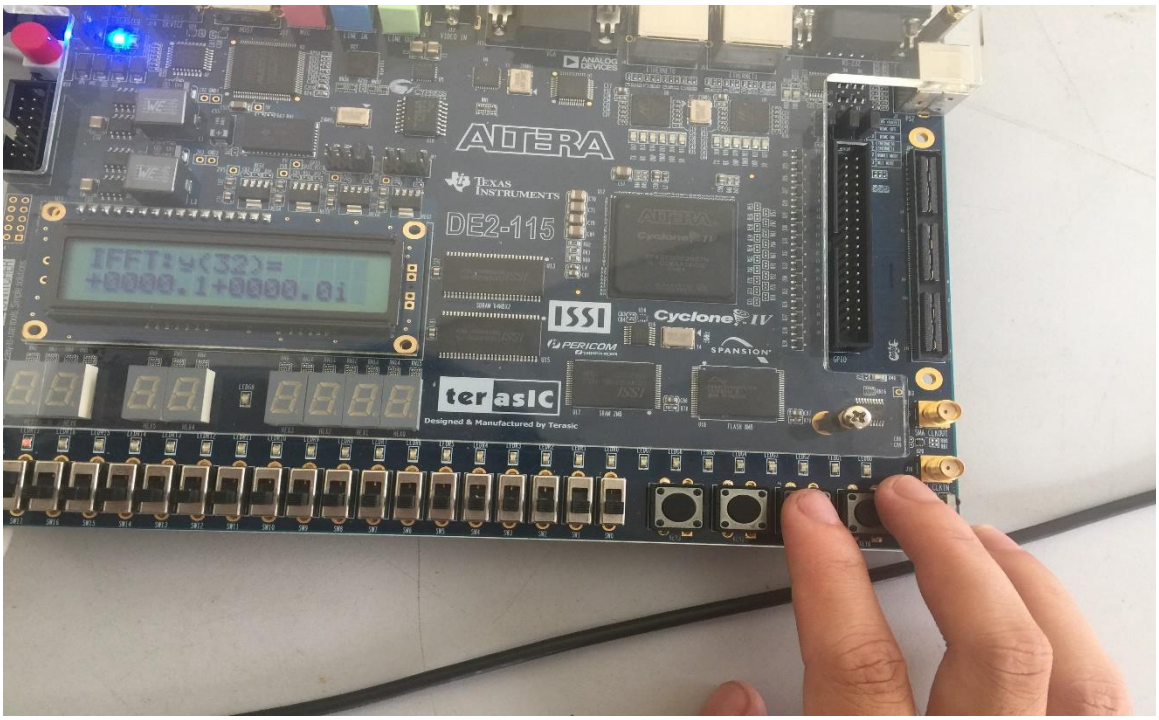
Sau khi đã bật chế độ IFFT và điều chỉnh ngõ ra x như sau:

x = [1, 0, 2, 0, 3, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

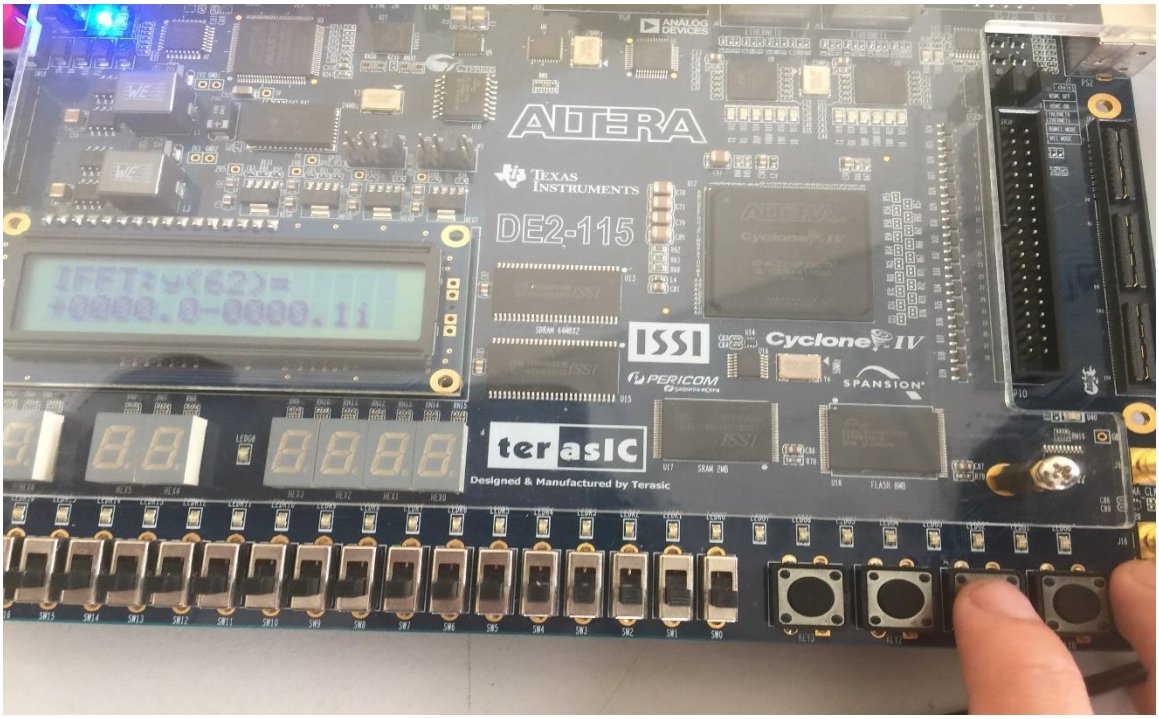
Kết quả ngõ ra như sau:



Hình 5-7: IFFT ngõ ra y(28)



Hình 5-8: IFFT ngõ ra y(32)



Hình 5-9: IFFT ngõ ra y(62)

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Việc thiết kế bộ FFT & IFFT 64 điểm trên nền công nghệ FPGA là một ứng dụng chứng minh sự đa dụng của công nghệ này. Dù không thể ra kết quả chính xác tuyệt đối như trong MATLAB, kết quả cho thấy công nghệ FPGA có thể thực hiện cho việc tính toán bậc cao và rất tốt cho việc nghiên cứu và phát triển. Ngoài ra, việc xử lý và kiểm tra của kết quả cũng tương đối nhanh với chip Cyclone IV.

Việc thiết kế và đưa code lên kit DE2-115 để thực hiện chức năng như 1 bộ FFT và IFFT có thể cho thấy sự đa dạng và hoạt động tốt của dòng công nghệ FPGA nói chung và kit DE2-115 nói riêng.

6.1.1 Ưu điểm:

- Có thể tùy ý thiết kế theo ý người dùng.
- Có thể ứng dụng nhiều bộ xử lý số khác trên nền FPGA 1 cách dễ dàng.
- Sử dụng được nhiều loại ngôn ngữ lập trình khác nhau (AHDL, VHDL, Verilog HDL,…).
- Giá thành tương đối phù hợp với việc nghiên cứu trong trường đại học và sinh viên.
- Nhiều thư viện code được thiết kế sẵn trong chương trình rất hữu dụng.

6.1.2 Nhược điểm:

- Độ chính xác không được cao như MATLAB.
- Tốn nhiều thời gian tổng hợp (Compile).
- Code phức tạp và nhiều.
- Không thể tổng hợp với số thực.

6.2 Hướng phát triển

Việc thực hiện FFT & IFFT đến 64 điểm trên nền FPGA có kết quả với độ chính xác tương đối cao, người lập trình còn có thể nâng lên bậc cao hơn và có thể đến 1024 điểm và cao hơn nữa. Ngoài ra, FFT & IFFT có thể sử dụng các phương thức khác như Radix-4, Radix-mix,… để thực hiện việc xử lý. Các phương thức khác nhau sẽ có ưu nhược điểm khác nhau. Nhưng sự khác biệt của các phương thức chỉ ảnh hưởng đến tốc độ thực hiện và độ phức tạp về code, tài nguyên sử dụng và việc tổng hợp code.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

[1] Lê Tiến Thường "Xử Lý Số Tín Hiệu Và Wavelets", trang 376 – 420, 2015.

[2] http://www.dientuvietnam.net

Tiếng Anh:

[3] http://www.terasic.com.tw

[4] Goran S. Nikolic, Milorad D. Cakic and Dragan J. Cvetkovic "Fourier Transforms – High-tech Application and Current Trends", Chapter 4, February 2017.

[5] Richard G.Lyons "Understanding Digital Signal Processing (2nd Edition)" Chapter 4, 2004.

[6] https://cnx.org/

[7] Terasic.com "DE2-115 User Manual" Chapter 4, 29 – 40 page, 2003 – 2010.

PHỤ LỤC A

File chính

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
entity VIP is
Port ( reset :in std_logic;
     u: in std_logic;
     inp,outp,updown : in std_logic;
     x : in sfixed(11 downto 0);
     y : out complex
);
end VIP;
architecture Behavioral of VIP is
component real_to_complex is
 Port ( input_real : in idata_array;
 input_comp : out odata_array2);
end component;
component butterfly is
 port( u: in std_logic;
 c1,c2 : in complex; --inputs
 w,iw :in complex; -- phase factor
 g1,g2 :out complex -- outputs
 );
 end component;
 component TWiddle_Factor is
 port( W,iW :out comp_array999 -- outputs
 );
end component;
component P32
Port (  u: in std_logic;
     x : in odata_array1;
```

```
        y : out odata_array1
    );
    end component;
    component Dao
    Port ( N: in integer;
        x : in odata_array2;
        y : out odata_array2
    );
    end component;
    signal w,iw: comp_array999;
    signal s: odata_array2 := (others =>
    ("0000000000000000","0000000000000000"));
    signal g1,g2,f1,f2: odata_array1 := (others =>
    ("0000000000000000","0000000000000000"));
    signal y1,y2 : odata_array2 := (others =>
    ("0000000000000000","0000000000000000"));
    signal a : sfixed(11 downto -5) := (others => '0');
    signal ii,oo : integer := 0;
    constant N : integer := 64;
    signal x1 : idata_array := (others => "0000000000000000");
    begin
    process(inp,ii,reset)
    begin
        if( reset = '0') then
        ii <= 0;
        x1 <= (
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
```

```
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000",
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000","0000000000000000"  );
        else
        if (falling_edge(inp)) then
                if(updown ='1') then
                        if (ii < 63) then
                                ii <= ii + 1;
                        else ii <= 0;
                        end if;
                else
                        if (ii > 0) then
                                ii <= ii - 1;
                        else ii <= 63;
                        end if;
                end if;
    end if;
        a(11 downto 0) <= x;
        a(-1 downto -5) <= "00000";
        x1(ii) <= a;
        end if;
    end process;

    TF : TWiddle_Factor port map (w,iw);
    --Real Input value transformed in complex value
    sixteen_bits_complex_transform1 : real_to_complex port map(x1, s);

    --first stage of butterfly's.
    bf1 : butterfly port map(u,s(0),s(32),w(0),iw(0),g1(0),g2(0));
    bf2 : butterfly port map(u,s(1),s(33),w(16),iw(16),g1(1),g2(1));
    bf3 : butterfly port map(u,s(2),s(34),w(32),iw(32),g1(2),g2(2));
    bf4 : butterfly port map(u,s(3),s(35),w(48),iw(48),g1(3),g2(3));
```

```
bf5 : butterfly port map(u,s(4),s(36),w(64),iw(64),g1(4),g2(4));
bf6 : butterfly port map(u,s(5),s(37),w(80),iw(80),g1(5),g2(5));
bf7 : butterfly port map(u,s(6),s(38),w(96),iw(96),g1(6),g2(6));
bf8 : butterfly port map(u,s(7),s(39),w(112),iw(112),g1(7),g2(7));
bf9 : butterfly port map(u,s(8),s(40),w(128),iw(128),g1(8),g2(8));
bf10: butterfly port map(u,s(9),s(41),w(144),iw(144),g1(9),g2(9));
bf11: butterfly port map(u,s(10),s(42),w(160),iw(160),g1(10),g2(10));
bf12: butterfly port map(u,s(11),s(43),w(176),iw(176),g1(11),g2(11));
bf13: butterfly port map(u,s(12),s(44),w(192),iw(192),g1(12),g2(12));
bf14: butterfly port map(u,s(13),s(45),w(208),iw(208),g1(13),g2(13));
bf15: butterfly port map(u,s(14),s(46),w(224),iw(224),g1(14),g2(14));
bf16: butterfly port map(u,s(15),s(47),w(240),iw(240),g1(15),g2(15));
bf17: butterfly port map(u,s(16),s(48),w(256),iw(256),g1(16),g2(16));
bf18: butterfly port map(u,s(17),s(49),w(272),iw(272),g1(17),g2(17));
bf19: butterfly port map(u,s(18),s(50),w(288),iw(288),g1(18),g2(18));
bf20: butterfly port map(u,s(19),s(51),w(304),iw(304),g1(19),g2(19));
bf21: butterfly port map(u,s(20),s(52),w(320),iw(320),g1(20),g2(20));
bf22: butterfly port map(u,s(21),s(53),w(336),iw(336),g1(21),g2(21));
bf23: butterfly port map(u,s(22),s(54),w(352),iw(352),g1(22),g2(22));
bf24: butterfly port map(u,s(23),s(55),w(368),iw(368),g1(23),g2(23));
bf25: butterfly port map(u,s(24),s(56),w(384),iw(384),g1(24),g2(24));
bf26: butterfly port map(u,s(25),s(57),w(400),iw(400),g1(25),g2(25));
bf27: butterfly port map(u,s(26),s(58),w(416),iw(416),g1(26),g2(26));
bf28: butterfly port map(u,s(27),s(59),w(432),iw(432),g1(27),g2(27));
bf29: butterfly port map(u,s(28),s(60),w(448),iw(448),g1(28),g2(28));
bf30: butterfly port map(u,s(29),s(61),w(464),iw(464),g1(29),g2(29));
bf31: butterfly port map(u,s(30),s(62),w(480),iw(480),g1(30),g2(30));
bf32: butterfly port map(u,s(31),s(63),w(496),iw(496),g1(31),g2(31));
bff1: P32 port map(u,g1,f1);
bff2: P32 port map(u,g2,f2);
process (f1,f2)
begin
for i in 0 to 31 loop
    y1(i) <= f1(i);
    y1(32+i) <= f2(i);
    end loop;
end process;
convert0 : Dao port map (N,y1,y2);
process(y2,outp,oo,reset)
```

```
begin
if( reset = '0') then
    oo <= 0;
    else
    if falling_edge(outp) then
            if(updown ='1') then
                    if (oo < 63) then
                            oo<= oo+1;
                    else oo <= 0;
                    end if;
            else
                    if (oo > 0) then
                            oo<= oo - 1;
                    else oo <= 63;
                    end if;
            end if;
    end if;
end if;
    y <= y2(oo);
end process;
end Behavioral;
```

File 32 điểm

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
entity P32 is
Port ( u: in std_logic;
    x : in odata_array1;
    y : out odata_array1
);
end P32;
architecture Behavioral of P32 is
component butterfly is
```

```
 port( u: in std_logic;
 c1,c2 : in complex; --inputs
 w,iw :in complex; -- phase factor
 g1,g2 :out complex -- outputs
 );
 end component;
component P16
Port ( u: in std_logic;
     x : in odata_array;
     y : out odata_array
);
end component;
 component TWiddle_Factor is
 port( W,iW :out comp_array999 -- outputs
 );
 end component;
signal w,iw: comp_array999;
signal g1,g2,f1,f2: odata_array := (others =>
("0000000000000000","0000000000000000"));
begin
TF : TWiddle_Factor port map (w,iw);
--first stage of butterfly's.
bf1 : butterfly port map(u,x(0),x(16),w(0),iw(0),g1(0),g2(0));
bf2 : butterfly port map(u,x(1),x(17),w(32),iw(32),g1(1),g2(1));
bf3 : butterfly port map(u,x(2),x(18),w(64),iw(64),g1(2),g2(2));
bf4 : butterfly port map(u,x(3),x(19),w(96),iw(96),g1(3),g2(3));
bf5 : butterfly port map(u,x(4),x(20),w(128),iw(128),g1(4),g2(4));
bf6 : butterfly port map(u,x(5),x(21),w(160),iw(160),g1(5),g2(5));
bf7 : butterfly port map(u,x(6),x(22),w(192),iw(192),g1(6),g2(6));
bf8 : butterfly port map(u,x(7),x(23),w(224),iw(224),g1(7),g2(7));
bf9 : butterfly port map(u,x(8),x(24),w(256),iw(256),g1(8),g2(8));
bf10: butterfly port map(u,x(9),x(25),w(288),iw(288),g1(9),g2(9));
bf11: butterfly port map(u,x(10),x(26),w(320),iw(320),g1(10),g2(10));
bf12: butterfly port map(u,x(11),x(27),w(352),iw(352),g1(11),g2(11));
bf13: butterfly port map(u,x(12),x(28),w(384),iw(384),g1(12),g2(12));
bf14: butterfly port map(u,x(13),x(29),w(416),iw(416),g1(13),g2(13));
bf15: butterfly port map(u,x(14),x(30),w(448),iw(448),g1(14),g2(14));
bf16: butterfly port map(u,x(15),x(31),w(480),iw(480),g1(15),g2(15));
```

```
bff1: P16 port map(u,g1,f1);
bff2: P16 port map(u,g2,f2);

process (f1,f2)
begin
for i in 0 to 15 loop
    y(i) <= f1(i);
    y(16+i) <= f2(i);
    end loop;
end process;
end Behavioral;
```

File 16 điểm

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
entity P16 is
Port ( u: in std_logic;
    x : in odata_array;
    y : out odata_array
);
end P16;
architecture Behavioral of P16 is
component butterfly is
 port( u: in std_logic;
 c1,c2 : in complex; --inputs
 w,iw :in complex; -- phase factor
 g1,g2 :out complex -- outputs
 );
end component;
component TWiddle_Factor is
 port( W,iW :out comp_array999 -- outputs
 );
 end component;
```

```
signal w,iw: comp_array999;
signal g,f,h: odata_array := (others =>
("0000000000000000","0000000000000000"));
begin
TF : TWiddle_Factor port map (w,iw);
--first stage of butterfly's.
bf11 : butterfly port map(u,x(0),x(8),w(0),iw(0),g(0),g(8));
bf12 : butterfly port map(u,x(1),x(9),w(64),iw(64),g(1),g(9));
bf13 : butterfly port map(u,x(2),x(10),w(128),iw(128),g(2),g(10));
bf14 : butterfly port map(u,x(3),x(11),w(192),iw(192),g(3),g(11));
bf15 : butterfly port map(u,x(4),x(12),w(256),iw(256),g(4),g(12));
bf16 : butterfly port map(u,x(5),x(13),w(320),iw(320),g(5),g(13));
bf17 : butterfly port map(u,x(6),x(14),w(384),iw(384),g(6),g(14));
bf18 : butterfly port map(u,x(7),x(15),w(448),iw(448),g(7),g(15));
--second stage of butterfly's.
bf21 : butterfly port map(u,g(0) ,g(4) ,w(0),iw(0),f(0),f(4));
bf22 : butterfly port map(u,g(1) ,g(5) ,w(128),iw(128),f(1),f(5));
bf23 : butterfly port map(u,g(2) ,g(6) ,w(256),iw(256),f(2),f(6));
bf24 : butterfly port map(u,g(3) ,g(7) ,w(384),iw(384),f(3),f(7));
bf25 : butterfly port map(u,g(8) ,g(12),w(0),iw(0),f(8),f(12));
bf26 : butterfly port map(u,g(9) ,g(13),w(128),iw(128),f(9),f(13));
bf27 : butterfly port map(u,g(10),g(14),w(256),iw(256),f(10),f(14));
bf28 : butterfly port map(u,g(11),g(15),w(384),iw(384),f(11),f(15));

--third stage of butterfly's.
bf31 : butterfly port map(u,f(0) ,f(2) ,w(0),iw(0),h(0),h(2));
bf32 : butterfly port map(u,f(1) ,f(3) ,w(256),iw(256),h(1),h(3));
bf33 : butterfly port map(u,f(4) ,f(6) ,w(0),iw(0),h(4),h(6));
bf34 : butterfly port map(u,f(5) ,f(7) ,w(256),iw(256),h(5),h(7));
bf35 : butterfly port map(u,f(8) ,f(10),w(0),iw(0),h(8),h(10));
bf36 : butterfly port map(u,f(9) ,f(11),w(256),iw(256),h(9),h(11));
bf37 : butterfly port map(u,f(12),f(14),w(0),iw(0),h(12),h(14));
bf38 : butterfly port map(u,f(13),f(15),w(256),iw(256),h(13),h(15));

--fourth stage of butterfly's.
bf41 : butterfly port map(u,h(0) ,h(1) ,w(0),iw(0),y(0),y(1));
bf42 : butterfly port map(u,h(2) ,h(3) ,w(0),iw(0),y(2),y(3));
bf43 : butterfly port map(u,h(4) ,h(5) ,w(0),iw(0),y(4),y(5));
bf44 : butterfly port map(u,h(6) ,h(7) ,w(0),iw(0),y(6),y(7));
```

```
bf45 : butterfly port map(u,h(8) ,h(9) ,w(0),iw(0),y(8),y(9));
bf46 : butterfly port map(u,h(10),h(11),w(0),iw(0),y(10),y(11));
bf47 : butterfly port map(u,h(12),h(13),w(0),iw(0),y(12),y(13));
bf48 : butterfly port map(u,h(14),h(15),w(0),iw(0),y(14),y(15));
end Behavioral;
```

File hiển thị LCD

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
ENTITY LCD_hienthi IS
  PORT(
    reset            : IN     std_logic;
    clock_50         : IN     std_logic;
    lcd_rs           : OUT    std_logic;
    lcd_e            : OUT    std_logic;
    lcd_rw           : OUT    std_logic;
    lcd_on           : OUT    std_logic;
    lcd_blon         : OUT    std_logic;
    data_bus_0       : INOUT  STD_LOGIC;
    data_bus_1       : INOUT  STD_LOGIC;
    data_bus_2       : INOUT  STD_LOGIC;
    data_bus_3       : INOUT  STD_LOGIC;
    data_bus_4       : INOUT  STD_LOGIC;
    data_bus_5       : INOUT  STD_LOGIC;
    data_bus_6       : INOUT  STD_LOGIC;
    data_bus_7       : INOUT  STD_LOGIC;
    u,u1: in std_logic;
          led_u,led_updown :out std_logic;
          inp,outp,updown: in std_logic;
          x: in sfixed(11 downto 0);
          out1 : out sfixed(11 downto 0);
          out2 : out std_logic_vector(5 downto 0);
          HEX1,HEX2,HEX3,HEX4 : out std_logic_vector(6 downto 0)
```

```
  );
END entity ;
ARCHITECTURE a OF LCD_hienthi IS
  type character_string is array ( 0 to 31 ) of STD_LOGIC_VECTOR( 7 downto 0 );

  type state_type is (lap_ham, mo_hienthi, mode_set, print_string,
                      line2, return_home, drop_lcd_e, reset1, reset2,
                      reset3, tat_hienthi, display_clear );
  component VIP
  port(       reset: in std_LOGIC;
  u: in std_logic;
     inp,outp,updown: in std_logic;
     x: in sfixed(11 downto 0);
     y : out complex
  );
  end component;

  signal state, next_command         : state_type;
  signal lcd_display_string          : character_string;
  signal lcd_display_string_01       : character_string;
  signal lcd_display_string_02       : character_string;
  signal data_bus_value, ky_tu_ke    : STD_LOGIC_VECTOR(7 downto 0);
  signal clk_count_400hz             : STD_LOGIC_VECTOR(23 downto 0);
  signal char_count                  : STD_LOGIC_VECTOR(4 downto 0);
  signal clk_400hz_enable,lcd_rw_int : std_logic;
  signal data_bus,bcd1               : STD_LOGIC_VECTOR(7 downto 0);
  signal y1 : complex;
  signal yr,yi : sfixed(11 downto -5);
  signal bcdr,bcdi : std_logic_vector(19 downto 0);
  signal step1,step2,a,b : std_logic_vector(5 downto 0) := "000000";
  signal begin1 : std_logic := '0';
BEGIN
data_bus_0 <= data_bus(0);
data_bus_1 <= data_bus(1);
data_bus_2 <= data_bus(2);
data_bus_3 <= data_bus(3);
data_bus_4 <= data_bus(4);
data_bus_5 <= data_bus(5);
data_bus_6 <= data_bus(6);
```

```
data_bus_7 <= data_bus(7);
 led_u <= u;
 led_updown <= updown;
 yr <= y1.r;
 yi <= y1.i;
 out1 <= x;
 FFTIFFT : VIP port map (reset,u,inp,outp,updown,x,y1);
 bcdr <= to_bcd(yr);
 bcdi <= to_bcd(yi);
 process(outp,u,inp,updown,reset,bcdi,bcdr)
 begin
if(reset = '0') then
    step1 <= "000000";
    else
    if falling_edge(outp) then
            if (updown = '1') then
                    if (step1 < 63) then
                    step1 <= step1 + 1;
                    else step1 <= "000000";
                    end if;
            else
                    if (step1 > 0 ) then
                    step1 <= step1 - 1;
                    else step1 <= "111111";
                    end if;
            end if;
    end if;
end if;
if(reset = '0') then
    step2 <= "000000";
    else
    if falling_edge(inp) then
            if(updown = '1') then
                    if (step2 < 63) then
                    step2 <= step2 + 1;
                    else step2 <=  "000000";
                    end if;
            else
                    if (step2 > 0 ) then
```

```
                    step2 <= step2 - 1;
                    else step2 <=  "111111";
                    end if;
                end if;
        end if;
    end if;
    -------
    if (step2 > 59) then
                a <= step2 - 60;
                b <= "000110";
        elsif (step2 > 49) then
                a <= step2 - 50;
                b <= "000101";
        elsif (step2 > 39) then
                a <= step2 - 40;
                b <= "000100";
        elsif (step2 > 29) then
                a <= step2 - 30;
                b <= "000011";
        elsif (step2 > 19) then
                a <= step2 - 20;
                b <= "000010";
        elsif (step2 > 9) then
                a <= step2 - 10;
                b <= "000001";
        else
                a <= step2;
                b <= "000000";
        end if;
    -------
     HEX1 <= to_7seg(a);
     HEX2 <= to_7seg(b);
     HEX3 <= "0001001";
     HEX4 <= "0110111";
     out2 <= step2;
     bcd1 <= to_bcd1(step1);
     if(u = '1') then
        lcd_display_string_02(0) <= x"49";
        else lcd_display_string_02(0) <= x"20";
```

```
end if;
lcd_display_string_02(1)  <= x"46";
lcd_display_string_02(2)  <= x"46";
lcd_display_string_02(3)  <= x"54";
lcd_display_string_02(4)  <= x"3A";
lcd_display_string_02(5)  <= x"79";
lcd_display_string_02(6)  <= x"28";
lcd_display_string_02(7)  <= "0011" & bcd1(7 downto 4);
lcd_display_string_02(8)  <= "0011" & bcd1(3 downto 0);
lcd_display_string_02(9)  <= x"29";
lcd_display_string_02(10) <= x"3D";
lcd_display_string_02(11) <= x"20";
lcd_display_string_02(12) <= x"20";
lcd_display_string_02(13) <= x"20";
lcd_display_string_02(14) <= x"20";
lcd_display_string_02(15) <=  x"20";
if (yr(11) = '1') then
   lcd_display_string_02(16) <= x"2D";
else
   lcd_display_string_02(16) <= x"2B";
end if;
lcd_display_string_02(17) <= "0011" & bcdr(19 downto 16);
lcd_display_string_02(18) <= "0011" & bcdr(15 downto 12);
lcd_display_string_02(19) <= "0011" & bcdr(11 downto 8);
lcd_display_string_02(20) <= "0011" & bcdr(7 downto 4);
lcd_display_string_02(21)<= x"2E";
lcd_display_string_02(22) <= "0011" & bcdr(3 downto 0);
if (yi(11) = '1') then
   lcd_display_string_02(23) <= x"2D";
else
   lcd_display_string_02(23) <= x"2B";
end if;
lcd_display_string_02(24) <= "0011" & bcdi(19 downto 16);
lcd_display_string_02(25) <= "0011" & bcdi(15 downto 12);
lcd_display_string_02(26) <= "0011" & bcdi(11 downto 8);
lcd_display_string_02(27) <= "0011" & bcdi(7 downto 4);
lcd_display_string_02(28)<= x"2E";
lcd_display_string_02(29) <= "0011" & bcdi(3 downto 0);
lcd_display_string_02(30)<= x"69";
```

```
 lcd_display_string_02(31)<= x"20";
end process;
 lcd_display_string_01 <=
  (
-- Line 1 W e l c o m e !
x"57",x"65",x"6C",x"63",x"6F",x"6D",x"65",x"21",x"20",x"20",x"20",x"20",x"20",x"20"
,x"20",x"20",
-- Line 2 T D T U n i v e r s i t y !
x"54",x"44",x"54",x"20",x"55",x"6E",x"69",x"76",x"65",x"72",x"73",x"69",x"74",x"79",
x"21",x"20"
   );
  data_bus <= data_bus_value when lcd_rw_int = '0' else "ZZZZZZZZ";
 lcd_rw <= lcd_rw_int;
---------------------------------- MAY TRANG THAI ----------------------------
process(u1)
begin
if(rising_edge(u1)) then
    if(begin1 = '0' ) then
            begin1 <= begin1 xor '1';
    else
            begin1 <= '0';
    end if;
end if;
    if(begin1 = '0' ) then
            ky_tu_ke <= lcd_display_string_01(CONV_INTEGER(char_count));
    else ky_tu_ke <= lcd_display_string_02(CONV_INTEGER(char_count));
    end if;
end process;
process(clock_50)
begin
    if (rising_edge(clock_50)) then
      if (reset = '0') then
        clk_count_400hz <= x"000000";
        clk_400hz_enable <= '0';
      else
       if (clk_count_400hz <= x"01E848") then
            clk_count_400hz <= clk_count_400hz + 1;
            clk_400hz_enable <= '0';
        else
```

```
                clk_count_400hz <= x"000000";
                clk_400hz_enable <= '1';
            end if;
          end if;
        end if;
    end process;
    process (clock_50, reset)
    begin
        if reset = '0' then
          state <= reset1;
          data_bus_value <= x"38";
          next_command <= reset2;
          lcd_e <= '1';
          lcd_rs <= '0';
          lcd_rw_int <= '0';
        elsif rising_edge(clock_50) then
            if clk_400hz_enable = '1' then
                case state is
                    when reset1 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38";
                        state <= drop_lcd_e;
                        next_command <= reset2;
                        char_count <= "00000";
                    when reset2 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38";
                        state <= drop_lcd_e;
                        next_command <= reset3;
                    when reset3 =>
                        lcd_e <= '1';
                        lcd_rs <= '0';
                        lcd_rw_int <= '0';
                        data_bus_value <= x"38";
                        state <= drop_lcd_e;
```

```
        next_command <= lap_ham;
    when lap_ham =>
        lcd_e <= '1';
        lcd_rs <= '0';
        lcd_rw_int <= '0';
        data_bus_value <= x"38";
        state <= drop_lcd_e;
        next_command <= tat_hienthi;
    when tat_hienthi =>
        lcd_e <= '1';
        lcd_rs <= '0';
        lcd_rw_int <= '0';
        data_bus_value <= x"08";
        state <= drop_lcd_e;
        next_command <= display_clear;
    when display_clear =>
        lcd_e <= '1';
        lcd_rs <= '0';
        lcd_rw_int <= '0';
        data_bus_value <= x"01";
        state <= drop_lcd_e;
        next_command <= mo_hienthi;
    when mo_hienthi =>
        lcd_e <= '1';
        lcd_rs <= '0';
        lcd_rw_int <= '0';
        data_bus_value <= x"0C";
        state <= drop_lcd_e;
        next_command <= mode_set;
    when mode_set =>
        lcd_e <= '1';
        lcd_rs <= '0';
        lcd_rw_int <= '0';
        data_bus_value <= x"06";
        state <= drop_lcd_e;
        next_command <= print_string;
    when print_string =>
        state <= drop_lcd_e;
        lcd_e <= '1';
```

```
                lcd_rs <= '1';
                lcd_rw_int <= '0';
                if (ky_tu_ke(7 downto 4) /= x"0") then
                    data_bus_value <= ky_tu_ke;
                end if;
                if (char_count < "11111") AND (ky_tu_ke /= x"fe") then
                    char_count <= char_count + 1;
                else
                    char_count <= "00000";
                end if;
                if char_count = 15 then
                    next_command <= line2;
                elsif (char_count = 31) or (ky_tu_ke = x"fe") then
                    next_command <= return_home;
                else
                    next_command <= print_string;
                end if;
            when line2 =>
                lcd_e <= '1';
                lcd_rs <= '0';
                lcd_rw_int <= '0';
                data_bus_value <= x"c0";
                state <= drop_lcd_e;
                next_command <= print_string;
            when return_home =>
                lcd_e <= '1';
                lcd_rs <= '0';
                lcd_rw_int <= '0';
                data_bus_value <= x"80";
                state <= drop_lcd_e;
                next_command <= print_string;
            when drop_lcd_e =>
                state <= next_command;
                lcd_e <= '0';
                lcd_blon <= '1';
                lcd_on   <= '1';
             end case;
        end if;
    end if;
```

```
end process;
END a;
```

File Butterfly

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
entity butterfly is
 port( u: in std_logic;
 c1,c2 : in complex; --inputs
 w,iw :in complex; -- phase factor
 g1,g2 :out complex -- outputs
 );
end butterfly;
architecture Behavioral of butterfly is
constant c: complex := ("00000000000010000", "00000000000000000");
signal a,b: complex;
begin
--butterfly equations.
process(u,c1,c2,w,iw,b,a)
begin
if u = '0' then
--FFT.
g1 <= add(c1,c2);
g2 <= mult(sub(c1,c2),w);
else
--IFFT.
b  <= add(c1,c2);
g1 <= mult(b,c);
a  <= mult(sub(c1,c2),iw);
g2 <= mult(a,c);
end if;
end process;
end Behavioral;
```

File đảo bit

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fft_pck.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
entity Dao is
Port ( N: in integer;
     x : in odata_array2;
     y : out odata_array2
);
end Dao;
architecture Behavioral of Dao is
signal z,c : i_array1 := (others => "0000000000");
signal q : i_array := (others => 0);
signal o : integer;
begin
  process(N,x,q)
  begin
    for i in 0 to 63 loop
    if(i < N) then
      z(i) <= div(i);
    c(i) <= convert(z(i),N);
    q(i) <= bin_to_dec1(c(i));
    y(i) <= x(q(i));
      end if;
   end loop;
end process;
end Behavioral;
```

File bảng hệ số nhân

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_SIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
```

```
use work.float_pkg.all;
use Work.fft_pck.all;
entity TWiddle_Factor is
 port( W,iW :out comp_array999
 );
end TWiddle_Factor;
architecture Behavioral of TWiddle_Factor is
signal i,n : integer;
constant M :integer := 511;
begin
process(i,n)
            begin
      for i in 0 to 40 loop
   W(i).r    <= "00000000000100000" ;
   iW(i).r   <= "00000000000100000" ;
   W(M-i).r  <= "11111111111100000" ;
   iW(M-i).r <= "11111111111100000" ;
  end loop;
  for i in 41 to 57 loop
   W(i).r    <= "00000000000011111" ;
   iW(i).r   <= "00000000000011111" ;
   W(M-i).r  <= "11111111111100001" ;
   iW(M-i).r <= "11111111111100001" ;
  end loop;
  for i in 58 to 71 loop
   W(i).r    <= "00000000000011110" ;
   iW(i).r   <= "00000000000011110" ;
   W(M-i).r  <= "11111111111100010" ;
   iW(M-i).r <= "11111111111100010" ;
  end loop;
  for i in 72 to 82 loop
   W(i).r    <= "00000000000011101" ;
   iW(i).r   <= "00000000000011101" ;
   W(M-i).r  <= "11111111111100011" ;
   iW(M-i).r <= "11111111111100011" ;
  end loop;
  for i in 83 to 92 loop
   W(i).r    <= "00000000000011100" ;
   iW(i).r   <= "00000000000011100" ;
```

```
  W(M-i).r  <= "11111111111100100" ;
  iW(M-i).r <= "11111111111100100" ;
end loop;
for i in 93 to 101 loop
  W(i).r    <= "00000000000011011" ;
  iW(i).r   <= "00000000000011011" ;
  W(M-i).r  <= "11111111111100101" ;
  iW(M-i).r <= "11111111111100101" ;
end loop;
for i in 102 to 109 loop
  W(i).r    <= "00000000000011010" ;
  iW(i).r   <= "00000000000011010" ;
  W(M-i).r  <= "11111111111100110" ;
  iW(M-i).r <= "11111111111100110" ;
end loop;
for i in 110 to 117 loop
  W(i).r    <= "00000000000011001" ;
  iW(i).r   <= "00000000000011001" ;
  W(M-i).r  <= "11111111111100111" ;
  iW(M-i).r <= "11111111111100111" ;
end loop;
for i in 118 to 125 loop
  W(i).r    <= "00000000000011000" ;
  iW(i).r   <= "00000000000011000" ;
  W(M-i).r  <= "11111111111101000" ;
  iW(M-i).r <= "11111111111101000" ;
end loop;
for i in 126 to 132 loop
  W(i).r    <= "00000000000010111" ;
  iW(i).r   <= "00000000000010111" ;
  W(M-i).r  <= "11111111111101001" ;
  iW(M-i).r <= "11111111111101001" ;
end loop;
for i in 133 to 139 loop
  W(i).r    <= "00000000000010110" ;
  iW(i).r   <= "00000000000010110" ;
  W(M-i).r  <= "11111111111101010" ;
  iW(M-i).r <= "11111111111101010" ;
end loop;
```

```
for i in 140 to 145 loop
  W(i).r    <= "00000000000010101" ;
  iW(i).r   <= "00000000000010101" ;
  W(M-i).r  <= "11111111111101011" ;
  iW(M-i).r <= "11111111111101011" ;
end loop;
for i in 146 to 152 loop
  W(i).r    <= "00000000000010100" ;
  iW(i).r   <= "00000000000010100" ;
  W(M-i).r  <= "11111111111101100" ;
  iW(M-i).r <= "11111111111101100" ;
end loop;
for i in 153 to 158 loop
  W(i).r    <= "00000000000010011" ;
  iW(i).r   <= "00000000000010011" ;
  W(M-i).r  <= "11111111111101101" ;
  iW(M-i).r <= "11111111111101101" ;
end loop;
for i in 159 to 164 loop
  W(i).r    <= "00000000000010010" ;
  iW(i).r   <= "00000000000010010" ;
  W(M-i).r  <= "11111111111101110" ;
  iW(M-i).r <= "11111111111101110" ;
end loop;
for i in 165 to 170 loop
  W(i).r    <= "00000000000010001" ;
  iW(i).r   <= "00000000000010001" ;
  W(M-i).r  <= "11111111111101111" ;
  iW(M-i).r <= "11111111111101111" ;
end loop;
for i in 171 to 176 loop
  W(i).r    <= "00000000000010000" ;
  iW(i).r   <= "00000000000010000" ;
  W(M-i).r  <= "11111111111110000" ;
  iW(M-i).r <= "11111111111110000" ;
end loop;
for i in 177 to 182 loop
  W(i).r    <= "00000000000001111" ;
  iW(i).r   <= "00000000000001111" ;
```

```
  W(M-i).r  <= "11111111111110001" ;
  iW(M-i).r <= "11111111111110001" ;
end loop;
for i in 183 to 187 loop
  W(i).r    <= "00000000000001110" ;
  iW(i).r   <= "00000000000001110" ;
  W(M-i).r  <= "11111111111110010" ;
  iW(M-i).r <= "11111111111110010" ;
end loop;
for i in 188 to 193 loop
  W(i).r    <= "00000000000001101" ;
  iW(i).r   <= "00000000000001101" ;
  W(M-i).r  <= "11111111111110011" ;
  iW(M-i).r <= "11111111111110011" ;
end loop;
for i in 194 to 198 loop
  W(i).r    <= "00000000000001100" ;
  iW(i).r   <= "00000000000001100" ;
  W(M-i).r  <= "11111111111110100" ;
  iW(M-i).r <= "11111111111110100" ;
end loop;
for i in 199 to 204 loop
  W(i).r    <= "00000000000001011" ;
  iW(i).r   <= "00000000000001011" ;
  W(M-i).r  <= "11111111111110101" ;
  iW(M-i).r <= "11111111111110101" ;
end loop;
for i in 205 to 209 loop
  W(i).r    <= "00000000000001010" ;
  iW(i).r   <= "00000000000001010" ;
  W(M-i).r  <= "11111111111110110" ;
  iW(M-i).r <= "11111111111110110" ;
end loop;
for i in 210 to 214 loop
  W(i).r    <= "00000000000001001" ;
  iW(i).r   <= "00000000000001001" ;
  W(M-i).r  <= "11111111111110111" ;
  iW(M-i).r <= "11111111111110111" ;
end loop;
```

```
for i in 215 to 220 loop
  W(i).r    <= "00000000000001000" ;
  iW(i).r   <= "00000000000001000" ;
  W(M-i).r  <= "11111111111111000" ;
  iW(M-i).r <= "11111111111111000" ;
end loop;
for i in 221 to 225 loop
  W(i).r    <= "00000000000000111" ;
  iW(i).r   <= "00000000000000111" ;
  W(M-i).r  <= "11111111111111001" ;
  iW(M-i).r <= "11111111111111001" ;
end loop;
for i in 226 to 230 loop
  W(i).r    <= "00000000000000110" ;
  iW(i).r   <= "00000000000000110" ;
  W(M-i).r  <= "11111111111111010" ;
  iW(M-i).r <= "11111111111111010" ;
end loop;
for i in 231 to 235 loop
  W(i).r    <= "00000000000000101" ;
  iW(i).r   <= "00000000000000101" ;
  W(M-i).r  <= "11111111111111011" ;
  iW(M-i).r <= "11111111111111011" ;
end loop;
for i in 236 to 240 loop
  W(i).r    <= "00000000000000100" ;
  iW(i).r   <= "00000000000000100" ;
  W(M-i).r  <= "11111111111111100" ;
  iW(M-i).r <= "11111111111111100" ;
end loop;
for i in 241 to 245 loop
  W(i).r    <= "00000000000000011" ;
  iW(i).r   <= "00000000000000011" ;
  W(M-i).r  <= "11111111111111101" ;
  iW(M-i).r <= "11111111111111101" ;
end loop;
for i in 246 to 250 loop
  W(i).r    <= "00000000000000010" ;
  iW(i).r   <= "00000000000000010" ;
```

```
  W(M-i).r  <= "11111111111111110" ;
  iW(M-i).r <= "11111111111111110" ;
end loop;
for i in 251 to 254 loop
  W(i).r    <= "00000000000000001" ;
  iW(i).r   <= "00000000000000001" ;
  W(M-i).r  <= "11111111111111111" ;
  iW(M-i).r <= "11111111111111111" ;
end loop;
for i in 255 to 256 loop
  W(i).r    <= "00000000000000000" ;
  iW(i).r   <= "00000000000000000" ;
end loop;
for n in 0 to 5 loop
    iW(n).i    <= "00000000000000000" ;
  W(n).i    <= "00000000000000000" ;
    W(M-n).i   <= "00000000000000000" ;
  iW(M-n).i <= "00000000000000000" ;
end loop;
for n in 6 to 10 loop
 iW(n).i    <= "00000000000000001" ;
  W(n).i    <= "11111111111111111" ;
    iW(M-n).i  <= "00000000000000001" ;
  W(M-n).i  <= "11111111111111111" ;
end loop;
for n in 11 to 15 loop
    iW(n).i    <= "00000000000000010" ;
  W(n).i    <= "11111111111111110" ;
 iW(M-n).i  <= "00000000000000010" ;
  W(M-n).i  <= "11111111111111110" ;
end loop;
for n in 16 to 20 loop
    iW(n).i    <= "00000000000000011" ;
  W(n).i    <= "11111111111111101" ;
    iW(M-n).i  <= "00000000000000011" ;
  W(M-n).i  <= "11111111111111101" ;
end loop;
for n in 21 to 25 loop
    iW(n).i    <= "00000000000000100" ;
```

```
  W(n).i    <= "11111111111111100" ;
 iW(M-n).i  <= "00000000000000100" ;
  W(M-n).i  <= "11111111111111100" ;
end loop;
for n in 26 to 30 loop
    iW(n).i    <= "00000000000000101" ;
  W(n).i    <= "11111111111111011" ;
    iW(M-n).i  <= "00000000000000101" ;
  W(M-n).i  <= "11111111111111011" ;
end loop;
for n in 31 to 35 loop
    iW(n).i    <= "00000000000000110" ;
  W(n).i    <= "11111111111111010" ;
    iW(M-n).i  <= "00000000000000110" ;
  W(M-n).i  <= "11111111111111010" ;
end loop;
for n in 36 to 41 loop
    iW(n).i    <= "00000000000000111" ;
  W(n).i    <= "11111111111111001" ;
    iW(M-n).i  <= "00000000000000111" ;
  W(M-n).i  <= "11111111111111001" ;
end loop;
for n in 42 to 46 loop
    iW(n).i    <= "00000000000001000" ;
  W(n).i    <= "11111111111111000" ;
    iW(M-n).i  <= "00000000000001000" ;
  W(M-n).i  <= "11111111111111000" ;
end loop;
for n in 47 to 51 loop
    iW(n).i    <= "00000000000001001" ;
  W(n).i    <= "11111111111110111" ;
    iW(M-n).i  <= "00000000000001001" ;
  W(M-n).i  <= "11111111111110111" ;
end loop;
for n in 52 to 57 loop
    iW(n).i    <= "00000000000001010" ;
  W(n).i    <= "11111111111110110" ;
    iW(M-n).i  <= "00000000000001010" ;
  W(M-n).i  <= "11111111111110110" ;
```

```
end loop;
for n in 58 to 62 loop
    iW(n).i    <= "0000000000001011" ;
  W(n).i    <= "1111111111110101" ;
    iW(M-n).i  <= "0000000000001011" ;
  W(M-n).i  <= "1111111111110101" ;
end loop;
for n in 63 to 68 loop
    iW(n).i    <= "0000000000001100" ;
  W(n).i    <= "1111111111110100" ;
    iW(M-n).i  <= "0000000000001100" ;
  W(M-n).i  <= "1111111111110100" ;
end loop;
for n in 69 to 73 loop
    iW(n).i    <= "0000000000001101" ;
  W(n).i    <= "1111111111110011" ;
    iW(M-n).i  <= "0000000000001101" ;
  W(M-n).i  <= "1111111111110011" ;
end loop;
for n in 74 to 79 loop
    iW(n).i    <= "0000000000001110" ;
  W(n).i    <= "1111111111110010" ;
    iW(M-n).i  <= "0000000000001110" ;
  W(M-n).i  <= "1111111111110010" ;
end loop;
for n in 80 to 85 loop
    iW(n).i    <= "0000000000001111" ;
  W(n).i    <= "1111111111110001" ;
    iW(M-n).i  <= "0000000000001111" ;
  W(M-n).i  <= "1111111111110001" ;
end loop;
for n in 86 to 91 loop
    iW(n).i    <= "0000000000010000" ;
  W(n).i    <= "1111111111110000" ;
    iW(M-n).i  <= "0000000000010000" ;
  W(M-n).i  <= "1111111111110000" ;
end loop;
for n in 92 to 97 loop
    iW(n).i    <= "0000000000010001" ;
```

```
  W(n).i    <= "1111111111101111" ;
    iW(M-n).i  <= "0000000000010001" ;
  W(M-n).i  <= "1111111111101111" ;
end loop;
for n in 98 to 103 loop
    iW(n).i    <= "0000000000010010" ;
  W(n).i    <= "1111111111101110" ;
    iW(M-n).i  <= "0000000000010010" ;
  W(M-n).i  <= "1111111111101110" ;
end loop;
for n in 104 to 110 loop
    iW(n).i    <= "0000000000010011" ;
  W(n).i    <= "1111111111101101" ;
    iW(M-n).i  <= "0000000000010011" ;
  W(M-n).i  <= "1111111111101101" ;
end loop;
for n in 111 to 116 loop
    iW(n).i    <= "0000000000010100" ;
  W(n).i    <= "1111111111101100" ;
    iW(M-n).i  <= "0000000000010100" ;
  W(M-n).i  <= "1111111111101100" ;
end loop;
for n in 117 to 123 loop
    iW(n).i    <= "0000000000010101" ;
  W(n).i    <= "1111111111101011" ;
    iW(M-n).i  <= "0000000000010101" ;
  W(M-n).i  <= "1111111111101011" ;
end loop;
for n in 124 to 130 loop
    iW(n).i    <= "0000000000010110" ;
  W(n).i    <= "1111111111101010" ;
    iW(M-n).i  <= "0000000000010110" ;
  W(M-n).i  <= "1111111111101010" ;
end loop;
for n in 131 to 138 loop
    iW(n).i    <= "0000000000010111" ;
  W(n).i    <= "1111111111101001" ;
    iW(M-n).i  <= "0000000000010111" ;
  W(M-n).i  <= "1111111111101001" ;
```

```
end loop;
for n in 139 to 146 loop
    iW(n).i    <= "00000000000011000" ;
  W(n).i    <= "11111111111101000" ;
    iW(M-n).i  <= "00000000000011000" ;
  W(M-n).i  <= "11111111111101000" ;
end loop;
for n in 147 to 154 loop
    iW(n).i    <= "00000000000011001" ;
  W(n).i    <= "11111111111100111" ;
    iW(M-n).i  <= "00000000000011001" ;
  W(M-n).i  <= "11111111111100111" ;
end loop;
for n in 155 to 163 loop
    iW(n).i    <= "00000000000011010" ;
  W(n).i    <= "11111111111100110" ;
    iW(M-n).i  <= "00000000000011010" ;
  W(M-n).i  <= "11111111111100110" ;
end loop;
for n in 164 to 173 loop
    iW(n).i    <= "00000000000011011" ;
  W(n).i    <= "11111111111100101" ;
    iW(M-n).i  <= "00000000000011011" ;
  W(M-n).i  <= "11111111111100101" ;
end loop;
for n in 174 to 184 loop
    iW(n).i    <= "00000000000011100" ;
  W(n).i    <= "11111111111100100" ;
    iW(M-n).i  <= "00000000000011100" ;
  W(M-n).i  <= "11111111111100100" ;
end loop;
for n in 185 to 198 loop
    iW(n).i    <= "00000000000011101" ;
  W(n).i    <= "11111111111100011" ;
    iW(M-n).i  <= "00000000000011101" ;
  W(M-n).i  <= "11111111111100011" ;
end loop;
for n in 199 to 215 loop
    iW(n).i    <= "00000000000011110" ;
```

```
   W(n).i    <= "11111111111100010" ;
     iW(M-n).i  <= "00000000000011110" ;
   W(M-n).i  <= "11111111111100010" ;
  end loop;
  for n in 216 to 254 loop
     iW(n).i    <= "00000000000011111" ;
   W(n).i    <= "11111111111100001" ;
     iW(M-n).i  <= "00000000000011111" ;
   W(M-n).i  <= "11111111111100001" ;
  end loop;
  for n in 255 to 256 loop
     iW(n).i    <= "00000000000100000" ;
   W(n).i    <= "11111111111100000" ;
  end loop;
end process;
end Behavioral;
```

File fft_package

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
use ieee.math_real.all;
use work.fixed_float_types.all;
use work.fixed_pkg.all;
use work.float_pkg.all;
package fft_pck is
type complex is
 record
 r : sfixed(11 downto -5);
 i : sfixed(11 downto -5);
 end record;
 type realcomplex is
 record
 r : real;
 i : real;
 end record;
```

```
type idata_array is array (0 to 63) of sfixed(11 downto -5);
type comp_array is array (0 to 7) of complex;
type comp_array1 is array (0 to 15) of complex;
type comp_array999 is array (0 to 511) of complex;
type comp_array2 is array (0 to 31) of complex;
type odata_array is array (0 to 15) of complex;
type odata_array1 is array (0 to 31) of complex;
type odata_array2 is array (0 to 63) of complex;
type i_array1 is array (0 to 63) of std_logic_vector(9 downto 0);
type i_array is array (0 to 63) of integer;
function add (n1,n2 : complex) return complex;
function sub (n1,n2 : complex) return complex;
function mult (n1,n2 : complex) return complex;
function div (n1 : integer) return std_logic_vector;
function bin_to_dec (n1 : sfixed(11 downto -5)) return real;
function bin_to_dec1 (n1 : std_logic_vector(9 downto 0)) return integer;
function convert (n1 : std_logic_vector(9 downto 0); n2 : integer) return
std_logic_vector;
function to_bcd ( bin : sfixed(11 downto -5) ) return std_logic_vector;
function to_bcd1 ( bin : std_logic_vector(5 downto 0) ) return std_logic_vector;
end fft_pck;
package body fft_pck is
function add (n1,n2 : complex) return complex is
variable sum : complex;
variable sum_r,sum_i : sfixed(12 downto -5);
begin
sum_r :=n1.r + n2.r;
sum_i :=n1.i + n2.i;
sum.r := sum_r(11 downto -5);
sum.i := sum_i(11 downto -5);
return sum;
end add;
function sub (n1,n2 : complex) return complex is
variable diff : complex ;
variable diff_r,diff_i : sfixed(12 downto -5) ;
begin
diff_r:=n1.r - n2.r;
diff_i:=n1.i - n2.i;
diff.r := diff_r(11 downto -5);
```

```
diff.i := diff_i(11 downto -5);
return diff;
end sub;
--multiplication of complex numbers.
function mult (n1,n2 : complex) return complex is
variable prod : complex;
variable prod_aux_r : sfixed(24 downto -10);
variable prod_aux_i : sfixed(24 downto -10);
begin
prod_aux_r := (n1.r * n2.r) - (n1.i * n2.i);
prod.r:= prod_aux_r(11 downto -5);
prod_aux_i := (n1.r * n2.i) + (n1.i * n2.r);
prod.i:= prod_aux_i(11 downto -5);
return prod;
end mult;
function div (n1 : integer) return std_logic_vector is
variable n : integer;
variable div : std_logic_vector(9 downto 0);
variable o : integer ;
begin
    n := n1;
    if n > 0 then
            for o in 9 downto 0 loop
            if (n >= 2**(o)) then div(o) := '1'; n := n - 2**(o);
            else div(o) := '0';
            end if;
            end loop;
    end if;
return div;
end div;
function bin_to_dec (n1 : sfixed(11 downto -5)) return real is
variable bin : sfixed(11 downto -5);
variable dec : real ;
begin
    bin := n1;
    dec := 0.0;
    if bin(11) = '1' then dec := dec - 2048.0;
    else dec := dec;
    end if;
```

```
if bin(10) = '1' then dec := dec + 1024.0;
else dec := dec;
end if;
if bin(9) = '1' then dec := dec + 512.0;
else dec := dec;
end if;
if bin(8) = '1' then dec := dec + 256.0;
else dec := dec;
end if;
if bin(7) = '1' then dec := dec + 128.0;
else dec := dec;
end if;
if bin(6) = '1' then dec := dec + 64.0;
else dec := dec;
end if;
if bin(5) = '1' then dec := dec + 32.0;
else dec := dec;
end if;
if bin(4) = '1' then dec := dec + 16.0;
else dec := dec;
end if;
if bin(3) = '1' then dec := dec + 8.0;
else dec := dec;
end if;
if bin(2) = '1' then dec := dec + 4.0;
else dec := dec;
end if;
if bin(1) = '1' then dec := dec + 2.0;
else dec := dec;
end if;
if bin(0) = '1' then dec := dec + 1.0;
else dec := dec;
end if;
if bin(-1) = '1' then dec := dec + 0.5;
else dec := dec;
end if;
if bin(-2) = '1' then dec := dec + 0.25;
else dec := dec;
end if;
```

```
    if bin(-3) = '1' then dec := dec + 0.125;
    else dec := dec;
    end if;
    if bin(-4) = '1' then dec := dec + 0.0625;
    else dec := dec;
    end if;
    if bin(-5) = '1' then dec := dec + 0.03125;
    else dec := dec;
    end if;
return dec;
end bin_to_dec;
function bin_to_dec1 (n1 : std_logic_vector(9 downto 0)) return integer is
variable bin : std_logic_vector(9 downto 0);
variable dec : integer;
begin
    bin := n1;
    dec := 0;
    if bin(9) = '1' then dec := dec + 512;
    else dec := dec;
    end if;
    if bin(8) = '1' then dec := dec + 256;
    else dec := dec;
    end if;
    if bin(7) = '1' then dec := dec + 128;
    else dec := dec;
    end if;
    if bin(6) = '1' then dec := dec + 64;
    else dec := dec;
    end if;
    if bin(5) = '1' then dec := dec + 32;
    else dec := dec;
    end if;
    if bin(4) = '1' then dec := dec + 16;
    else dec := dec;
    end if;
    if bin(3) = '1' then dec := dec + 8;
    else dec := dec;
    end if;
    if bin(2) = '1' then dec := dec + 4;
```

```
        else dec := dec;
        end if;
        if bin(1) = '1' then dec := dec + 2;
        else dec := dec;
        end if;
        if bin(0) = '1' then dec := dec + 1;
        else dec := dec;
        end if;
    return dec;
    end bin_to_dec1;
    function convert (n1 : std_logic_vector(9 downto 0);n2 : integer) return std_logic_vector is
    variable a : std_logic_vector(9 downto 0);
    variable b : std_logic_vector(9 downto 0);
    variable c : integer;
    begin
      a := n1;
      c := n2;
      if(c <= 2) then
      for i in 0 to 0 loop
      b(i) := a(i);
      end loop;
      elsif(c <= 4) then
      for i in 0 to 1 loop
      b(i) := a(1-i);
      end loop;
      elsif (c <= 8) then
      for i in 0 to 2 loop
        b(i) := a(2-i);
      end loop;
      elsif (c <= 16) then
      for i in 0 to 3 loop
        b(i) := a(3-i);
      end loop;
      elsif (c <= 32) then
      for i in 0 to 4 loop
        b(i) := a(4-i);
      end loop;
      elsif (c <= 64) then
```

```
  for i in 0 to 5 loop
    b(i) := a(5-i);
  end loop;
  elsif (c <= 128) then
  for i in 0 to 6 loop
    b(i) := a(6-i);
  end loop;
  elsif (c <= 256) then
  for i in 0 to 7 loop
    b(i) := a(7-i);
  end loop;
  elsif (c <= 512) then
  for i in 0 to 8 loop
    b(i) := a(8-i);
  end loop;
  elsif (c <= 1023) then
  for i in 0 to 9 loop
    b(i) := a(9-i);
  end loop;
  end if;
return b;
end convert;
function to_bcd ( bin : sfixed(11 downto -5) ) return std_logic_vector is
variable i,o : integer:=0;
variable bcd : std_logic_vector(19 downto 0) := "00000000000000000000";
variable bint : sfixed(11 downto -5) := bin;
variable bintt,binttt : sfixed(11 downto -5);
variable bint1,bint11 : sfixed(23 downto -10);
variable bin1 : std_logic_vector(11 downto 0);
constant bint2 : sfixed(11 downto -5) := "11111111111100000";
constant bint3 : sfixed(11 downto -5) := "00000000101000000";
begin
if bint(11) = '1' then
     bint1 := bint2 * bint;
     bintt := bint1(11 downto -5);
     else bintt := bint;
end if;
     bint11 := bintt * bint3;
     binttt := bint11(11 downto -5);
```

```
    for o in 0 to 11 loop
    bin1(o) := binttt(o);
    end loop;
for i in 0 to 11 loop  -- repeating 12 times.
bcd(19 downto 1) := bcd(18 downto 0);  --shifting the bits.
bcd(0) := bin1(11);
bin1(11 downto 1) := bin1(10 downto 0);
bin1(0) :='0';

if(i < 11 and bcd(3 downto 0) > "0100") then --add 3 if BCD digit is greater than 4.
bcd(3 downto 0) := bcd(3 downto 0) + "0011";
end if;

if(i < 11 and bcd(7 downto 4) > "0100") then --add 3 if BCD digit is greater than 4.
bcd(7 downto 4) := bcd(7 downto 4) + "0011";
end if;

if(i < 11 and bcd(11 downto 8) > "0100") then  --add 3 if BCD digit is greater than 4.
bcd(11 downto 8) := bcd(11 downto 8) + "0011";
end if;
if(i < 11 and bcd(15 downto 12) > "0100") then  --add 3 if BCD digit is greater than 4.
bcd(15 downto 12) := bcd(15 downto 12) + "0011";
end if;
if(i < 11 and bcd(19 downto 16) > "0100") then  --add 3 if BCD digit is greater than 4.
bcd(19 downto 16) := bcd(19 downto 16) + "0011";
end if;
end loop;
return bcd;
end to_bcd;
function to_bcd1 ( bin : std_logic_vector(5 downto 0) ) return std_logic_vector is
variable i : integer:=0;
variable bcd : std_logic_vector(7 downto 0) := (others => '0');
variable bint : std_logic_vector(5 downto 0) := bin;
begin
for i in 0 to 5 loop  -- repeating 6 times.
bcd(7 downto 1) := bcd(6 downto 0);  --shifting the bits.
bcd(0) := bint(5);
bint(5 downto 1) := bint(4 downto 0);
bint(0) :='0';
```

```
if(i < 5 and bcd(3 downto 0) > "0100") then --add 3 if BCD digit is greater than 4.
bcd(3 downto 0) := bcd(3 downto 0) + "0011";
end if;

if(i < 5 and bcd(7 downto 4) > "0100") then --add 3 if BCD digit is greater than 4.
bcd(7 downto 4) := bcd(7 downto 4) + "0011";
end if;
end loop;
return bcd;
end to_bcd1;
end fft_pck;
```