CHAPTER 2

LITERATURE REVIEW

As this music box project involved FPGA, Verilog HDL language, and Altera Education Kit (UP2 Board), information on the basic of the above mentioned has to be studied.

2.1 Introduction of FPGA

Before the advent of programmable logic, custom logic circuits were built at the board level using standard components, or at the gate level in expensive application-specific (custom) integrated circuits. The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities. The individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit [9][14].

2.1.1 Function of a logic cell

The logic cell architecture varies between different device families. Generally speaking, each logic cell combines a few binary inputs (typically between 3 and 10) to one or two outputs according to a boolean logic function specified in the user program. In most families, the user also has the option of registering the combinatorial output of the cell, so that clocked logic can be easily implemented. The cell's combinatorial logic may be physically implemented as a small look-up table memory (LUT) or as a set of multiplexers and gates. LUT devices tend to be a bit more flexible and provide more inputs per cell than multiplexer cells at the expense of propagation delay [9][14].

2.1.2 Field Programmable

Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. A typical integrated circuit performs a particular function defined at the time of manufacture. In contrast, the FPGA's function is defined by a program written by someone other than the device manufacturer. Depending on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up. This user programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits [9][14].

2.1.3 Creation of FPGA Programs

Individually defining the many switch connections and cell logic functions would be a daunting task. Fortunately, this task is handled by special software. The software translates a user's schematic diagrams or textual hardware description language code then places and routes the translated design. Most of the software packages have hooks to allow the user to influence implementation, placement and routing to obtain better performance and utilization of the device. Libraries of more complex function macros (eg. adders) further simplify the design process by providing common circuits that are already optimized for speed or area [9][14].

2.2 Introduction of UP2 Education Kit

The University Program UP2 Education Kit was designed to meet the needs of universities teaching digital logic design with state-of-the-art development tools and programmable logic devices (PLDs). The package provides all of the necessary tools for creating and implementing digital logic designs, including the following features:

- Quartus® II Web-Edition development software
- UP2 Education Board
  – An EPF10K70 device in a 240-pin power quad flat pack (RQFP) package
  – An EPM7128S device in an 84-pin plastic J-lead chip carrier (PLCC) package
- ByteBlasterTM II parallel ports download cable

2.2.1 UP2 Education Board

The UP2 Education Board is a stand-alone experiment board based on a FLEX® 10K device and includes a MAX® 7000 device. When used with the Quartus II software, the board provides a superior platform for learning digital logic design using industry-standard development tools and PLDs.

The board is designed to meet the needs of instructors and students in a laboratory environment. The UP2 Education Board supports both look-up table (LUT) -based and product term-based architectures. The EPF10K70 device can be configured in-system with either the ByteBlaster II download cable or an EPC1 configuration device. Additional download cables can be purchased separately. The EPM7128S device can be programmed in-system with the ByteBlaster II download cable [2][3].

2.2.1.1 EPF10K70 Device

The EPF10K70 device is based on SRAM technology. It is available in a 240-pin RQFP package and has 3,744 logic elements (LEs) and nine embedded array blocks (EABs). Each LE consists of a four-input LUT, a programmable flipflop, and dedicated signal paths for carry-and-cascade functions. Each EAB provides 2,048 bits of memory which can be used to create RAM, ROM, or first-in first-out (FIFO) functions. EABs can also implement logic functions, such as multipliers, microcontrollers, state machines, and digital signal processing (DSP) functions. With 70,000 typical gates, the EPF10K70 device is ideal for intermediate to advanced digital design courses, including computer architecture, communications, and DSP applications [2][3].

2.2.1.2 EPM7128S Device

The EPM7128S device, a member of the high-density, high-performance MAX 7000S family, is based on erasable programmable read-only memory (EEPROM) elements. The EPM7128S device features a socket mounted 84-pin plastic j-lead chip carrier (PLCC) package and has 128 macro-cells. Each macro-cell has a programmable-AND/fixed-OR array as well as a configurable register with independently-programmable clock, clock enable, clear, and preset functions. With a capacity of 2,500 gates and a simple architecture, the EPM7128S device is ideal for introductory designs as well as larger combinatorial and sequential logic functions [2][3].

2.2.1.3 ByteBlaster II Parallel Port Download Cable

Designs can be easily and quickly downloaded into the UP2 Education Board using the ByteBlaster II download cable, which is a hardware interface to a standard parallel port. This cable sends programming or configuration data between the Quartus II software and the UP2 Education Board. Because design changes are downloaded directly to the devices on the board, prototyping is easy and multiple design iterations can be accomplished in quick succession [2][3].

2.2.2 UP2 Education Board Description

The UP2 Education Board, shown in Figure 2.0, contains the features described in this section

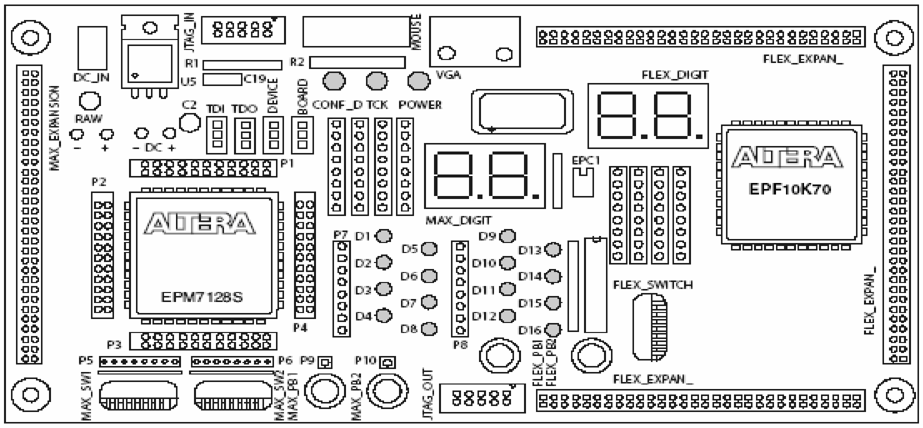


Figure 2.0: UP2 Educational Board Block Diagram [3].

2.2.2.1 DC_IN & RAW Power Input

The DC_IN power input accepts a 2.5-mm × 5.55-mm female connector. The acceptable DC input is 7 to 9 V at a minimum of 350 mA. The RAW power input consists of two holes for connecting an unregulated power source. The hole marked with a plus sign (+) is the positive input; the hole marked with a minus sign (–) is board-common [2][3].

2.2.2.2 Oscillator

The UP2 Education Board contains a 25.175-MHz crystal oscillator. The output of the oscillator drives a global clock input on the EPM7128S device (pin 83) and a global clock input on the FLEX 10K device (pin 91) [2][3].

2.2.2.3 JTAG_IN Header

The 10-pin female plug on the ByteBlaster II download cable connects with the JTAG_IN 10-pin male header on the UP2 Education Board. The board provides power and ground to the ByteBlaster II download cable. Data is shifted into the devices via the TDI pin and shifted out of the devices via the TDO pin. Table 1 identifies the JTAG_IN pin names when the ByteBlaster II is operating in Joint Test Action Group (JTAG) mode [2][3].

Table 2.0: JTAG_IN 10-Pin Header Pin-Outs [3].

Pin	JTAG Signal
1	TCK
2	GND
3	TDO
4	VCC
5	TMS
6	No Connect
7	No Connect
8	No Connect
9	TDI
10	GND

2.2.2.4 Jumpers

The UP2 Education Board has four three-pin jumpers (TDI, TDO, DEVICE, and BOARD) that set the JTAG configuration. The JTAG chain can be set for a variety of configurations (i.e., to program only the EPM7128S device, to configure only the FLEX 10K device, to configure and program both devices, or to connect multiple UP2 Education Boards together). Figure 2.1 shows the positions of the three connectors (C1, C2, and C3) on each of the four jumpers. Table 2.1 defines the settings for each configuration [2][3].

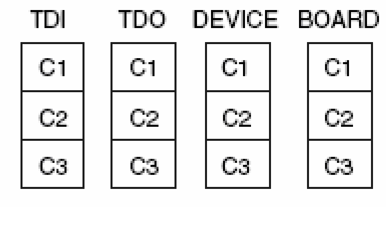
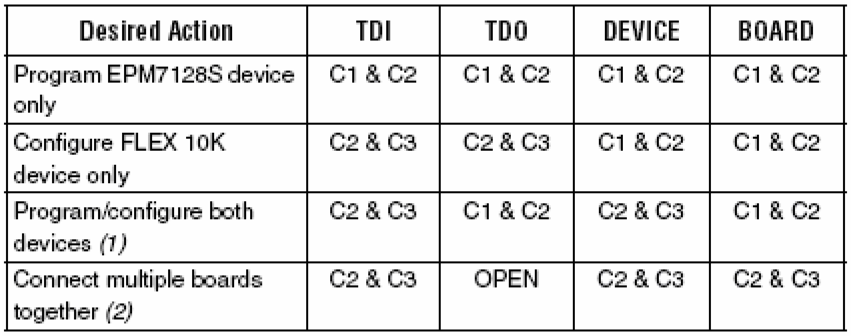


Figure 2.1: Position of C1, C2 & C3 Connectors [3].

Table 2.1: JTAG Jumper Settings [3].



Desired Action	TDI	TDO	DEVICE	BOARD
Program EPM7128S device only	C1 & C2	C1 & C2	C1 & C2	C1 & C2
Configure FLEX 10K device only	C2 & C3	C2 & C3	C1 & C2	C1 & C2
Program/configure both devices *(1)*	C2 & C3	C1 & C2	C2 & C3	C1 & C2
Connect multiple boards together *(2)*	C2 & C3	OPEN	C2 & C3	C2 & C3

Notes to Table 2.1:

(1) The first device in the JTAG chain is the FLEX 10K device, and the second device is the EPM7128S device.

(2) The first device in the JTAG chain is the FLEX 10K device, and the second device is the EPM7128S device. The last board in the chain must be set for a single board configuration (i.e., for programming only the EPM7128S device, configuring only the FLEX 10K device, or configuring/programming both devices). The last board cannot be set for connecting multiple boards together. During configuration, the green CONF_D LED will turn off and the green TCK LED will modulate to indicate that data is transferring. After the device has successfully configured, the CONF_D LED will illuminate [2][3].

2.2.2.5 EPM7128S Device

The UP2 Education Board provides the following resources for the EPM7128S device.

- Socket-mounted 84-pin PLCC package
- Signal pins that are accessible via female headers

- ■ JTAG chain connection for the ByteBlaster II cable
- ■ Two momentary push-button switches
- ■ Two octal dual inline package (DIP) switches
- ■ 16 LEDs
- ■ Dual-digit seven-segment display
- ■ On-board oscillator (25.175 MHz)
- ■ Expansion port with 42 I/O pins and the dedicated global CLR, OE1, and 2/GCLK2 pins

Pins from the EPM7128S device are not pre-assigned to switches and LEDs, but are instead connected to female headers. With direct access to the pins, students can oncentrate on design fundamentals and learn about the programmability of I/O pins and PLDs. After successfully compiling and verifying a design with the Quartus II software, students can easily connect the assigned I/O pins to the switches and LEDs using a common hook-up wire. Students can then download their design into the device and compare their design's simulation to the actual hardware implementation [2][3].

2.2.2.6 EPM7128S Prototyping Headers

The EPM7128S prototyping headers are female headers that surround the device and provide access to the device's signal pins. The 21 pins on each side of the 84-pin PLCC package connect to one of the 22-pin, dual-row 0.1-inch female headers. The pin numbers for the EPM7128S device are printed on the UP2 Education Board (an "X" indicates an unassigned pin). Table 2.2 lists the pin numbers for the four female headers: P1, P2, P3, and P4. The power, ground, and JTAG signal pins are not accessible through these female headers [2][3].

Table 2.2: Pin umbers for Each Prototyping Header [3].

P1		P2		P3		P4	
Outside	Inside	Outside	Inside	Outside	Inside	Outside	Inside
75	76	12	13	33	34	54	55
77	78	14	15	35	36	56	57
79	80	16	17	37	38	58	59
81	82	18	19	39	40	60	61
83	84	20	21	41	42	62	63
1	2	22	23	43	44	64	65
3	4	24	25	45	46	66	67
5	6	26	27	47	48	68	69
7	8	28	29	49	50	70	71
9	10	30	31	51	52	72	73
11	X	32	X	53	X	74	X

2.2.2.7 MAX_PB1 & MAX_PB2 Push-Buttons

MAX_PB1 and MAX_PB2 are two push-buttons that provide active-low signals and are pulled-up through 10-KΩ resistors. Connections to these signals are easily made by inserting one end of the hook-up wire into the push-button female header. The other end of the hook-up wire should be inserted into the appropriate female header assigned to the I/O pin of the EPM7128S device [2][3].

2.2.2.8 MAX_SW1 & MAX_SW2 Switches

MAX_SW1 and MAX_SW2 each contain eight switches that provide logic level signals. These switches are pulled-up through 10-KΩ resistors. Connections to these signals are made by inserting one end of the hookup wire into the female header aligned with the appropriate switch. Insert the other end of the hook-up wire into the appropriate female header assigned to the I/O pin of the EPM7128S device. The switch output is set to logic 1 when the switch is open and set to logic 0 when the switch is closed [2][3].

2.2.2.9 FLEX10K Device

The UP2 Education Board provides the following resources for the FLEX 10K device. The pins from the FLEX 10K device are pre-assigned to switches and LEDs on the board [2][3].

- JTAG chain connection for the ByteBlaster II cable
- Socket for an EPC1 configuration device
- Two momentary push button switches
- One octal DIP switch
- Dual-digit seven-segment display
- On-board oscillator (25.175 MHz)
- VGA port
- Mouse port
- Three expansion ports, each with 42 I/O pins and seven global pins

2.2.2.10 FLEX_PB1 & FLEX_PB2 Push Buttons

FLEX_PB1 and FLEX_PB2 are two push buttons that provide active-low signals to two general-purpose I/O pins on the FLEX 10K device. FLEX_PB1 connects to pin 28, and FLEX_PB2 connects to pin 29. Each push button is pulled-up through a 10-KΩ resistor [2][3].

2.2.2.11 FLEX_SW1 Switches

FLEX_SW1 contains eight switches that provide logic-level signals to eight general-purpose I/O pins on the FLEX 10K device. An input pin is set to logic 1 when the switch is open and set to logic 0 when the switch is closed. Table 2.3 lists the pin assignment for each switch [2][3].

Table 2.3: FLEX_SW1 Pin Assignments [3].

Switch	FLEX 10K Pin
FLEX_SWITCH-1	41
FLEX_SWITCH-2	40
FLEX_SWITCH-3	39
FLEX_SWITCH-4	38
FLEX_SWITCH-5	36
FLEX_SWITCH-6	35
FLEX_SWITCH-7	34
FLEX_SWITCH-8	33

2.2.2.12 FLEX_EXPAN_A, FLEX_EXPAN_B & FLEX_EXPAN_C

FLEX_EXPAN_A, FLEX_EXPAN_B, and FLEX_EXPAN_C are dual rows of 0.1-inch spaced holes for accessing signal I/O pins and global signals on the FLEX 10K device, power, and ground. Figure 6 shows the numbering convention for these holes [2][3].

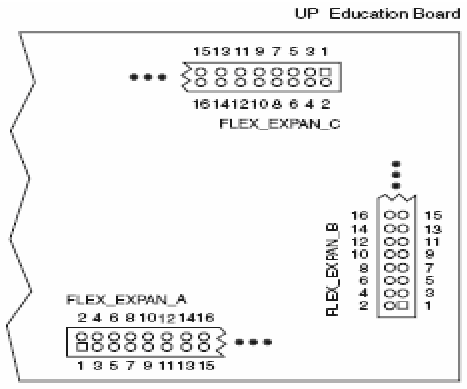


Figure 2.2: FLEX_EXPAN_A, FLEX_EXPAN_B & FLEX_EXPAN_C Numbering Convention [3].

2.3 Introduction of Verilog HDL

Verilog HDL is a description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic-level to the switch-level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete electronic digital system, or anything in between. The digital system can be described hierarchically and timing can be explicitly modeled within the same description.

The Verilog HDL language includes capabilities to describe the behavioral nature of a design, the dataflow nature of a design, a design's structural composition, delays and a waveform generation mechanism including aspects of response monitoring and verification, all modeled using one single language. In addition, the language provides a programming language interface through which the internals of a design can be accessed during simulation including the control of a simulation run.

The language not only defines the syntax but also defines very clear simulation semantics for each language construct. Therefore, models written in this language can be verified using a Verilog simulation. The language inherits many of its operator symbols and constructs from the C programming language. Verilog HDL provide an extensive range of modeling capabilities, some of which are quite difficult to comprehend initially. However, a core subset of the language is quite easy to learn and use. This is sufficient to model most applications. The complete language, however, has sufficient capabilities to capture the descriptions from the most complex chips to a complete electronic system [4][10][11][12][13][15][16].

2.3.1 Major Capabilities

Listed below are the major capabilities of the Verilog hardware description language:

- Primitive logic gates, such as *and*, *or* and *nand*, are built-in into the language.
- Flexibility of creating a user-defined primitive (UDP). Such a primitive could either be a combinational logic primitive or a sequential logic primitive.
- Switch-level modeling primitive gates, such as *pmos* and *nmos*, are also built-in into the language.
- Explicit language constructs are provided for specifying pin-to-pin delays, path delays and timing checks of a design.
- A design can be modeled in three different styles or in a mixed style. These styles are: behavioral style - modeled using procedural constructs; dataflow style – modeled using continuous assignments; and structural style – modeled using gate and module instantiations.
- There are two data types in Verilog HDL; the net data type and the register data type. The net type represents a physical connection between structural element while a register type represents an abstract data storage element.
- Hierarchical design can be described, up to any level, using the module instantiation construct.
- A design can be of arbitrary size; the language does not impose a limit.
- Verilog HDL is non-proprietary and is an IEEE standard.

- It is human and machine readable. Thus it can be used as an exchange language between tools and designers.
- The capabilities of the Verilog HDL language can be further extended by using the programming language interface (PLI) mechanism. PLI is a collection of routines that allow foreign functions to access information within a Verilog module and allows for designer interaction with the simulator.
- A design can be described in a wide range of levels, raging from switch-level, gate-level, register-transfer-level (RTL) to algorithmic-level, including process and queuing-level.
- A design can be modeled entirely at the switch-level using the built-in switch-level primitives.
- The same single language can be used to generate stimulus for the design and for specifying test constraints, such as specifying the values of inputs.
- Verilog HDL can be used to perform response monitoring of the design under test, that values of a design under test can be monitored and displayed. These values can also be complete with expected values, and in case of a mismatch, a report message can be printed.
- At the behavioral-level, Verilog HDL can be used to describe a design not only at the RTL-level, but also at the architectural-level and its algorithmic-level behavior.
- At the structure-level, gate and module instantiations can be used [4][10][11][12][13][15][16].

2.4 Introduction to Quartus II

The Altera® Quartus® II design software provides a complete, multiplatform design environment that easily adapts to user specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes solutions for all phases of FPGA and CPLD design (Figure 2.3).

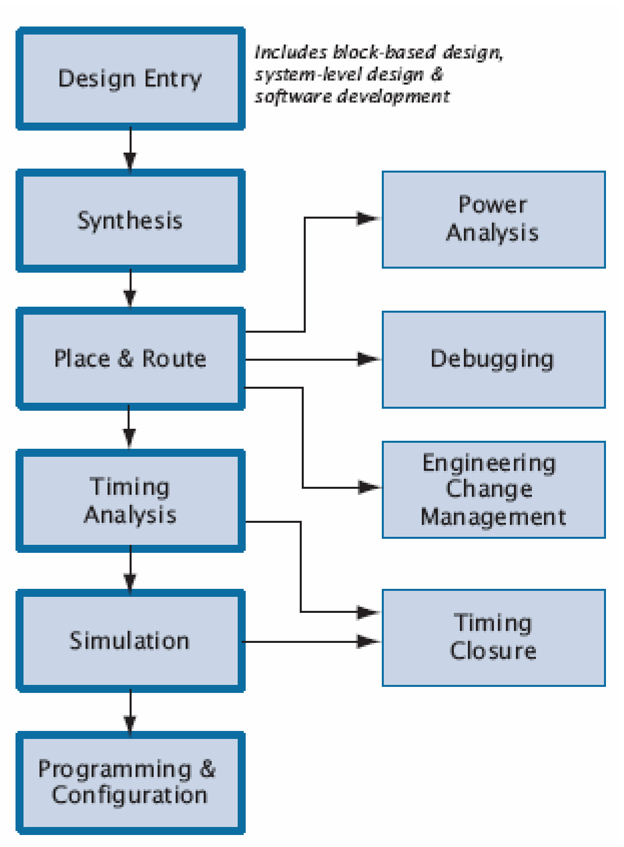

Design Entry
Includes block-based design, system-level design & software development
Synthesis
Place & Route
Timing Analysis
Simulation
Programming & Configuration
Power Analysis
Debugging
Engineering Change Management
Timing Closure

Figure 2.3: Quartus II Design Flow [1].

In addition, the Quartus II software allows all user use the Quartus II graphical user interface and command-line interface for each phase of the design flow. User can use one of these interfaces for the entire flow, or use different options at different phases [1].

2.4.1 Graphical User Interface Design Flow

Quartus II software can perform all stages of the design flow; it is a complete, easy-to-use, stand-alone solution. Figure 2.4 shows the Quartus II graphical user interface as it appears when you first start the software [1].

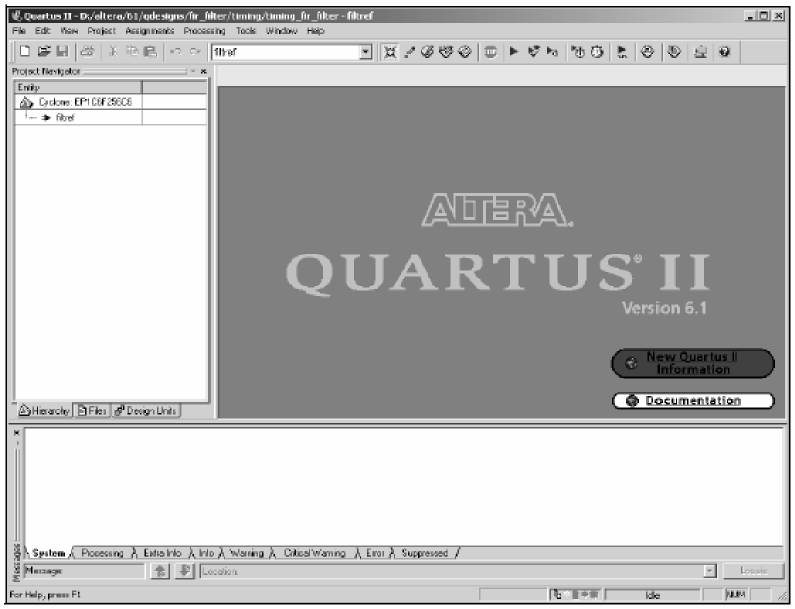


Figure 2.4 Quartus II Graphical User Interface [1].

The Quartus II software includes a modular Compiler. The Compiler includes the following modules (modules marked with an asterisk are optional during a full compilation, depending on user settings):

■ Analysis & Synthesis
■ Partition Merge*
■ Fitter
■ Assembler*
■ Classic Timing Analyzer and TimeQuest Timing Analyzer*
■ Design Assistant*
■ EDA Netlist Writer*
■ HardCopy® Netlist Writer*

To run all Compiler modules as part of a full compilation, on the processing menu click Start Compilation. User also can run each module individually by pointing to start of the Processing menu then clicking the command for the module that user want to start. User also can run some of the Compiler modules incrementally. In addition, to start the Compiler modules individually, click Compiler Tool on the Tools menu and run each module from the Compiler Tool window (Figure 2.5). The Compiler Tool window also allows user to open the settings file or report file for the module, or to open other related windows [1].

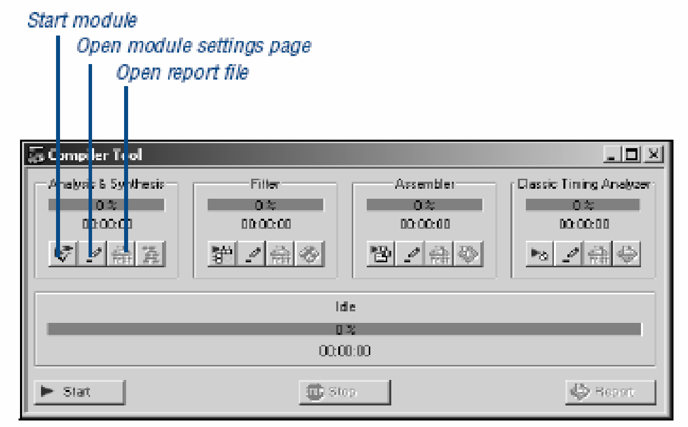


Figure 2.5: Compiler Tool Window [1].

The Quartus II software also provides predefined compilation flows that user can use with commands from the Processing menu. Table 2.3 lists the commands for some of the most common compilation flows [1].

Table 2.4: Commands for Common Compiler Flows [1].

Flow	Description	Quartus II Command from Processing Menu
Full compilation flow	Performs a full compilation of the current design.	**Start Compilation** command
Compilation and simulation flow	If the simulation mode is timing, flow performs a full compilation and then a simulation of the current design. If the simulation mode is functional, the flow runs only the **Generate Functional Simulation Netlist** command and then simulates the current design.	**Start Compilation and Simulation** command
SignalProbe™ flow	Routes user-specified signals to output pins without affecting the existing fitting in a design, so that you can debug signals without completing a full compilation.	**Start SignalProbe Compilation** command
Early Timing Estimate	Performs a partial compilation, but stops and generates early timing estimates with the Classic Timing Analyzer before the Fitter is complete.	**Start Early Timing Estimate** command