ARCHITECTURES OF FPGAs AND INTRODUCTION OF VHDL

ABSTRACT

The architectures of FPGAs of Altera and Xilinx vendors used in the present work are discussed in this chapter. Further the introduction of Hardware description language (VHDL) is given.
Chapter 3  Architectures of FPGAs and Introduction of VHDL

3.1 Introduction

This chapter deals with the description of field programmable gate arrays (FPGAs). In this Altera FPGAs and Xilinx Spartan3 are discussed. Further the advantages of FPGAs for DSP applications and comparison of FPGA based applications with programmable DSP processors are covered.

3.1.1 Architecture of FPGA:

The general structure of FPGA is as shown in the Fig 3.1. It contains the following resources: logic blocks, I/O blocks, and Interconnections. The I/O blocks are used to connect internal circuitry to the pins of the package and interconnections are related to wires and switches. The logic blocks are arranged in a two-dimensional array and the interconnection wires are organized as horizontal and vertical routing channels between rows and columns of logic blocks. The routing channels contain wires and programmable switches that allow the logic blocks to be inter-connected in many ways. Fig 3.1 shows two locations for programmable switches: the solid boxes adjacent to logic blocks hold switches that connect the logic block input & output terminals to the interconnection wires and the solid boxes that are diagonally between logic blocks connect one interconnection wire to another (such as a vertical wire to a horizontal wire). Programmable blocks are exists between the I/O blocks and the interconnection wires. FPGAs can be used to implement logic circuits of more than a few hundred thousand equivalent gates in size. As mentioned earlier two popular examples of FPGAs are the Altera and Xilinx. Each logic block in an FPGA typically has a small number of inputs and outputs. The most commonly used logic block is a lookup table (LUT) which
contains storage cells. These storage cells are used to implement a small logic function. Each cell is capable of holding a single logic value, either 0 or 1. The stored value is produced as the output of the storage cell. FPGAs memory is volatile (SRAM-Based). The FPGA loses its configuration during a power failure and must be reprogrammed when power is restored. Xilinx and Altera offer several families of PROM devices and other FPGA Programming methods through which automatic reloading takes place after the power is retained.

An FPGA is a regular structure [1] of logic cells or modules and interconnect which is under the designer's complete control. This means that the user can design, program and make changes to his circuit whenever he needs.

There are two basic types of FPGAs:

- SRAM – based One-Time programmable (OTP). Further they are classified into the following:
  - (i) Static Ram cells
  - (ii) Anti-Fuse
  - (iii) EPROM Transistors
  - (iv) EEPROM Transistors.

i. Static Ram Technology:

In the static Ram FPGA, programmable connections are made up of pass transistors, transmission gates or multiplexers that are controlled by SRAM cells. The advantage of their technology is that it allows fast-in-circuit reconfiguration. The major disadvantage of this technology is that it requires more chip area.
ii. Anti-fuse technology:

Anti-fuse technology works on the principle of high impedance state and can be programmed into low impedance or "fused" state. It is less expensive than RAM technology, but this device is programmable only once. Hence, it is popularly known as one time programmable (OTP) device.

iii. EPROM/EEPROM Technology:

This technique is same as that used in the EPROM memories. One merit of this technology is that it can be reprogrammed without external storage of computation though the EPROM transistors cannot be reprogrammed in circuit. The dominant type of FPGA is SRAM-based and can be reprogrammed by the user as many number of times as he chooses.

The SRAM based FPGA:

An SRAM FPGA is reprogrammed every time it is powered-up because the FPGA is really a fancy memory chip. That is the reason why user needs a serial PROM or system memory with every SRAM FPGA.

The SRAM determines inter connect and it defines logic in look up table (LUT) while in case of one-time programmable. The inter connect is anti-fuse and logic as traditional gates.

In the SRAM logic cell, instead of conventional gates, there is a look up table (LUT) which determines the output based on the values of the inputs. SRAM bits are also used to make connections.
A field programmable gate array (FPGA) provides the benefit of custom CMOS, VLSI, and avoids the initial cost, time delay, inherent cost and risk of a conventional marked gate array. Loading configuration data into internal memory cells customizes the FPGAs. The FPGA can either actively read its configuration data out of external serial or byte parallel PROM (master mode) or the configuration data can be written into the FPGA (slave and peripheral mode).

Customized configuration is established by programming internal static memory cells that determines the logic functions and internal connections implemented in the FPGA.

A FPGA with a two dimensional array of logic blocks that can be interconnected by interconnection wires. All internal connections are composed of metal segments with programmable switching point to implement the desired routing. Different routing resources are provided to achieve an efficient automated routing. There are four main classes of interconnections which are being distinguished by the relative length of their segments. These are single-length lines, double length lines and long lines. In addition, there will be global buffers which drive the signals fast with low skew nets.

The internal configurable logic block (CLB) contains a pair of Flip-flops and two independent 4-input function generators. These function generators have a good deal of flexibility as most combinational logic functions required less than four inputs. CLB is the one which implement the most of the logic in an FPGA. The flexibility and symmetry of the CLB architecture facilitates the placement and routing of a given application.
Chapter 3  
Architectures of FPGAs and Introduction of VHDL

The architectural features of the Xilinx FPGAs used in the present work along with the Altera FPGAs are described below.

Fig 3.1: FPGA Block diagram
3.2 ALTERA FPGAs:

3.2.1 Cyclone FPGA:

The cyclone field programmable gate array family is based on a 1.5V, 0.13μm, all layer copper SRAM process with densities up to 20,060 logic elements (Les) and up to 288 Kbits of RAM. With the features like phase locked loops (PLLs) for clocking and dedicated double data rate (DDR) interfaces to meet DDR SDRAM and fast cycle RAM (FCRAM) memory requirements. Cyclone devices are cost-effective solution for data path applications. These devices support various I/O standards, including LVDS at data rates up to 311 megabits per second (Mbps) and 66-MHz, 32-bit peripheral component inter connect (PCI) for connecting width and supporting ASSP & ASIC devices. Altera also offers new low-cost serial configuration devices to configure cyclone devices [1].

3.2.2 Features:

- 2,910 to 20,060 Les
- Up to 294912 RAM bits (36,864 bytes)
- Supports configuration through low-cost serial configuration device.
- Support for LVTTL, LVCMOS, SSTL-2, and SSTL-3 I/O standards
- Support for 66-MHz, 32-bit-PCI standard.
- Low speed (311 Mbps) LVDS I/O support.
- Up to two PLLs per device provide clock multiplication and phase shifting.
- Up to eight global clock lines with six clock resources available per logic array block (LAB) row.
Chapter 3 Architectures of FPGAs and Introduction of VHDL

- Support external memory, including DDRSDRAM (133 MHz), FCRAM and single data rate (SDR) (SDRAM)
- Supports for multiple intellectual property (IP) cores, including Altera Mega core functions and Altera Mega functions partner’s program (AMPP) mega functions.

3.2.3 Functional Description:

Cyclone device contains two-dimensional row and column-based architecture to implement logic circuits. Column and row interconnects of varying speeds provide a signal interconnects between logic array blocks (LAB) and embedded memory blocks. The logic array consists of LABs with 10 logic elements (LES) in each LAB. An LE is a small unit of logic providing efficient implementation of user logic functions. LABs are grouped into rows and columns across the device. Cyclone devices have a range of 2,910 to 20,060 LES.

M4KRAM blocks are true dual-port memory blocks with 4k bits of memory plus parity (4,608 bits). These blocks provide dedicated true dual-port, simple dual port or single-port memory up to 36-bits wide at up to 200MHz. These blocks are grouped into columns across the device in between certain LABs. Cyclone devices offer between 60 to 288 k-bits of embedded RAM. Each cyclone device I/O pin is fed by an I/O element (IOE) located at the ends of LAB rows and columns around the periphery of the device. I/O pins support various single ended and differential I/O standards, such as the 66 MHz, 32-bit PCI standard and LVDS I/O standard at up to 311 Mbps. Each IOE contains a bi-directional I/O buffer and three for registering input, output and output enable signals. Dual -purpose DQS, DQ and DM pins along with delay chains (used to phase align DDR
Chapter 3 Architectures of FPGAs and Introduction of VHDL

signals) provide interface support with external memory devices such as DDR, SDRAM and FCRAM devices at up to 133 MHz (266 Mbps).

Cyclone devices provide a global clock network and up to two PLLS. The global clock network consists of eight global clock lines that drive throughout the entire device. The global clock network can provide clocks for all resources within the device such as IOEs, LEs, and memory blocks. The global clock lines can also be used for control signals. Cyclone PLLs provide general-purpose clocking with clock multiplication and phase shifting as well as external outputs for high speed differential I/O support.

3.2.4 Logic array blocks (LAB):

Each LAB consists of 10 LES, LE carry chains, LAB control signals, a local interconnect, look-up-tale (LUT) chain and register chain connection lines. The local interconnect transfers signals among LEs in the same LAB. LUT chain connections transfer the output of one LE’s LUT to the adjacent LE for fast sequential LUT connections within the same LAB. Register chain connections transfer the output of one LE’s register to the adjacent LE’s register within the LAB. The Quartus 11 compiler places associated logic within the LAB or adjacent LAB, allowing the use of local LUT chain and register chain connections for performance and area efficiency.

3.3 Stratix FPGA:

Stratix family of PLDs is based on a 1.5V, 0.13 μm, all layer copper SRAM process with densities up to 114,140 logic elements (LEs) and up to 10M bits of RAM. Stratix devices offer up to 28 digital signal processing (DSP) blocks with up to 224 (9 bit x 9 bit) embedded multipliers, optimized for DSP applications that enable efficient
implementation of high performance filters and multipliers. Stratix devices support various I/O Standards and also offer a complete clock management solution with its hierarchical clock structure with up to 420 MHz performances and up to 12 phase locked loops (PLLs).

3.3.1 Features:

- 10570 to 114,140 LEs
- Up to 10,118,016 RAM bits (1,264,752 bytes) available without reducing logic resources.
- Tri Matrix Memory consisting of three RAM block sizes to implement true dual port memory and first in first out (FIFO) buffers.
- High-speed DSP blocks provide dedicated implementations of multipliers (up to 250 MHz) multiply-accumulate functions, and finite impulse response (FIR) filters.
- Up to 16 global clocks with 22 clocking resources per device region.
- Up to 12 enhanced PLLs per device provide spread spectrum, programmable bandwidth, clock switchover, real time PLL reconfiguration and advanced multiplication and phase shifting.
- Support for numerous single-ended and differential I/O standards.
- High speed differential I/O support up to 116 channels with up to 80 channels optimized for 840 megabits per second (Mbps)
- Support for high speed net working and communication bus standards including rapid IO, UTOPIAIV, CSIX, Hyper Transport-technology 10G Ethernet XSB1, SPI-4 phase 2 (POS-PHY level – 4) and SFI-4.
Terminator technology provides on-chip termination for differential and single ended I/O pins with impedance matching.

Support for high speed external memory, including zero bus turn around (ZBT) SRAM, quad data rate (QDR/QDR11) SRAM, double data rate (DDR) SDRAM, DDR fast cycle RAM (FCRAM) and single data rate (SDR) SDRAM.

Support for multiple intellectual property mega functions from Altera Mega core functions and Altera Mega Function partners program mega functions.

Support for remote configuration updates.

3.3.2 Functional Description:

Stratix devices contain a two-dimensional row and column-based architecture to implement custom logic. A series of column and row interconnects of varying length and speed provides signal interconnects between logic array blocks (LABs), memory block structures and DSP blocks.

The logic array consists of LABs, with 10 logic elements (LEs) in each LAB. An LE is a small unit of logic providing efficient implementation of user logic functions. LABs are grouped into rows and columns across the device.

M512 RAM blocks are simple dual-port memory blocks with 512 bits plus parity (576 bits). These blocks provide dedicated true dual-port, simple dual-port or single-port memory up to 36-bits wide at up to 291 MHz. These blocks are grouped into columns across the device in between certain LABs. M-RAM blocks are true dual-port with 512 k bits plus parity (589,824 bits). The blocks provide dedicated true dual-port, simple dual-
port, or single port memory up to 144 bits wide at up to 296 MHz. Several M-RAM blocks are located individually or in pairs within the devices logic array.

Digital signal processing (DSP) blocks can implement up to either eight full-precision 9x9 multiplier, four full precision 18x18 bit multipliers or one full precision 36x36-bit multiplier with add or subtract features. These blocks also contain 18-bit input shift registers for digital signal processing applications including finite impulse response (FIR) and infinite impulse response (IIR) filters. DSP blocks are grouped into two columns in each device. Each stratix device I/O pin is fed by an I/O element (IOE) located at the end of LAB rows and columns around the periphery of the device. I/O pins support numerous single-ended and differential I/O standards. Each IOE contains a bi-directional I/O buffer and six registers for registering input, output and output enable signals. When used with dedicated clocks, these registers provide exceptional performance and interface support with external memory devices such as DDR SDRAM, FCRAM, ZBT and QDR SRAM devices. High-speed serial interface channels support transfers up to 840 Mbps using LVDs, LVPECL, and 3.3-V PC ML or Hyper Transport technology I/O standards. The number of M512 RAM, M4KRAM, and DSP blocks varies from device to device along with the row and column numbers and M-RAM blocks.
3.4 Flex10ke FPGA:

FLEX10KE is an embedded programmable logic device and it is enhanced version of FLEX10KE devices. Based on the re-configurable CMOS SRAM elements, the FLEX10KE architecture incorporates all features necessary to implement common gate array mega functions. With up to 2,0000 typical gates, FLEX10KE devices provide the density speed and features to integrate entire systems including multiple 32-bit buses into a single device.

The ability to reconfigure FLEX10KE devices enables 100% testing prior to shipment and allows the designer to focus on simulation and design verification. FLEX10KE reconfigurability eliminates inventory management for gate array designs and generation of test vectors for fault coverage [2].

3.4.1 Features:

- Embedded programmable logic devices (PLDs) providing system-on-a-programmable chip (SOPC) integration in a single device.
- Enhanced embedded array for implementing mega functions such as efficient memory and specialized logic functions.
- Dual-port capability with up to 16-bit width per embedded array block (EAB).
- Logic array for general logic functions.
- High density
- 30,000 to 200,000 typical gates.
- Up to 98,304 RAM bits (4,096 bits per EAB), all of which can be used without reducing logic capacity.
3.4.2 System-level features.

- Multi-volt I/O pins can drive or be driven by 2.5V, 3.3V, or 5.0V devices.
- Low power consumption.
- Bi-directional I/O performance up to 212 MHz.
- Fully compliant with the PCI special interest group. PCI Local Bus specification, revision 2.2 for 3.3-V operation at 33MHz or 66 MHz.
- Built in Joint Test Action Group (JTAG) boundary scans Test (BST) and circuitry complaint with IEEE std. 1149.1-1990, available without consuming additional logic device.
- Fabricated on advanced process and operate with a 2.5V internal supply voltage.
- In-circuit reconfigurability (ICR) via external configuration devices, intelligent controller, or JTAG port.
- Clock lock and Clock Boost options for reduced clock delay/skew and clock multiplication.
- Built-in-low-skew clock distribution trees
- 100% functional testing of all devices test vectors or scan chains are not required.
- Pull-up on I/O pins before and during configuration.

3.4.3 Flexible inter connect

- Fast tract-inter connect continuous routing structure for fast, predictable and inter connect delays dedicated carry chain that implements high-speed and high fan-ion logic functions (automatically used by software tools and mega functions)
- Tri-State emulation that implements internal tri-state buses.
- Up-to six global clock signals and four global clear signals.
3.4.4 Powerful I/O pins

- Individual tri-state output enable control for each pin.
- Open-drain option on each I/O pin.
- Programmable output slew-rate control to reduce switching noise.
- Clamp to Vccio user selectable on a pin-by-pin basis.
- Supports hot-socketing.
- Software designs support and automatic place and route provided by Altera’s development system for Windows-based PCs and Sun SPRAC station and HP 900 series 700/800.

3.4.5 Flexible package options

- Available in a variety of packages with 144 to 672 pins, including the innovative find line B9A packages.
- Same frame pin-out compatibility between FLEX10KA and FLEX10KE devices across a range of device densities and pin counts.
- Additional design entry and simulation support provided by EDIF 200 and 300 net list files, library of parameterized modules (LPM), Design ware components, Verilog HDL, VHDL and other interfaces to popular EDA tools from manufactures such as cadence, Exempler logic, Mentor Graphics, or CAD synopsis etc.
3.4.6 Functional Description:

Each FLEX10KE device contains an enhanced embedded array and logic array. The embedded array is used to implement a variety of memory functions or complex logic functions, such as digital signal processing (DSP), wide data path manipulation, micro controller applications and data-transfer functions. The logic array performs the same function as the sea-of-gates in the gate array and is used to implement general logic such as counters, adders, state machines and multiplexers. The combination of embedded and logic arrays provide the high performance and high density to embedded gate arrays enabling designers to implement an entire system on a single device.

The embedded array consists of series of embedded array blocks (EABs). When implementing memory functions, each EAB provides 4,096 bits which can be used to create RAM, ROM, dual-port RAM or first in first out (FIFO) functions. When implementing logic, each EAB can contribute 100 to 600 gates towards complex logic functions.

The logic array consists of logic array blocks (LABs). Each LAB contains eight LEs and local interconnect. An LE consists of a four input look-up-table, a programmable flip-flop and dedicated signal paths for carry and cascade functions. The eight LEs can be used to create medium-sized blocks of logic – such as eight bit counters and address decoders. Combined across LAB to create larger logic blocks. Each LAB represents about 96 usable logic gates.

FLEX10KE devices are configured at system power-up with data stored in an Altera serial configuration device or provided by a system controller. Altera offers the
EPC1, EPC2, and EPC16 configuration devices which configure FLEX10KE devices via a serial data stream.

FLEX10KE devices contain an interface that permits microprocessor to configure FLEX10KE devices synchronously or asynchronously. The interface also enables microprocessors to treat a FLEX10KE device as memory and configure it by writing to virtual memory location, making it easy to reconfigure the device. The Fast Track Inter Connect Routing Structure provides signal interconnections within FLEX10KE devices which is a series of fast, continuous row and column channels that run the entire length and width of the device.

Each group of LEs is combined into an LAB. Groups of LABs are arranged into rows and columns. Each row also contains a single EAB. The Fast Track Inter Connect Routing Structure interconnects the LABs and EAB. IOEs are located at the end of each row and column of the Fast Track Inter Connect Routing Structure.

3.5 Mercury FPGA:

Mercury devices integrate high-speed differential transceivers and support for CDR with a speed-optimized PLD architecture. These transceivers are implemented through the dedicated serializer, deserializer and clock recovery circuitry. This circuitry together with enhanced I/O elements and support for numerous I/O standards, allows Mercury devices to meet high-speed interface requirements.

Mercury devices are the first PLDs optimized for core performance. These LUT-based enhanced memory devices use a network of fast routing resources to achieve
optimal performance. These resources are ideal for data-path, register-intensive, mathematical, digital signal processing (DSP) or communication designs.
3.5.1 Configuration:

The logic circuitry and interconnections in the Mercury architecture are configured with CMOS SRAM elements. Mercury devices are reconfigurable and are 100% tested prior to shipment. Mercury devices can be configured on the board for the specific functionally required. Mercury devices are configured at a system power-up with data stored in an Altera serial configuration or provided by the system controller.

3.5.2 Features:

High Performance Programmable Logic Device (PLD) Family [3].

- Integrated high-speed transceivers with support for clock data recovery (CDR) at up to 1.25 giga bits per second (Gbps).
- Look-up table (LUT) – based architecture optimized for high speed.
- Advanced inter-connect structure for fast routing of critical paths.
- Enhanced I/O structure for versatile standards and interface support.
- Up to 14,400 logic elements (LE’s)

3.5.3 System – Level Features:

- Up to four general-purpose phase-locked loops (PLLs) with programmable multiplication and delay shifting.
- Up to 12PLL output ports.
- Dedicated multiplier circuitry for high-speed implementation of signed or unsigned multiplication up to 16x16.
Chapter 3 Architectures of FPGAs and Introduction of VHDL

- Embedded system blocks (ESBs) used to implement memory functions including quad-port RAM, true dual-port RAM, first-in-first out buffers (FIFO) and content-addressable memory (CAM).
- Each ESB contains 4096 bits and can be split and used as two 2048-bit unidirectional dual-port RAM blocks.

3.5.4 Advanced high-speed I/O features:

- Robust I/O standard support, including LUTTL and PCI up to 66 MHz, 3.3 V
- High-speed differential interface (HSDI) with dedicated circuitry for CDR at up to 1.25 Gbps.
- Support for source-synchronous True-LVDS circuitry up to 840 megabits per second (Mbps) for LVDS and LVPECL.
- Up to 18 input and 18 output dedicated differential channels of high speed LVDS and LVPECL.
- Built-in 100Ω termination resistor on HSDI data and clock differential pairs.
- Flexible-LVDS-circuitry provides 624-Mbps supports on up to 100 channels with EP1M 350 device.
- Versatile three-register I/O element supporting double data rate I/O (DDRIO), double data rate (DDR) SD RAM, Zero bus turn around (ZBT) SRAM, and quad data rate (QDR) SRAM.

3.5.5 Designed for low-power operation:

- 1.8v internal supply storage ($V_{CCINT}$) Multi volt I/O interface voltage levels ($V_{CCIO}$) compatible with 1.5V, 1.8V, 2.5V and 3.3V devices.
3.5.6 Advanced interconnect structure:

- Multi-level fast track interconnects structure providing fast, predictable interconnect delays.
- Optimized high-speed priority fast track interconnects for routing critical path in a design.
- Dedicated carry chain that implements arithmetic functions such as fast adders, counters and comparators (automatically used by software tools and mega functions).
- Fast LUT connection allowing high-speed direct connection between LEs in the same logic array blocks (LAB).
- Leap lines allowing a single LAB to directly drive LEs in adjacent rows.
- The Rapid LAB interconnects providing a high-speed connection to a 10-LAB wide region.
- Dedicated clock and control signal resources, including four dedicated clocks, six dedicated fast global signals and additional row-global signals.

3.5.7 Functional Description:

The Mercury architecture contains a row-based logic array to implement general logic and a row-based embedded and specialized logic functions. Signal interconnections within Mercury devices are provided by a series of row and column interconnects with varying lengths and speeds. The priority Fast Track Interconnect structure is faster than other interconnects. The Quartus II compiler places design critical paths on these faster lines to improve design performance.
Mercury device I/O pins are evenly distributed across the entire device area. Other Altera device families have I/O pins placed on the device periphery.

3.6 Xilinx

Introduction

The Xilinx Spartan-3 Starter Kit provides a low-cost, easy-to-use development and evaluation platform for Spartan-3 FPGA designs compared to others.

3.6.1 Key Components and Features

In the present work the Spartan-3 Starter Kit board is used, which includes the following components and features:

i. 200,000-gate Xilinx Spartan-3 XC3S200 FPGA in a 256-ball thin Ball Grid Array package (XC3S200FT256)
   - 4,320 logic cell equivalents
   - Twelve 18K-bit block RAMs (216K bits)
   - Twelve 18x18 hardware multipliers
   - Four Digital Clock Managers (DCMs)
   - Up to 173 user-defined I/O signals

ii. 2Mbit Xilinx XCF02S Platform Flash in-system programmable configuration PROM
   - 1Mbit non-volatile data or application code storage available after FPGA configuration.
   - Jumper option allow FPGA application to read PROM data or FPGA configuration from other sources.
iii. 1M-byte of Fast Asynchronous SRAM (bottom side of board)
   • Two 256Kx16 ISSI IS6LV25616AL-10T 10 ns SRAMs
   • Configurable memory architecture
     • Single 256Kx32 SRAM array, ideal for MicroBlaze code images
     • Two independent 256Kx16 SRAM arrays
   • Individual chip select per device
   • Individual byte enables

iv. 3-bit, 8-color VGA display port

v. 9-pin RS-232 Serial Port
   • Rs-232 transceiver / level translator
   • RS-Uses straight-through serial cable to connect to computer or workstation serial port
   • Second RS-232 transmit and receive channel available on board test points

vi. PS/2-style mouse / keyboard port

vii. Four-character, seven-segment LED display

viii. Eight slide switches

ix. Eight individual LED outputs

x. Four momentary-contact push button switches

xi. 50 MHz crystal oscillator clock source (bottom side of board)

xii. Socket for an auxiliary crystal oscillator clock source

xiii. FPGA configuration mode selected via jumper settings

xiv. Push button switch to force FPGA reconfiguration (FPGA configuration happens automatically at power-on)
Chapter 3 - Architectures of FPGAs and Introduction of VHDL

xv. LED indicates when FPGA is successfully configured

xvi. Three 40-pin expansion connection ports to extend and enhance the Spartan-3 Starter Kit Board

- FPGA serial configuration interface signals available on the A2 and B1 connectors – PROG_B, DONE, INIT_B, CCLK, DONE

xvii. JTAG port for low-cost download cable

xviii. Digilent JTAG download/debugging cable connects to PC parallel port

xix. JTAG download / debug port compatible with the Xilinx Parallel Cable IV and MultiPRO Desktop Tool

xx. AC power adapter input for included international unregulated +5V power supply

xxi. Power-on indicator LED

xxii. On-board 3.3V, 2.5V, and 1.2V regulators
Figure 3.3: Xilinx Spartan3 starter kit board Block Diagram
Figure 3.4: Xilinx Spartan3 starter kit board (Top view)

Figure 3.5: Xilinx Spartan3 starter kit board (Bottom view)
3.6.2 Four-Digit, Seven-Segment LED Display

The Spartan-3 Starter Kit board has a four-character, seven segment LED display controlled by FPGA user-I/O pins. Each digit shares eight common control signals to light individual LED segments. Each individual character has a separate anode control input. A detailed schematic for the display.

The pin number for each FPGA pin connected to the LED display appears in parentheses. To light an individual signal, drive the individual segment control signal Low along with the associated anode control signal for the individual character. For example, the left-most character displays the value ‘2’. The digital values driving the display in this example are shown in blue. The AN3 anode control signal is Low, enabling the control inputs for the left-most character. The segment control inputs, A through G and DP, drive the individual segments that comprise the character. A Low value lights the individual segment, a High turns off the segment. A Low on the A input signal, lights segment ‘a’ of the display. The anode controls for the remaining characters, AN[2:0] are all High, and these characters ignore the values presented on A through G and DP.

The LED control signals are time-multiplexed to display data on all four characters. Present the value to be displayed on the segment control inputs and select the specified character by driving the associated anode control signal Low. Through persistence of vision, the human brain perceives that all four characters appear simultaneously, similar to the way the brain perceives a TV display.
This “scanning” technique reduces the number of I/O pins required for the four characters. If an FPGA pin were dedicated for each individual segment, then 32 pins are required to drive four 7-segment LED characters. The scanning technique reduces the required I/O down to 12 pins. The drawback to this approach is that the FPGA logic must continuously scan data out to the displays – a small price to save 20 additional I/O pins.

3.6.3 VGA Port

The Spartan-3 Starter Kit board includes a VGA display port and DB15 connector, indicated as $\circ$ in Figure 3.3. Connect this port directly to most PC monitors or flat-panel LCD displays using a standard monitor cable.

Spartan-3 FPGA controls five VGA signals: Red (R), Green (G), Blue (B), Horizontal Sync (HS), and Vertical Sync (VS), all available on the VGA connector. The FPGA pins that drive the VGA port appear in Table 3.1

<table>
<thead>
<tr>
<th>Signal</th>
<th>FPGA Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red (R)</td>
<td>R12</td>
</tr>
<tr>
<td>Green (G)</td>
<td>T12</td>
</tr>
<tr>
<td>Blue (B)</td>
<td>R11</td>
</tr>
<tr>
<td>Horizontal Sync (HS)</td>
<td>R9</td>
</tr>
<tr>
<td>Vertical Sync (VS)</td>
<td>T10</td>
</tr>
</tbody>
</table>
Each color line has a series resistor to provide 3-bit color, with one bit each for Red, Green and Blue. The series resistor uses the $75\Omega$ VGA cable termination to ensure that the color signals remain in the VGA-specified 0V to 0.7V range. The HS and VS signals are TTL level.

Figure 3.6: Xilinx Spartan3 family Architecture
Figure 3.7: Basic Xilinx Spartan2 family FPGA Block Diagram
Chapter 3 Architectures of FPGAs and Introduction of VHDL

Drive the R, G, and B signals High or Low to generate the eight possible colors shown in Table 3-2.

Table 3-2: 3-Bit Display Color Codes

<table>
<thead>
<tr>
<th>Red (R)</th>
<th>Green (G)</th>
<th>Blue (B)</th>
<th>Resulting Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Blue</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Cyan</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Magenta</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Yellow</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>White</td>
</tr>
</tbody>
</table>

VGA signals timing is specified, published, copyrighted and sold by the Video Electronics Standards Association (VESA). The following VGA system and timing information is provided as an example of how the FPGA might drive VGA monitor in 640 by 480 mode.

3.6.4 Signal Timing for a 60Hz, 640x480 VGA Display

CRT-based VGA displays use amplitude-modulated, moving electron beams (or cathode rays) to display information on a phosphor-coated screen. LCD displays use an array of switches that can impose a voltage across a small amount of liquid crystal, thereby changing light permittivity through the crystal on a pixel-by-pixel basis. Although the following description is limited to CRT displays, LCD displays have evolved to use the same signal timings as CRT displays. Consequently, the following discussion pertaining to both CRTs and LCD displays.

93
Within a CRT display, current waveforms pass through the coils to produce magnetic fields that deflect electron beams to transverse the display surface in a "raster" pattern, horizontally from left to right and vertically from top to bottom. Information is only displayed when the beam is moving in the "forward" direction-left to right and top to bottom-and not during the time the beam returns back to the left or top edge of the display. Much of the potential display time is therefore lost in "blanking" periods when the beam is reset and stabilized to begin a new horizontal or vertical display pass.

The sizes of the beam, the frequency at which the beam traces across the display and the frequency at which the electron beam is modulates determine the display resolution.

Modern VGA displays support multiple display resolutions and the VGA controller dictates the resolution by producing timing signals to control the raster patterns. The controller produces TTL-level synchronizing pulses that set the frequency at which current flows through the deflection coils and it ensures that pixel or video data is applied to the electron guns at the correct time.

Video data typically comes from a video refresh memory with one or more bytes assigned to each pixel location. The Spartan-3 Starter Kit board uses three bits per pixel, producing one of the eight possible colors shown in Table 3-2. The controller indexes into the video data buffer as the beams move across the display. The controller then retrieves and applies video data to the display at precisely the time the electron beam is moving across a given pixel [4].
Chapter 3 Architectures of FPGAs and Introduction of VHDL

The VGA controller generates the HS (horizontal sync) and VS (vertical sync) timings signals and coordinates the delivery of video data on each pixel clock. The pixel clock defines the time available to display one pixel of information. The VS signal defines the "refresh" frequency of the display or the frequency at which all information on the display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity with practical refresh frequencies in the 60Hz to 120 Hz range. The number of horizontal lines displayed at a given refresh frequency defines the horizontal "retrace" frequency.

3.6.5 VGA Signal Timing

The signal timings in Table 3.3 are derived for a 640-pixel by 480-row display using a 25MHz pixel clock and 60 Hz ± 1 refresh. The timing for the sync pulse width ($T_{pw}$) and front and back porch intervals ($T_{fp}$ and $T_{bp}$) are based on observations from various VGA displays. The front and back porch intervals are the pre- and post-sync pulse times. Information cannot be displayed during these times.

Table 3-3: 640x480 Mode VGA Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Vertical Sync</th>
<th>Horizontal Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>Clocks</td>
</tr>
<tr>
<td>$T_S$</td>
<td>Sync pulse time</td>
<td>16.7 ms</td>
<td>416,800</td>
</tr>
<tr>
<td>$T_{DISP}$</td>
<td>Display time</td>
<td>15.36 ms</td>
<td>384,000</td>
</tr>
<tr>
<td>$T_{PW}$</td>
<td>Pulse width</td>
<td>64 µs</td>
<td>1,600</td>
</tr>
<tr>
<td>$T_{FP}$</td>
<td>Front porch</td>
<td>320 µs</td>
<td>8,000</td>
</tr>
<tr>
<td>$T_{BP}$</td>
<td>Back porch</td>
<td>928 µs</td>
<td>23,200</td>
</tr>
</tbody>
</table>
Generally, a counter clocked by the pixel clock controls the horizontal timing. Decoded counter values generate the HS signal. This counter tracks the current pixel display location on a given row.

A separate counter tracks the vertical timing. The vertical-sync counter increments with each HS pulse and decoded values generate the VS signal. This counter tracks the current display row. These two continuously running counters from the address into a video display buffer. For example, the on-board fast SRAM is an ideal display buffer.

No time relationship is specified between the onset of the HS pulse and the onset of the VS pulse. Consequently the counters can be arranged to easily from video RAM addresses or to minimize decoding logic for sync pulse generation.

3.6.6 PS/2 Mouse / Keyboard Port

The Spartan-3 Starter Kit board includes a PS/2 mouse / keyboard port and the standard 6-pin mini-DIN connector labeled J3 on the board and indicated as in Figure 3.3. The PS/2 connector and Table 3-4 shows the signals on the connector. Only pins 1 and 5 of the connector attach to the FPGA.

<table>
<thead>
<tr>
<th>PS/2 DIN Pin</th>
<th>Signal</th>
<th>FPGA Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA (PS2D)</td>
<td>M15</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>Voltage Supply</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>CLK (PS2C)</td>
<td>M16</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>
Both a PC mouse and keyboard use the two-wire PS/2 serial bus to communicate with a host device, the Spartan-3 FPGA in this case. The PS/2 bus includes both clock and data. Both a mouse and keyboard drives the bus with identical signal timings and both use 11-bit words that include a start, stop and odd parity bit. However, the data packets are organized differently for a mouse and keyboard. Furthermore, the keyboard interface allows bidirectional data transfers so the host device can illuminate state LEDs on the keyboard.

The PS/2 bus timing appears in Table 3-5. The clock and data signals are only driven when data transfers occur and otherwise they are held in the idle state at logic High. The timings define signal requirements for mouse-to-host communications and bidirectional keyboard communications. The attached keyboard or mouse writes a bit on the data line when the clock signal is High and the host reads the data line when the clock signal is Low.

Table 3-5: PS/2 Bus Timing

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{UK}$</td>
<td>Clock High or Low time</td>
<td>30 $\mu$s</td>
<td>50 $\mu$s</td>
</tr>
<tr>
<td>$T_{SU}$</td>
<td>Data-to-clock setup time</td>
<td>5 $\mu$s</td>
<td>25 $\mu$s</td>
</tr>
<tr>
<td>$T_{HLD}$</td>
<td>Clock-to-data hold time</td>
<td>5 $\mu$s</td>
<td>25 $\mu$s</td>
</tr>
</tbody>
</table>
3.6.7 Keyboard

The keyboard uses open-collector drivers so that either the keyboard or the host can drive the two-wire bus. If the host never sends data to the keyboard, then the host can use simple input pins.

A PS/2-style keyboard uses scan codes to communicate key press data. Nearly all keyboards in use today are PS/2 style. Each key has a single, unique scan code that is sent whenever the corresponding key is pressed.

If the key is pressed and held, the keyboard repeatedly sends the scan code every 100 ms or so. When a key is released, the keyboard sends a "F0" key-up code, followed by the scan code of the released key. The keyboard sends the same scan code, regardless if a key has different "shift" and "non-shift" characters and regardless whether the Shift key is pressed or not. The host determines which character is intended.

Some keys called extended keys send an "E0" ahead of the scan code and furthermore, they may send more than one scan code. When an extended key is released, a "E0 F0" key-up code is sent followed by the scan code. The host can also send data to the keyboard. Table 3-6 provides a short list of some often used commands.
Table 3-6: Common PS/2 Keyboard Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED</td>
<td>Turn on/off Num Lock, Caps Lock, and Scroll Lock LEDs. They keyboard acknowledges receipt of an “ED” command by replying with an “FA”, after which the host sends another byte to set LED status. The bit positions for the keyboard LEDs appear in Table 6-4. Write a ‘1’ to the specific bit to illuminate the associated keyboard LED.</td>
</tr>
<tr>
<td>EE</td>
<td>Echo. Upon receiving an echo command, the keyboard replies with the same scan code “EE”.</td>
</tr>
<tr>
<td>F3</td>
<td>Set scan code repeat rate. The keyboard acknowledges receipt of an “F3” by returning an “FA”, after which the host sends a second byte to set the repeat rate.</td>
</tr>
<tr>
<td>FE</td>
<td>Resend. Upon receiving a resend command, the keyboard resends the last scan code sent.</td>
</tr>
<tr>
<td>FF</td>
<td>Reset. Resets the keyboard.</td>
</tr>
</tbody>
</table>

The keyboard sends data to the host only when both the data and clock lines are High, the idle state.

Because the host is the “bus master”, the keyboard checks whether the host is sending data before driving the bus. The clock line can be used as a “clear to send” signal. If the host pulls the clock line Low, the keyboard must not send any data until the clock is released.

The keyboard sends data to the host in 11-bit words that contain a ‘0’ start bit, followed by eight bits of scan code (LSB first), followed by an odd parity bit and terminated with a ‘1’ stop bit. When the keyboard sends data, it generates 11 clock transitions at around 20 to 30 kHz, and data is valid on the falling edge of the clock [5].
3.6.8 Mouse

A mouse generates a clock and data signal when moved. Otherwise, these signals remain High indicating the Idle state. Each time the mouse is moved, the mouse sends three 11-bit words to the host. Each of the 11-bit words contains a ‘0’ start bit, followed by 8 data bits (LSB first), followed by an odd parity bit and terminated with a ‘1’ stop bit. Each data transmission contains 33 total bits, where bits 0, 11 and 22 are ‘0’ start bits and bits 10, 21 and 32 are ‘1’ stop bits. The three 8-bit data fields contain movement data. Data is valid at the falling edge of the clock and the clock period is 20 to 30 kHz.

As shown in Figure 3.5, a PS/2 mouse employs a relative coordinate system wherein moving the mouse to the right generates a positive value in the X field and moving to the left generates a negative value. Likewise, moving the mouse up generates a positive value in the Y field and moving down represents a negative value. The XS and YS bits in the status byte define the sign of each value, where a ‘1’ indicates a negative value.

The magnitude of the X and Y values represent the rate of mouse movement. The larger the value, the faster the mouse is moving. The XV and YV bits in the status byte indicate when the X or Y values exceed their maximum value, an overflow condition. A ‘1’ indicates when an overflow occurs. If the mouse moves continuously, the 33-bit transmissions repeat every 50 ms or so.

The L and R fields in the status byte indicate Left and Right button presses. A ‘1’ indicates that the associated mouse button is being pressed.
3.6.9 Voltage Supply

Most modern keyboards and mice work equally well from a 3.3V or 5V supply. The voltage supply for the PS/2 port is selectable via the JP2 jumper, indicated as (30) in Figure 3.3 located immediately above the PS/2 connector along the right edge. The 3.3V setting is preferred as the FPGA's output signals operate from the 3.3V supply.

Some older keyboards and mice are 5V only. Consequently, the JP2 jumper should be set for 5V operation. The Spartan-3 FPGA can tolerate 5V signals due to the 270Ω series resistors on the PS/2 data and clock signals connected to the FPGA [6].

3.6.10 RS-232 Port

The Spartan-3 Starter Kit board has an RS-232 serial port. The RS-232 transmit and receive signals appear on the female DB9 connector, labeled J2, indicated as (6) in Figure 3.3. The connector is a DCE-style port and connects to the DB9 DTE-style port connector available on most personal computers and workstations. Use a standard straight-through serial cable to connect the Spartan-3 Starter Kit board to the PC's serial port.

The connection between the FPGA and the DB9 connector, including the Maxim MAX3232 RS-232 voltage converter, indicated as (7) in Figure 3.3. The FPGA supplies serial output data as LVTLL or LVCMOS levels to the Maxim device, which in turn converts the logic value to the appropriate RS-232 voltage level. Likewise, the Maxim device converts the RS-232 serial input data to LVTLL levels for the FPGA. A series resistor between the Maxim output pin and the FPGA’s RXD pin protects against accidental logic conflicts.
Hardware flow control is not supported on the connector. The port’s DCD, DTR and DSR signals connect together. Similarly, the port’s RTS and CTS signals connect together.

An auxiliary RS-232 serial channel from the Maxim device is available on two 0.1-inch stake pins, indicated as J1 in the schematic and 8 in Figure 3.3. The FPGA connections driving the Maxim device appear in Table 7-1 with signals RXD-A and TXD-A.

3.6.11 Clock Sources

The Spartan-3 Starter Kit board has a dedicated 50 MHz Epson SG-8002JF series clock oscillator source and an optional socket for another clock oscillator source.

The 50 MHz clock oscillator is mounted on the bottom side of the board, indicated as in Figure 3.3. Use the 50 MHz clock frequency as is or derives other frequencies using the FPGAs Digital Clock Managers (DCMs).

The oscillator socket, indicated as 14 in Figure 3.3, accepts oscillators in an 8-pin DIP footprint.

<table>
<thead>
<tr>
<th>Oscillator Source</th>
<th>FPGA Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 MHz (IC4)</td>
<td>T9</td>
</tr>
<tr>
<td>Socket (IC8)</td>
<td>D9</td>
</tr>
</tbody>
</table>
3.7 FPGA Configuration Modes and Functions

3.7.1 FPGA Configuration Mode Settings

In most applications for the Spartan-3 Starter Kit Board, the FPGA automatically boots from the on-board Platform Flash memory whenever power is applied or the PROG push button is pressed. However, the board supports all the available configuration modes via the J8 header, indicated as \[16\] in Figure 3.3. Additionally, the JP1 jumper setting is required when using Master Serial configuration mode, as further described in "Platform Flash Jumper Options (JP1)."

The default jumper settings for the board are:

- All jumpers in the J8 header are installed
- The JP1 jumper is in the "Default" position

3.7.2 Program Push Button/DONE Indicator LED

The Spartan-3 Starter Kit Board includes two FPGA configuration functions, located near the VGA connector and the AC power input connector drives the FPGA’s PROG_B programming pin. When pressed, the PROG push button forces the FPGA to reconfigure and reload its configuration data.

3.7.3 Platform Flash Configuration Storage

The Spartan-3 Starter Kit Board has an XCF02S serial configuration Flash PROM to store FPGA configuration data and potentially additional non-volatile data, including MicroBlaze application code. To configure the FPGA from Platform Flash memory, all three jumpers must be installed on the J8 header, indicated as \[16\] in Figure 3.3.
3.7.4 Platform Flash Jumper Options (JP1)

The Platform Flash has three optional settings controlled by the JP1 jumper, which is located in the upper right-hand corner of the board, adjacent to the Platform Flash configuration PROM. The JP1 jumper is indicated as ![JP1 jumper](image) in Figure 3.3.

3.7.5 “Default” Option

For most applications, this is the default jumper setting. The Platform Flash is enabled only during configuration when the FPGA’s DONE pin is Low. When the DONE pin goes High at the end of configuration, the Platform Flash is disabled and placed in low-power mode [7].

3.7.6 “Flash Read” Option

The Spartan-3 Starter Kit Board includes a 2Mbit Platform Flash configuration PROM. The XC3S200 FPGA on the board only requires slightly less than 1 Mbit for configuration data. The remainder of the Platform Flash is available to store other non-volatile data, such as revision codes, serial numbers, coefficients, an Ethernet MAC ID or code for an embedded processor, such as MicroBlaze within the FPGA.

To allow the FPGA to read from Platform Flash after configuration, the JP1 jumper must be properly positioned. When the jumper is in this position, the Platform Flash is always enabled. After FPGA configuration completes, the FPGA application derives the INIT_B pin High, FPGA pin N9. Consequently, the Platform Flash data pointer is not reset and points to the additional data following the FPGA configuration data. To read any subsequent data, the FPGA application generates additional clock
pulses on the RCLK signal from FPGA pin A14. After configuration, the FPGA’s CCLK output is three-state with a pull-up resistor to \( V_{ccaux} \) (2.5V). The Platform Flash presents serial data on the FPGA’s DIN pin, pin M11.

The resistor between the CCLK output and FPGA pin A14 prevents any accidental conflicts between the two signals.

3.7.7 “Disable” Option

If the JP1 jumper is removed, then the Platform Flash is disabled, potentially allowing configuration via an expansion board connected to one of the expansion connectors.

3.8 Introduction to VHDL

VHDL is a language for describing digital systems. It came from United States Government’s Very High Speed Integrated Circuits (VHSIC) program. It is a standard language, hence the VHSIC Hardware Description Language (VHDL) was developed. A few details of this language is given below [8].

3.8.1 Elements of Behavior

In VHDL, a description of the internal implementation of an entity is called an architecture body of the entity. There may be a number of different architecture bodies of the one interface to an entity, corresponding to alternative implementations that perform the same function. We can write a behavioral architecture body of an entity which describes the function in an abstract way. Such an architecture body includes only process statements, which are collections of actions to be executed in sequence. These
actions are called sequential statements and are much like the kinds of statements we see in a conventional programming language. The types of actions that can be performed include evaluating expressions, assigning values to variables, conditional execution, repeated execution and subprogram calls. In addition, there is a sequential statement that is unique to hardware modeling languages, the signal assignment statement. This is similar to variable assignment, except that it causes the value on a signal to be updated at some future time.

Example: Behavioral architecture for the four-bit register

To illustrate these ideas, let us look at a behavioral architecture body for the reg4 entity of below.

```
architecture behav of reg4 is
begin

storage : process is
  variable stored_d0, stored_d1, stored_d2, stored_d3 : bit;

begin

  wait until clk;

  if en then
    stored_d0:=d0;
    stored_d1:=d1;
    stored_d2:=d2;
    stored_d3:=d3;
  end if;

  q0<=stored_d0 after 5 ns;

```

106
In this architecture body, the part after the first begin keyword includes one process statement which describes how the register behaves. It starts with the process name, storage and finishes with the keywords end process.

The process statement defines a sequence of actions that are to take place when the system is simulated. These actions control how the values on the entity’s ports change over time; that is, they control the behavior of the entity. This process can modify the values of the entity’s ports using signal assignment statements [9].

The way this process works is as follows. When the simulation is started, the signal values are set to ‘0’; and the process is activated. The process’s variables (listed after the keyword variable) are initialized to ‘0’; then the statements are executed in order. The first statement is a wait statement that causes the process to suspend, that is, to become inactive. It stays suspended until one of the signals to which it is sensitive changes the value. In this case, the process is sensitive only to the signal clock since that is the only one named in the wait statement. When that signals changes value, the process is resumed and continues executing statements. The next statement is a condition that tests whether the value of the en signal is ‘1’. If it is, the statements between the keywords then and end if are executed, updating the process’s variables using the values
on the input signals. After the conditional if statement, there are four signal assignment statements that cause the output signals to be updated 5 ns later.

When all of these statements in the process have been executed, the process starts again from the keyword `begin` and the cycle repeats. Notice that while the process is suspended, the values in the process’s variables are not lost. This is how the process can represent the state of a system.

### 3.8.2 Elements of structure

An alternative way of describing the implementation of an entity is to specify how it is composed of subsystems. We can give a structural description of the entity’s implementation. An architecture body that is composed only of interconnected subsystems is called a structural architecture body.

**Example: Structural architecture for the four-bit register**

If we are to describe this in VHDL, we will need entity declarations and architecture bodies for the subsystems. For the flipflops, the entity and architecture are

```vhdl
entity d_ff is
    port ( d, clk : in bit; q : out bit );
end d_ff;
architecture basic of d_ff is
begin
    ff_behavior : process is
    begin
        wait until clk;
    end ff_behavior;
end basic;
```

```vhdl
architecture basic of d_ff is
begin
    ff_behavior : process is
    begin
        wait until clk;
    end ff_behavior;
end basic;
```
For the two-input and gate, the entity and architecture are

```
entity and2 is
  Port (a, b : in bit; y : out bit);
end and2:
architecture basic of and2 is
begin
  and2_behavior : process is
  begin
    y <= a and b after 2 ns;
    wait on a, b;
  end process and2_behavior;
end architecture basic;
```

We can now proceed to a VHDL architecture body declaration as following.

```
architecture struct of reg4 is
begin
  signal int_clk : bit;
  bit0 : entity work.d_ff(basic)
  port map (d0, int_clk, q0);
  bit1 : entity work.d_ff(basic)
  port map (d1, int_clk, q1);
  bit2 : entity work.d_ff(basic)
  port map (d2, int_clk, q1);
```

The signal declaration, before the keyword begin, defines the internal signals of the architecture. In this example, the signal int_clk is declared to carry a bit value ('0' or '1'). In general, VHDL signals can be declared to carry arbitrarily complex values. Within the architecture body the ports of the entity are also treated as signals [10].

In the second part of the architecture body, a number of component instances are created, representing the subsystems from which the reg4 entity is composed. Each component instance is a copy of the entity representing the subsystem, using the corresponding basic architecture body. (The name work refers to the current working library, in which all of the entity and architecture body descriptions are assumed to be held.)

The port map specifies the connection of the ports of each component instance to signals within the enclosing architecture body. For example, bit0, an instance of the d_ff entity has its port d connected to the signal d0, its port clk connected to the signal int_clk and its port q connected to the signal q0.
3.9 BASIC RADAR SYSTEMS

Radar systems, like other complex electronics systems, are composed of several major subsystems and many individual circuits. Although modern radar systems are quite complicated, one can easily understand their operation by using a basic block diagram of a pulse-radar system[11].

3.9.1 FUNDAMENTAL RADAR SYSTEM

Since most radars used today are some variation of the pulse-radar system, this section discusses those used in a pulse radar. All other types of radars use some variation of these units. The block diagram of the fundamental Radar system is as shown in figure 3.8.

![Figure 3.8—Block diagram of a fundamental radar system](image-url)
Modulator

One can see on the block diagram that the heart of the radar system is the modulator. It generates all the necessary timing pulses (triggers) for use in the radar and its associated systems. Its function is to ensure that all subsystems of the radar system operate in a definite time relationship with one another and that the intervals between pulses, as well as the pulses themselves, are of the proper length.

Transmitter

The transmitter generates powerful pulses of electromagnetic energy at precise intervals. The required power is obtained by using a high-power microwave oscillator (such as a magnetron) or a microwave amplifier (such as a klystron) that is supplied by a low-power RF source.

Duplexer

The duplexer is essentially an electronic switch that permits a radar system to use a single antenna to both transmit and receive. The duplexer must connect the antenna to the transmitter and disconnect the antenna from the receiver for the duration of the transmitted pulse. The switching time is called receiver recovery time, and must be very fast if close-in targets are to be detected.

Antenna System

The antenna system routes the pulse from the transmitter, radiates it in a directional beam, picks up the returning echo, and passes it to the receiver with a minimum of loss. The antenna system includes the antenna, transmission lines, and
waveguide from the transmitter to the antenna, and transmission lines and waveguide from the antenna to the receiver.

**Receiver**

The receiver accepts the weak RF echoes from the antenna system and routes them to the indicator as discernible video signals. Because the radar frequencies are very high and difficult to amplify, a super-heterodyne receiver is used to convert the echoes to a lower frequency, called the intermediate frequency (IF), which is easier to amplify.
References:


8. VHDL by Sjoholm Stefan.

9. The student guide to VHDL, Peter J. Ashenden.


11. Fire control man volume02-firecontrol radar fundamentals