Fpga – college essay

FPGAs ABSTRACT This report provides a survey of architectures of commercially available highcapacity field-programmable logic devices (FPLDs) ex. FPGAs and also the applications of FPGAs. We first define the relevant terminology in the field and then describe the recent evolution of FPLDs. The three main categories of FPLDs are delineated: Simple PLDs (SPLDs), Complex PLDs (CPLDs) and Field-Programmable Gate Arrays (FPGAs). The details of the architectures of the most important commercially available FPGAs are given. 1 FPGAs CHAPTER 1 INTRODUCTION TO HIGH CAPACITY FPLDs

Prompted by the development of new types of sophisticated field-programmable logic devices (FPLDs), the process of designing digital hardware has changed dramatically over the past few years. Unlike previous generations of technology, in which board-level designs included large numbers of SSI chips containing basic gates, virtually every digital design produced today consists mostly of high-density devices. This applies not only to custom devices like processors and memory, but also for logic circuits such as state machine controllers, counters, registers, and decoders.

When such circuits are destined for high-volume systems, they have been integrated into high-density gate arrays. However, gate array NRE costs often are too expensive and gate arrays take too long to manufacture to be viable for prototyping or other low-volume scenarios. For these reasons, most prototypes, and also many production designs are now built using FPLDs. The most compelling advantages of FPLDs are instant manufacturing turnaround, low start-up costs, low financial risk and (since programming is done by the end user) ease of design changes.

The market for FPLDs has grown dramatically over the past decade to the point where there is now a wide assortment of devices to choose from. A designer today faces a daunting task to research the different types of chips, understand what they can best be used for, choose a particular manufacturer’s product, learn the intricacies of vendor-specific software and then design the hardware. Not only not only the sheer number of FPLDs available exacerbates confusion for designers, but also by the complexity of the more sophisticated devices.

The purpose of this paper is to provide an overview of the architecture of the various types of FPLDs. The emphasis is on devices with relatively high logic capacity; all of the most important commercial products are discussed. Before proceeding, we provide definitions of the terminology in this field. This is necessary because the technical jargon has become somewhat inconsistent over the past few years as companies have attempted to compare and contrast their products in literature. 2 FPGAs 1. 1 Definitions of Relevant Terminology The most important terminology used in this report is defined below. Field-Programmable Logic Device (FLPD) — a general term that refers to any type of integrated circuit used for implementing digital hardware, where the chip can be configured by the end user to realize different designs. Programming of such a device often involves placing the chip into a special programming unit, but some chips can also be configured “ in-system”. Another name for FPLDs is programmable logic devices (PLDs); although PLDs encompass the same types of chips as FPLDs, we prefer the term FPD because historically the word PLD has referred to relatively simple types of devices. PLA — a Programmable Logic Array (PLA) is a relatively

small FPLD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable (note: although PLA structures are sometimes embedded into full-custom chips, we refer here only to those PLAs that are provided as separate integrated circuits and are userprogrammable). • PAL—a Programmable Array Logic (PAL) is a relatively small FPLD that has a programmable AND-plane followed by a fixed OR-plane. • SPLD — refers to any type of Simple PLD, usually either a PLA or PAL. CPLD — a more Complex PLD that consists of an arrangement of multiple SPLD-like blocks on a single chip. Alternative names (that will not be used in this paper) sometimes adopted for this style of chip are Enhanced PLD (EPLD), Super PAL, Mega PAL, and others. • FPGA — a Field-Programmable Gate Array is an FPLD featuring a general structure that allows very high logic capacity. Whereas CPLDs feature logic resources with a wide number of inputs (AND planes), FPGAs offer more narrow logic resources. FPGAs also offer a higher ratio of flip-flops to logic resources than do CPLDs. FPGAs • HCPLDs — high-capacity PLDs: a single acronym that refers to both CPLDs and FPGAs. This term has been coined in trade literature for providing an easy way to refer to both types of devices. We do not use this term in the report. • Interconnect — the wiring resources in an FPLD. • Programmable Switch — a user-programmable switch that can connect a logic element to an interconnect wire, or one interconnect wire to another. • Logic Block — a relatively small circuit block that is replicated in an array in an FPLD.

When a circuit is implemented in an FPLD, it is first decomposed into smaller subcircuits that can each be mapped into a logic block. The term logic block

is mostly used in the context of FPGAs, but it could also refer to a block of circuitry in an CPLD. • Logic Capacity — the amount of digital logic that can be mapped into a single FPLD. This is usually measured in units of " equivalent number of gates in a traditional gate array". In other words, the capacity of an FPLD is measured by the size of gate array that it is comparable to.

In simpler terms, logic capacity can be thought of as " number of 2input NAND gates". • Logic Density — the amount of logic per unit area in an FPLD. • Speed-Performance — measures the maximum operable speed of a circuit when implemented in an FPLD. For combinational circuits, it is set by the longest delay through any path, and for sequential circuits it is the maximum clock frequency for which the circuit functions properly. In the remainder of this section, to provide insight into FPLD development the evolution of FPLDs over the past two decades is described.

Additional background information is also included on the semiconductor technologies used in the manufacture of FPLDs. 4 FPGAs • Full Custom / VLSI IC Technology — In this technology, all layers are optimized for a particular implementation. Such optimization includes placing the transistors to minimize interconnection lengths, sizing the transistors to optimize signal transmissions and routing wires among the transistors. This is often referred as VLSI design; – has very high NRE cost and long turn around times.

Has excellent performance with small size and power. • Semi Custom / ASIC Technology (Gate array and Standard cell) — In an application specific IC (ASIC) technology, the lower layers are fully or partially built, leaving the

designers to finish the upper layers. • Gate array — The masks for the transistors and gate levels are already built (array of gates) for the particular implementation. • Standard cell — Logic level cells such as AND gate or AND-OR-INVERT combination mask portions are pre designed.

Remaining task is to arrange these portions in to complete masks for the gate level and to connect the cells. ASICs are most popular among them, as they provide good performance and size, less NRE cost; but may take weeks / months to get manufactured. 5 FPGAs CHAPTER 2 EVOLUTION OF PROGRAMMABLE LOGIC DEVICES The first type of user-programmable chip that could implement logic circuits was the Programmable Read-Only Memory (PROM), in which address lines can be used as logic circuit inputs and data lines as outputs.

Logic functions, however, rarely require more than a few product terms, and a PROM contains a full decoder for its address inputs. PROMS are thus an inefficient architecture for realizing logic circuits, and so are rarely used in practice for that purpose. The first device developed later specifically for implementing logic circuits was the Field-Programmable Logic Array (FPLA), or simply PLA for short. A PLA consists of two levels of logic gates: a programmable “ wired” AND-plane followed by a programmable “ wired” OR-plane.

A PLA is structured so that any of its inputs (or their complements) can be ANDed together in the AND-plane; each AND-plane output can thus correspond to any product term of the inputs. Similarly, each OR-plane output can be configured to produce the logical sum of any of the AND-plane

outputs. With this structure, PLAs are well suited for implementing logic functions in sum-of-products form. They are also quite versatile, since both the AND terms and OR terms can have many inputs (this feature is often referred to as wide AND and OR gates).

When Philips introduced PLAs in the early 1970s,, their main drawbacks were that they were expensive to manufacture and offered somewhat poor speed-performance. Both disadvantages were due to the two levels of configurable logic, because programmable logic planes were difficult to manufacture and introduced significant propagation delays. To overcome these weaknesses, Programmable Array Logic (PAL) devices were developed. As Figure 1 illustrates, PALs feature only a single level of programmability, consisting of a programmable “ wired” AND plane that feeds fixed OR-gates.

To compensate for lack of generality incurred because the Or-outputs plane is fixed, several variants of PALs are produced, with different numbers of inputs and outputs, and various sizes of OR-gates. PALs usually contain flip-flops connected to the OR-gate outputs so that sequential circuits can be realized. PAL devices are important because when introduced they had a profound effect on digital hardware design, and also they are the basis for some of the newer, more sophisticated architectures that will be described 6 FPGAs shortly.

Variants of the basic PAL architecture are featured in several other products by different acronyms. All small PLDs, including PLAs, PALs, and PAL-like devices are grouped into a single category called Simple PLDs (SPLDs), whose most important characteristics are low cost and very high pin-to-pin

speed-performance. As technology has advanced, it has become possible to produce devices with higher capacity than SPLDs. The difficulty with increasing capacity of a strict SPLD architecture is that the Figure 1: Structure of a PAL tructure of the programmable logic-planes grows too quickly in size as the number of inputs is increased. The only feasible way to provide large capacity devices based on SPLD architectures is then to integrate multiple SPLDs onto a single chip and provide interconnect to programmable connect the SPLD blocks together. Many commercial FPD products exist on the market today with this basic structure, and are collectively referred to as Complex PLDs (CPLDs). Altera, pioneered CPLDs first in their family of chips called Classic EPLDs, and then in three additional series, called MAX 5000, MAX 7000 and MAX 9000.

Because of a rapidly growing market for large FPLDs, other manufacturers developed devices in the CPLD category and there are now many choices available. CPLDs provide logic capacity up to the equivalent of about 50 typical SPLD devices, but it is somewhat difficult to extend these architectures to higher densities. To build FPLDs with very high logic capacity, a different approach is needed. The highest capacity general-purpose logic 7 FPGAs chips available today are the traditional gate arrays sometimes referred to as MaskProgrammable Gate Arrays (MPGAs).

MPGAs consist of an array of pre-fabricated transistors that can be customized into the user’s logic circuit by connecting the transistors with custom wires. Customization is performed during chip fabrication by specifying the metal interconnect, and this means that in order for a user to employ an MPGA a large setup cost is involved and manufacturing time is

long. Although MPGAs are clearly not FPLDs, they are mentioned here because they motivated the design of the user-programmable equivalent: Field-Programmable Gate Arrays (FPGAs).

Like MPGAs, FPGAs comprise an array of uncommitted circuit elements, called logic blocks, and interconnect resources, but FPGA configuration is performed through programming by the end user. An illustration of a typical FPGA architecture appears in Figure 2. As the only type of FPLD that supports very high logic capacity, FPGAs have been responsible for a major shift in the way digital circuits are designed. Figure 2: Structure of an FPGA Figure 3 summarizes the categories of FPLDs by listing the logic capacities available in each of the three categories.

In the figure, " equivalent gates" refers loosely to " number of 2-input NAND gates". The chart serves as a guide for selecting a specific device for a given application, depending on the logic capacity needed. However, as we will discuss 8 FPGAs shortly, each type of FPLD is inherently better suited for some applications than for others. Figure 3: FPLD Categories by Logic Capacity It should also be mentioned that there exist other special-purpose devices optimized for specific applications (e. g. state machines, analog gate arrays, large interconnection problems).

However, since use of such devices is limited they will not be described here. The next sub-section discusses the methods used to implement the user-programmable switches that are the key to the user-customization of FPLDs. 9 FPGAs CHAPTER 3 FIELD-PROGRAMMABLE GATE ARRAY 3. 1 What is an 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 interconnect form a fabric of basic building blocks for logic circuits. Combining these basic blocks to create the desired circuit creates complex designs. 3. 2 What does a logic cell do? 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. . 3 What does ‘ Field Programmable’ mean? 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 10 FPGAs function defined at the time of manufacture. In contrast, a program written by someone other than the

device manufacturer defines the FPGA’s function. 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. 3. 4 Classes of FPGA Architecture There are two primary classes of FPGA architectures namely • • coarse-grained fine-grained Coarse-grained architectures consist of fairly large logic blocks, often containing two or more look-up tables and two or more flip-flops. In a majority of these architectures, a four-input look-up table (think of it as a 16×1 ROM) implements the actual logic. The larger logic block usually corresponds to improved performance.

The other architecture type is called fine-grained. In these devices, there are a large number of relatively simple logic blocks. The logic block usually contains either a two-input logic function or a 4-to1 multiplexer and a flip-flop. These devices are good at systolic functions and have some benefits for designs created by logic synthesis. Table 1: Classes of FPGA Architecture 11 FPGAs 3. 5 How are FPGA programs created? 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. Characteristics of FPGA Technology Technology Volatile Re-Prog Chip Area Static RAM yes in-circuit large PLICE Antianti-fuse — small no no Fuse prog. rans. — large ViaLink Antianti-fuse — small no no Fuse prog. trans. — large out of EPROM no small circuit out of EEPROM no 2x EPROM circuit R (ohm) C (ff) 1 -2 K 10 – 20 ff 300 – 500 3 – 5 ff 50 – 60 2-4k 2-4k 3 – 5 ff 10 -20 ff 10 -20 ff Table 2: Characteristics of FPGA Technology 12 FPGAs 3. 6 Programming the FPGA The standard digital design flow for Hardware Object implementation is as below STEP 1: DESIGN ENTRY Entry Tools • • • • • Viewlogic – ProSeries Synopsis Exemplar Logic Synthesis System ALDEC – Active FPGA DATA I/O – Synario Output – Translated to FPGA Netlist Figure 4: Design Flow Diagram — Design Entry

In the Digital Design Stage the digital design is created with a schematic digital design editor or a Hardware Description Language (HDL). The Schematic entry program utilizes graphic symbols of the circuitry. The HDL entry program uses a descriptive language (i. e. Verilog, ABEL or VHDL). There are a wide variety of design entry software programs available. As the output of these programs produce netlists, one must be sure the library sets of the targeted FPGA are available in the tool we have selected. STEP 2: DESIGN IMPLEMENTATION – MAPPING In the Design Implementation stage, the netlist produced by the design entry rogram is converted into the bitstream file, which configures the FPGA. The first step maps the design

onto the FPGA resources. The second step Places or assigns logic blocks created in the mapping process in specific locations in the FPGA. The third step Routes the interconnect paths between the logic blocks. The output is a Logic Cell Array File (LCA) for the particular FPGA. This LCA file than then converted into a bitstream file for configuring the FPGA. 13 FPGAs Entry Tools • • • • • Xilinx – XACT Synopsis Exemplar Logic Synthesis System ALDEC – Active FPGA DATA I/O – Synario

Output – Routes and Maps the design Generates a FPGA Bitstream Configuration File Figure 5: Design Flow Diagram — Design Implementation STEP 3: DESIGN VERIFICATION – SIMULATION Entry Tools • • • • • Viewlogic – ProSeries Synopsis Cadence VHDL – XL, Verilog – XL ALDEC – Active FPGA DATA I/O – Synario Output – Simulation of design – Pass/Fail If Fail >> Return to Design Entry If Pass >> Configure FPGA Figure 6: Design Flow Diagram — Design Verification The Design Verification Step tests the design's logic and timing using input stimuli. Various CAE software packages provide verification/simulation tools.

These tools are designed to perform detailed characterization of the design, by performing both functional and timing simulations. In-circuit verification is another way to test the design. 14 FPGAs In-circuit verification tests the circuit under typical operating conditions. The Virtual Computer, reconfigurable computer can be used as an in-circuit verification system. STEP 4: FPGA CONFIGURATION – RUN Configuration Methods • Master Mode: loads from external memory (PROM) in parallel or Serial data mode Non-Master Mode: loads from another device (MPU, FPGA) in Peripheral Sync/Async or slave serial Figure 7: Design Flow Diagram — FPGA

Configuratic Configuration is a process in which the circuit design (bitstream file) is downloaded into the FPGA. The method of configuring the FPGA determines the type of bitstream file. A PROM can configure FPGAs. The serial PROM is the most common. 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).

Where the FPGA is used in a Reconfigurable Computing Platform, the bitstream file is converted into a High Level Language (i. e. ‘ C”) function. Through this method the FPGA is configured from within an application program. 3. 7 Placement and Routing Due to the fixed layout and limited routing resources of FPGAs, placement and routing are very important issues in FPGAs. In general, synthesis tools do not consider placement and routing. Low-level operations such as place and route are normally the problem domain of fitters, which understand the architecture of the specific target device.

Only so can one expect reasonable operations, even though few fitters today feature sophisticated place and route strategies. 15 FPGAs Fitters also perform low-level optimizations to take full advantage of the target architecture. In many cases, these fitters are also responsible for mapping a generic net list onto LUT-based CLB structures. Depending on the synthesis tool used, partitioning into CLBs may occur either during the synthesis or routing step. A good example is Synopsys, which offers two different synthesis tools, Design Compiler and FPGA Compiler.

While FPGA compiler understands the LUT-based structure of CLBs, Design Compiler views design as sea-of-gates structures, with each gate contributing a given delay. Obviously, the FPGA compiler reports more accurate timing information and can perform better target-specific optimization. However, not all FPGA families are supported with FPGA Compiler. Thus, while the Xilinx 4k series is supported by FPGA Compiler (a library for Design Compiler also exists), the Xilinx 3k and 7k and the Altera Flex8000 families are only supported by Design Compiler.

The level and quality of timing information given by Design Compiler varies, depending on the family. While Design Compiler does not provide any timing information at all for Xilinx 7k EPLD series, the Altera Flex8000 family does contain timing information. The timing information provided by the Altera Flex8k library seems to contain worst-case timing information for single gates. Since multiple gates can normally be allocated to a single LUT, delays are grossly over-estimated. Realistic timing reports can only be generated after the placement and routing using Altera’s fitter maxplus2.

As timing characteristics are influenced heavily by placement and routing in FPGA technology, meaningful gatelevel simulation can only be performed with back-annotated timing information generated by the appropriate fitter. 3. 8 Non-Deterministic And Deterministic FPGAs Non-deterministic: No accurate timing analysis can be obtained before placement and routing. Examples: Xilinx XC4000, Actel ACT 3. For Xilinx XC4000, a non-deterministic coarse-grained FPGA, The Xilinx LCA architecture does not permit an accurate timing analysis until after place and route.

This is because of the coarse-grained nondeterministic architecture. For Actel ACT 3, a non-deterministic fine-grained FPGA, the 16 FPGAs Actel ACT architecture is non-deterministic, but the fine-grained structure allows fairly accurate preroute timing prediction. Deterministic: Accurate timing analysis can be obtained before placement and routing. Examples: Altera MAX 7000. For Altera MAX 7000, a deterministic complex PLD, the Altera MAX CPLD requires logic to be fitted to the product steering and programmable array logic. The Altera MAX 7000 has an almost deterministic architecture, which allows accurate preroute timing. . 9 General FPGA Problems 1. Interconnect delays are not as predictable as in CPLDs. 2. Vendor specific tools to map a design in their device. (Special fitting software) => almost impossible to migrate a design from one vendor to another and maintain some anything like same propagation delays. 3. Tools generate netlists targeted for ASICs but not FPGAs. => Difficult to predict timings before placement and routing. 17 FPGAs CHAPTER 4 USER-PROGRAMMABLE SWITCH TECHNOLOGIES 4. 1 Fuse: The first type of user-programmable switch developed was the fuse used in PLAs.

Although fuses are still used in some smaller devices, we will not discuss them here because they are quickly being replaced by newer technology. For higher density devices, where CMOS dominates the IC industry, different approaches to implementing programmable switches have been developed. For CPLDs the main switch technologies (in commercial products) are floating gate transistors like those used in EPROM and EEPROM, and for FPGAs they are SRAM and antifuse. Each of these is briefly discussed below. 4. 2 EPROM and EEPROM: Figure 8: EPROM Programmable Switches

An EEPROM or EPROM transistor is used as a programmable switch for CPLDs (and also for many SPLDs) by placing the transistor between two wires in a way that facilitates implementation of wired-AND functions. This is illustrated in Figure 8, which shows EPROM transistors as they might be connected in an AND-plane of a CPLD. An input to the AND-plane can drive a product wire to logic level ‘ 0’ through an EPROM 18 FPGAs transistor, if that input is part of the corresponding product term. For inputs that are not involved for a product term, the appropriate EPROM transistors are programmed to be permanently turned off.

A diagram for an EEPROM based device would look similar. Although there is no technical reason why EPROM or EEPROM could not be applied to FPGAs, current commercial FPGA products are based either on SRAM or antifuse technologies, as discussed below. 4. 3 Static Memory Technology (SRAM): Figure 9: SRAM-controlled Programmable Switches SRAM in the form of Look-Up Tables (LUTs) are widely used in FPGAs to implement the programmable combinatorial logic of the device. It is similar to the technology used in static RAM devices but with a few modifications.

The RAM cells in a memory device are optimized for fastest possible read/write performance. The RAM cells in a programmable device are usually designed for stability instead of read/write performance. Consequently, RAM cells in a programmable device have a low-impedance connect to VCC and ground to provide maximum stability over voltage fluctuations. An example of usage of SRAM-controlled switches is illustrated in Figure 9, showing two applications of SRAM cells: for controlling the gate nodes of pass-transistor

switches 19 FPGAs and to control the select lines of multiplexers that drive logic block inputs.

The figures gives an example of the connection of one logic block (represented by the AND-gate in the upper left corner) to another through two pass-transistor switches, and then a multiplexer, all controlled by SRAM cells. Whether an FPGA uses pass-transistors or multiplexers or both depends on the particular product. 4. 4 Anti-fuse: The other type of programmable switch used in FPGAs is the antifuse. Antifuses are originally open-circuits and take on low resistance only when programmed. Antifuses are suitable for FPGAs because they can be built using modified CMOS technology.

As an example, Actel’s antifuse structure, known as PLICE, is depicted in Figure 10. The figure shows that an antifuse is positioned between two interconnect wires and physically consists of three sandwiched layers: the top and bottom layers are conductors, and the middle layer is an insulator. When unprogrammed, the insulator isolates the top and bottom layers, but when programmed the insulator changes to become a low-resistance link. PLICE uses Poly-Si and n+ diffusion as conductors and ONO as an insulator, but other antifuses rely on metal for conductors, with amorphous silicon as the middle layer.

Table 3 lists the most important characteristics of the programming technologies. The left-most column of the table indicates whether the programmable switches are one-time programmable (OTP), or can be re-programmed (RP). The next column lists whether the switches are volatile,

and the last column names the underlying transistor technology. Figure 10: Actel Antifuse Structure 20 FPGAs 4. 5 Flash technology: Flash-erased (or bulk erased) electrically erasable programmable is read-only memory. Flash has the electrically erasable benefits of EEPROM but the small, economical cell size of EPROM technology.

Switch type Fuse EPROM Reprogrammable No Yes (out of circuit) Yes (in circuit) Yes (in circuit) No Yes Volatile No No Technology Bipolar UVCMOS EEPROM No EECMOS SRAM Antifuse Flash Yes No No CMOS CMOS+ Table 3: Summary of Programming Technologies 21 FPGAs CHAPTER 5 OVERVIEW OF COMMERCIALLY AVAILABLE FPGAs As one of the largest growing segments of the semiconductor industry, the FPGA market place is volatile. As such, the pool of companies involved changes rapidly and it is somewhat difficult to say which products will be the most significant when the industry reaches a stable state.

For this reason, and to provide a more focused discussion, we will not mention all of the FPGA manufacturers that currently exist, but will instead focus on those companies whose products are in widespread use at this time. In describing each device we will list its capacity, nominally in 2-input NAND gates as given by the vendor. Gate count is an especially contentious issue in the FPGA industry, and so the numbers given in this paper for all manufacturers should not be taken too seriously. Wags have taken to calling them “ dog” gates, in reference to the traditional ratio between human and dog years.

There are two basic categories of FPGAs on the market today: 1. SRAM-based FPGAs and 2. Antifuse-based FPGAs. In the first category, Xilinx and Altera are the leading manufacturers in terms of number of users, with the major competitor being AT&T. For antifuse-based products, Actel, Quicklogic and Cypress, and Xilinx offer competing products. Company Actel Altera QuickLogic Xilinx Architecture Row-based Hierarchial-PLD Symmetrical Array Symmetrical Array Logic Block Type Multiplexer-Based PLD Block Multiplexer-Based Look-up Table Programming Technology Anti-fuse EPROM Anti-fuse Static RAM

Table 4: Commercially available FPGAs 22 FPGAs 5. 1 Xilinx SRAM-based FPGAs Figure 11: Basic building blocks for XILINX FPGAs The basic structure of Xilinx FPGAs is array-based, meaning that each chip comprises a two dimensional array of logic blocks that can be interconnected via horizontal and vertical routing channels. An illustration of this type of architecture was shown in Figure 11. Xilinx introduced the first FPGA family, called the XC2000 series, in about 1985 and now offers three more generations: XC3000, XC4000, and XC5000.

Although the XC3000 devices are still widely used, we will focus on the more recent and more popular XC4000 family. We note that XC5000 is similar to XC4000, but has been engineered to offer similar features at a more attractive price, with some penalty in speed. We should also note that Xilinx has recently introduced FPGA family-based anti-fuses, called the XC8100. The XC8100 has many interesting features, but since it is not yet in widespread use, we will not discuss it here. The Xilinx 4000 family devices range in capacity from about 2000 to more than 15, 000 equivalent gates.

The XC4000 features a logic block (called a Configurable Logic Block (CLB) by Xilinx) that is based on look-up tables (LUTs). A LUT is a small one-bit wide memory array, where the address lines for the memory are inputs of the logic block and the one bit output from the memory is the LUT output. A LUT with K inputs would then correspond to a 2K x 1 bit memory, and can 23 FPGAs realize any logic function of its K inputs by programming the logic function’s truth table directly into the memory. The XC4000 CLB contains three separate LUTs, in the configuration shown in Figure 12.

There are two 4-input LUTS that are fed by CLB inputs, and the third LUT can be used in combination with the other two. This arrangement allows the CLB to implement a wide range of logic functions of up to nine inputs, two separate functions of four inputs or other possibilities. Each CLB also contains two flip-flops. Figure 12: Xilinx XC4000 Configurable Logic Block (CLB) Toward the goal of providing high-density devices that support the integration of entire systems, the XC4000 chips have “ system oriented” features. For instance, each CLB contains circuitry that allows it to efficiently perform arithmetic (i. . , a circuit that can implement a fast carry operation for adder-like circuits) and also the LUTs in a CLB can be configured as read/write RAM cells. A new version of this family, the 4000E, has the additional feature that the RAM can be configured as a dual port RAM with a single write and two read ports. In the 4000E, RAM blocks can be synchronous RAM. Also, each XC4000 chip includes very wide AND-planes around the periphery of the logic block array to facilitate implementing circuit blocks such as wide decoders. Besides logic, the 24 FPGAs Figure 13: Block diagram of XC4000 IOB ther key feature that characterizes an FPGA is

its interconnect structure. The XC4000 interconnect is arranged in horizontal and vertical channels. Each channel contains some number of short wire segments that span a single CLB (the number of segments in each channel depends on the specific part number), longer segments that span two CLBs, and very long segments that span the entire length or width of the chip. Programmable switches are available (see Table 2) to connect the inputs and outputs of the CLBs to the wire segments, or to connect one wire segment to another.

A small section of a routing channel representative of an XC4000 device appears in Figure 13. The figure shows only the wire segments in a horizontal channel, and does not show the vertical routing channels, the CLB inputs and outputs, or the routing switches. An important point worth noting about the Xilinx interconnect is that signals must pass through switches to reach one CLB from another, and the total number of switches traversed depends on the particular set of wire segments used. Thus, speed-performance of an implemented circuit 25 FPGAs depends in part on how the wire segments are allocated to individual signals by CAD tools.

Figure 14: Xilinx XC4000 Wire Segments 5. 2 Programmable Interconnects All internal connections are made up of metal segments with programmable switching points to implement the routing. There are three main interconnect types: Single Length Lines – a grid of horizontal and vertical lines, which intersect at a switch matrix between each CLB. Double Length Lines – a grid of horizontal and vertical lines, which intersect at the switch matrixes between TWO CLBs. Long Lines – form a grid of metal interconnection segments that run the entire length or width of the array.

Global buffers can drive additional vertical lines designed to distribute clock signals and other high fanout signals The router/fitter software, to take your synthesized design and place it in the desired FPGA part, uses all these elements. 26 FPGAs 5. 3 Altera FLEX 8000 and FLEX 10000 FPGAs Altera’s FLEX 8000 series consists of a three-level hierarchy much like that found in CPLDs. However, the lowest level of the hierarchy consists of a set of lookup tables, rather than an SPLD like block, and so the FLEX 8000 is categorized here as an FPGA. It should be noted, however, that FLEX 8000 is a combination of FPGA and CPLD technologies.

FLEX 8000 is SRAM-based and features a four-input LUT as its basic logic block. Logic capacity ranges from about 4000 gates to more than 15, 000 for the 8000 series. The overall architecture of FLEX 8000 is illustrated in Figure 15. The basic logic block, called a Logic Element (LE) contains a four-input LUT, a flip-flop, and special-purpose carry circuitry for arithmetic circuits (similar to Xilinx XC4000). The LE also includes cascade circuitry that allows for efficient implementation of wide AND functions. Details of the LE are illustrated in Figure 16.

In the FLEX 8000, LEs are grouped into sets of 8, called Logic Array Blocks (LABs, a term borrowed from Altera’s CPLDs). As shown in Figure 17, each LAB contains local interconnect and each local wire can connect any LE to any other LE within the same LAB. Local interconnect also connects to the FLEX 8000’s global interconnect, called FastTrack. FastTrack is similar to Xilinx long lines in that each FastTrack wire extends the full width or height of the device. However, a major difference between FLEX 8000 and Xilinx

chips is that FastTrack consists of only long lines. This makes the FLEX 8000 easy for CAD tools to automatically configure.

All Fast- Track wires horizontal wires are identical, and so interconnect delays in the FLEX 8000 are more predictable than FPGAs that employ many smaller length segments because there are fewer programmable switches in the longer paths. Predictability is furthered aided by the fact that connections between horizontal and vertical lines pass through active buffers. The FLEX 8000 architecture has been extended in the state-of- the-art FLEX 10000 family. FLEX 10000 offers all of the features of FLEX 8000, with the addition of variable-sized blocks of SRAM, called Embedded Array Blocks (EABs).

This idea is illustrated in Figure 18, which shows that each row in a FLEX 10000 chip has an EAB on one end. Each EAB is configurable to serve as an 27 FPGAs Figure 15: Architecture of Altera FLEX 8000 FPGAs Figure 16: Altera FLEX 8000 Logic Element (LE) SRAM block with a variable aspect ratio: 256 x 8, 512 x 4, 1K x 2, or 2K x 1. In addition, an EAB can alternatively be configured to implement a complex logic circuit, such as a multiplier, by employing it as a large multi-output lookup table. Altera provides, as part of their CAD tools, several macro-functions that implement useful logic circuits in EABs. 8 FPGAs Counting the EABs as logic gates, FLEX 10000 offers the highest logic capacity of any FPGA, although it is hard to provide an accurate number. Figure 17: Altera FLEX 8000 Logic Array Block (LAB) 5. 4 AT&T ORCA FPGAs AT&T’s SRAM-based FPGAs feature an overall structure similar to that in Xilinx FPGAs, and is called Optimized Reconfigurable Cell Array (ORCA). The ORCA logic block is based on LUTs, containing an array of

Programmable Function Units (PFUs). The structure of a PFU is shown in Figure 19. A PFU possesses a unique kind of configurability among LUT-based logic blocks, in hat it can be configured in the following ways: as four 4-input LUTs, as two 5-input LUTs, and as one 6-input LUT. A key element of this architecture is that when used as four 4 input LUTs, several of the LUTs' inputs must come from the same PFU input. While this reduces the apparent functionality of the PFU, it also significantly reduces the cost of the wiring associated with the chip. The PFU also includes arithmetic circuitry, like Xilinx XC4000 and Altera FLEX 8000, and like Xilinx XC4000 a PFU can be configured as a RAM block. A recently announced version of the ORCA chip also allows dual-port and synchronous RAM.

ORCA's interconnect structure is also different from those in other SRAM-based FPGAs. 29 FPGAs Figure 18: Architecture of Altera FLEX 10K FPGAs Each PFU connects to interconnect that is configured in four-bit buses. This provides for more efficient support for " system-level" designs, since buses are common in such applications. The ORCA family has been extended in the ORCA 2 series, and offers very high logic capacity up to 40, 000 logic gates. ORCA 2 features a two-level hierarchy of PFUs based on the original ORCA architecture. 30 FPGAs Figure 19: AT&T Programmable Function Unit (PFU) 5. Actel FPGAs In contrast to FPGAs described above, the devices manufactured by Actel are based on antifuse technology. Actel offers three main families: Act 1, Act 2, and Act 3. Although all three generations have similar features, this paper will focus on the most recent devices, since they are apt to be more widely used in the longer term. Unlike the FPGAs described above, Actel devices are based on a structure similar to traditional gate arrays; the

logic blocks are arranged in rows and there are horizontal routing channels between adjacent rows. This architecture is illustrated in Figure 20.

The logic blocks in the Actel devices are relatively small in comparison to the LUT based ones described above, and are based on multiplexers. Figure 20 illustrates the logic block in the Act 3 and shows that it comprises an AND and OR gate that are connected to a multiplexer based circuit block. The multiplexer circuit is arranged such that, in combination with the two logic gates, a very wide range of functions can be realized in a single logic block. About half of the logic blocks in an Act 3 device also contain a flip-flop. 31 FPGAs Figure 20: Structure of Actel FPGAs

As stated above, Actel’s interconnect is organized in horizontal routing channels. The channels consist of wire segments of various lengths with antifuses to connect logic blocks to wire segments or one wire to another. Also, although not shown in Figure 20, Actel chips have vertical wires that overlay the logic blocks, for signal paths that span multiple rows. In terms of speed-performance, it would seem probable that Actel chips are not fully predictable, because the number of antifuses traversed by a signal depends on how the wire segments are allocated during circuit implementation by CAD tools.

However, Actel provides a rich selection of wire segments of different length in each channel and has developed algorithms that guarantee strict limits on the number of antifuses traversed by any two-point connection in a circuit which improves speedperformance significantly. 5. 6 Quicklogic pASIC FPGAs The main competitor for Actel in antifuse-based FPGAs is Quicklogic; whose

has two families of devices, called pASIC and pASIC-2. The pASIC-2 is an enhanced version that has only recently been introduced, and will not be discussed here. The pASIC, as illustrated in Figure 22, has similarities to everal other FPGAs: the overall structure is 32 FPGAs array-based like Xilinx FPGAs, its logic blocks use multiplexers similar to Actel FPGAs, and the interconnect consists of only long- lines like in Altera FLEX 8000. Figure 21: Actel Act 3 Logic Module We note that the pASIC architecture is now independently developed by Cypress as well, but this discussion will focus only on Quicklogic's version of their parts. Quicklogic's antifuse structure, called ViaLink, is illustrated on the left-hand side of Figure 22. It consists of a top layer of metal, an insulating layer of amorphous silicon, and a bottom layer of metal.

When compared to Actel's PLICE antifuse, ViaLink offers a very low onresistance of about 50 ohms (PLICE is about 300 ohms) and a low parasitic capacitance. Figure 22 shows that ViaLink antifuses are present at every crossing of logic block pins and interconnect wires, providing generous connectivity. pASIC's multiplexer-based logic block is depicted in Fig 23. It is more complex than Actel's Logic Module, with more inputs and wide (6-input) AND-gates on the multiplexer select lines. Every logic block also contains flip flop. . Figure 22: Quicklogic (Cypress) Logic Cell 33 FPGAs

Figure 23: Structure of Quicklogic pASIC FPGA 34 FPGAs CHAPTER 6 APPLICATIONS OF FPGAs FPGAs have gained rapid acceptance and growth over the past decade because they can be applied to a very wide range of applications. A list of typical applications includes: random logic, integrating multiple SPLDs, device controllers, communication encoding and filtering,

small to medium sized systems with SRAM blocks, and many more. Other interesting applications of FPGAs are prototyping of designs later to be implemented in gate arrays, and also emulation of entire large hardware systems.

The former of these applications might be possible using only a single large FPGA (which corresponds to a small Gate Array in terms of capacity), and the latter would entail many FPGAs connected by some sort of interconnect; for emulation of hardware, QuickTurn (and others) has developed products that comprise many FPGAs and the necessary software to partition and map circuits. Another promising area for FPGA application, which is only beginning to be developed, is the usage of FPGAs as custom computing machines.

This involves using the programmable parts to “ execute” software, rather than compiling the software for execution on a regular CPU. The reader is referred to the FPGA-Based Custom Computing Workshop (FCCM) held for the last four years and published by the IEEE. However, designs mapped into an FPGA are broken up into logic block-sized pieces and distributed through an area of the FPGA. Depending on the FPGA’s interconnect structure, there may be various delays associated with the interconnections between these logic blocks.

Thus, FPGA performance often depends more upon how CAD tools map circuits into the chip than is the case for CPLDs. Some other advantages are as follows: Prototyping: Many times an FPGA will be used in a prototype system. A small device may be present to allow the designers to change a board’s glue logic more easily during product development and testing. Or a

large device may be included to allow prototyping of a system-on-a-chip design that will eventually find its way into an ASIC. Either way, the basic idea is the same: allow the hardware to be flexible during product development. 5 FPGAs When the product is ready to ship in large quantities, the programmable device will be replaced with a less expensive, though functionally equivalent, hard-wired alternative. Embedded Cores: More and more vendors are selling or giving away their processors and peripherals in a form that is ready to be integrated into a programmable logic-based design. They either recognize the potential for growth in the system-on-a-chip area and want a piece of the royalties or want to promote the use of their particular FPGA by providing libraries of ready-to-use building blocks.

Either way, you will gain with lower system costs and faster time-to-market. Why develop your own hardware when you can Reconfigurable Computing: As mentioned earlier, an SRAM-based programmable device can have its internal design altered on the fly. This practice is known as reconfigurable computing. Though originally proposed in the late 1960’s by a researcher at UCLA, this is still a relatively new field of study. The decades-long delay had mostly to do with a lack of acceptable reconfigurable hardware.

On-the-fly reprogrammable logic chips have only recently reached gate densities making them suitable for anything more than academic research. But the future of reconfigurable computing is bright and it is already finding a niche in high-end communications, military, and intelligence applications. Hardware Emulation: In hardware emulation, the software partitions a large design (millions of gates) into partitions where each partition fits into a single

FPGA chip. It then places and routes the partitions inside each FPGA chip and between the FPGA chips.

The design is finally ready for emulation. The emulated design executes up to a million times faster than simulation. Small changes are handled incrementally across all FPGA chips in the system, limiting re-implementation times to minutes. 36 FPGAs CHAPTER 7 CONCLUSIONS We have presented a survey of field-programmable devices, describing the basic technology that provides the programmability and a description of many of the architectures in the current marketplace. We believe that over time programmable logic will become the dominant form of digital logic design and implementation.

Their ease of access, principally through the low cost of the devices, makes them attractive to small firms and small parts of large companies. The fast manufacturing turn-around they provide is an essential element of success in the market. As architecture and CAD tools of FPGAs improve, compared to Mask-Programmed Gate Arrays and programmable devices will dominate. 37 FPGAs FURTHER READING A reasonable introduction to FPGAs can be found in the book: S. Brown, R. Francis, J. Rose, Z. Vranesic, Field-Programmable Gate Arrays, Kluwer Academic Publishers, May 1992.

A more specific discussion of FPGA architectures can be found in: S. Trimberger, Ed. , Field-Programmable Gate Array Technology, Kluwer Academic Publishers, 1994. J. Rose, A. El Gamal, A. Sangiovanni Vincentelli, “ Architecture of field programmable gate arrays”, Proceedings of the IEEE, vol. 81, no. 7, pp. 10131029, July 1993. S. Brown, J. Rose, “ FPGA and CPLD architectures: A tutorial”, IEEE Design & Test of Computers, vol. 13, no. 2,

pp. 42-57, Summer 1996. S. Trimberger, “ A reprogrammable gate array and applications”, Proceedings of the IEEE, vol. 81, no. 7, pp. 1030- 1056, July 1993. 38