Field Programmable Gate Array (FPGA)

FPGAs combine the architecture of gate arrays with programmability of PLDs. Some of the FPGA real estate is occupied by vendor logic to implement the field programmability feature of the FPGA, and a large portion of the die is for programmable routing.

Henry Selvaraj

The logic block (known also as LC, CLB and etc.) consists of some universal gates, that is gates that can be programmed to represent any function. The connectivity between blocks is programmed via different types of devices, SRAM (static random-access memory), EEPROM, or antifuse.

Henry Selvaraj
Two basic architectures:

- Matrix based: CLBs form islands in a matrix with horizontal and vertical channels. Eg.: Xilinx and QuickLogic
- CLBs form rows separated by routing channels like in a mask-programmable gate array. Eg.: Actel

Advantages of FPGAs

- Replacement of SSI and MSI chips
  (eg. A circuit with 250 TTL7400 SSI chips (1000 NAND gates) can be replaced by a Xilinx3000 series chip.)
- Availability of parts off the shelf
- rapid turnaround
- low risk
- reprogrammability
Limitation of FPGAs - your opportunities

The FPGA-based circuit delay depends on the performance of the design implementation tools.

The mapping of the logic design into the FPGA’s architecture requires sophisticated design implementation (CAD) tools.

The Design cycle

1. entering the design in the form of schematic, netlist, logic expressions or hardware description languages
2. simulating the design for functional verification
3. mapping the design into the FPGA architecture
4. placing and routing the FPGA design

Minimization, technology mapping, placement and routing
5. extracting delay parameters of the routed design
6. resimulating for timing verification
7. generating the FPGA device configuration format
8. configuring or programming the device
9. testing the product for undesirable functional behavior

For every FPFA, the vendor provides design implementation tools to perform steps 3 through 8. Steps 1 and 2 can be performed using simulation software like PALASM, ABEL or VHDL editors/simulators.
Verification

At the design stage, testing is known as *design verification*. Most common means of verification is simulation.

*Formal verification*: proving the properties of a design to guarantee correctness.

Testing

Testing after manufacturing is usually known as *digital testing*. This type of testing detects not only failures due to manufacturing defects but also failures due to incorrect design.
Often, simulation patterns developed for design verification are complemented with patterns that are generated manually or by an automatic test pattern generator (ATPG) to obtain a complete test set, a test that also verifies the functionality of its logic.

*Design for testability:*
Designing with testing in mind.

---

**Synthesis**

Simply put, synthesis is the translation of a design representation to a form that is amendable to minimal realization.

High level synthesis: eg. VHDL to RTL
Logic Synthesis: RTL-gate-level representation
Factors that can influence the design:

- architecture
- gate density
- routing resources
- programming method

Some FPGAs

<table>
<thead>
<tr>
<th>Product</th>
<th>Architecture</th>
<th>Basic Cell</th>
<th>Programming Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actel</td>
<td>Gate Array</td>
<td>MUX</td>
<td>Antifuse</td>
</tr>
<tr>
<td>Algotonix</td>
<td>Sea-of-gates</td>
<td>Functional</td>
<td>SRAM</td>
</tr>
<tr>
<td>Altera</td>
<td>Extended PLA</td>
<td>PLA</td>
<td>EPROM</td>
</tr>
<tr>
<td>Concurrent</td>
<td>Matrix</td>
<td>XOR, AND</td>
<td>SRAM</td>
</tr>
<tr>
<td>Crosspoint</td>
<td>Gate Array</td>
<td>Transistors</td>
<td>Antifuse</td>
</tr>
<tr>
<td>Plessey</td>
<td>Sea-of-gates</td>
<td>NAND</td>
<td>SRAM</td>
</tr>
<tr>
<td>Quick Logic</td>
<td>Matrix</td>
<td>MUX</td>
<td>Antifuse</td>
</tr>
<tr>
<td>Xilinx</td>
<td>Matrix</td>
<td>RAM block</td>
<td>SRAM</td>
</tr>
</tbody>
</table>