VLSI 1 - Notes Week 2

Ruben Schenk, ruben.schenk@inf.ethz.ch

January 10, 2022

1.2.3 The Fabrication Point of View

Field-programmable logic (FPL) uses neither custom layout structures nor proprietary photomasks. Instead, pre-manufactured subcircuits get configured into the target via purely electrical means, such as programmable links.

Compared to mask-programmed ICs:

- Easy and extremely fast to modify
- I/O subcircuits, clock and power distribution, embedded memories, testability, etc. come at no extra effort shut in the component
- Large overhead in terms of area, delay and energy

The basic structure of an FPGA is composed of the following elements:

- Look-up table (LUT) in Logic Block: Performs the logic operations
- Flip-Flop (FF): Performs the storing of the results of the LUT
- Wires: Connect elements to one another, both Logic and clock
- Input/Output pads: These physically available ports get signals in and out of the FPGA

To what extent is a circuit manufactured to user specifications?

- *Full-custom IC:* All fabrication layers, full set of photomasks.
- *Semi-custom IC:* A few metal layers only, subset of photomasks.
- *Field-programmable logic:* Customization occurs electrically, no masks involved.
- *Standard part:* Catalog part with no customization whatsoever.

1.2.4 The Business Point of View

How are the industrial activities shared among business partners?

We look at several players in the semiconductor market:

- *Integrated device manufacturer (IDM):* A chip vendor who operates his own wafer processing facilities.
- *Silicon foundry:* A company that operates a wafer processing line and that offers its manufacturing services to others.
- *Fabless chip vendor:* Develops and markets proprietary semiconductor components but has their manufacturing commissioned to an independent silicon foundry.
- *Fab-lite chip vendor:* Retains just the specialized manufacturing capabilities to integrate sensors, actuators, RF components, or photonic devices a Si substrate along with electronic circuitry.
- *Intellectual property (IP) vendor:* A company that develops hardware subfunctions and licenses them to others for incorporation into their own ICs.
- *System house:* A company that integrates both hardware and software into their products. Hardware is based on microprocessors, memories, ASSPs and FPGAs. USICs are being designed if and only if they provide competitive advantage.

1.2.5 The Design Engineer's Point of View

Which levels of detail are being addressed during a part's design?

- *Hand layout:* Desired geometric shapes are manually drawn to scale. This allows for optimum density, performance and device matching. However, it is slow, cumbersome, and prone to errors.
- *Cell-based design by means of schematic entry:* Manual drawing of cell-level circuits followed by automatic place & route.
  - *Standard cells:* Small universal building blocks, preestablished layout, full characterized, e.g. logic gates, flip-flops, adders, etc.
  - *Megacells:* Much larger size and complexity than standard cells, e.g. microprocessor cores.
  - *Macrocells:* Layout put together on a per-case basis according to specs from a limited collection of layout tiles, e.g. RAM and ROM.

Example: An example of a standard cell would be a 3-input NOR gate:

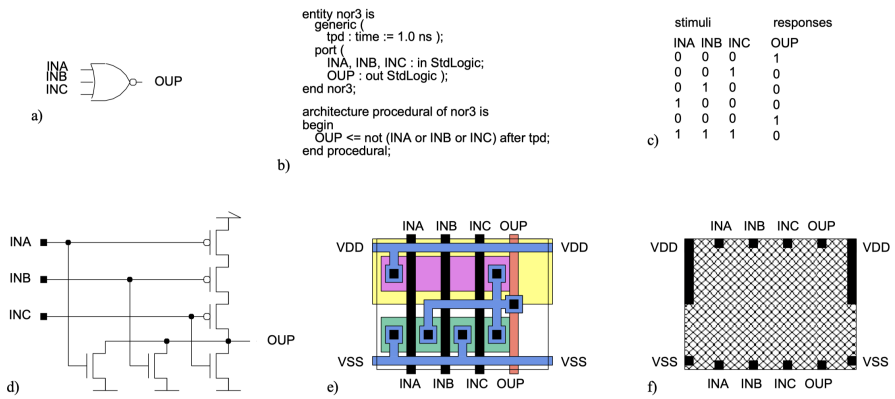

Figure 1: 3-inpput NOR gate

Figure 2.1: Icon (a), simulation model (b), test vector set (c), transistor-level schematic (d), detailed layout (e), and cell abstract (f).

Automatic circuit synthesis is based on the following processes:

- Logic synthesis: Accepts logic equations, truth tables, and state graphs. Generates a gate-level netlists for combinatorial logic and for finite state machines (FSM). This whole process is absorbed in today's EDA flows.
- Register transfer level (RTL) synthesis: The circuit is viewed as a network of storage elements, such as registers and also RAMs, held together by combinational logic. Behavioral specs are allowed to include arithmetic functions, string operations, arrays, enumerated types, etc.

Architecture synthesis starts from a purely algorithmic description. Source code includes no explicit indications for how to marshal data processing operations and hardware resource. It follows 5 steps:

1. Identify the computational and storage requirements.
2. Select a suitable building block for each processing and storage operation.
3. Establish a cycle-based schedule for carrying out the algorithm.
4. Decide on a hardware organization able to execute the resulting work plan.
5. Translate the data moves and operations for each clock cycle into the necessary instruction for RTL synthesis.

Design with virtual components (VCs) works based on HDL synthesis packages made available to others on a commercial basis. In contrast to standard cells, macro- and megacells, which are process-specific (*hard modules*), VCs are portable across fabrication technologies (*soft modules*).

In summary, we have seen the following options for capturing a design with EDA tools:

- Hand layout: Confined to niches.
- Schematic entry: Important at its time, largely confined to analog circuit design today.
- RTL synthesis: VHDL or SystemVerilog code is automatically translated to gate-level netlist. Universally adopted.
- Architecture synthesis: SW code is automatically translated to an RTL synthesis model.
- Incorporation of VCs: Purchased HDL code is automatically translated to gate-level netlist. Routine practice today.

Multiple of the above-mentioned approaches are usually combined!

1.3 The VLSI Design Flow

1.3.1 Major Stages in Digital VLSI Design

At the stage of **system-level design,** the decisions take determine the final outcome more than anything else:

- Specify the functionality and characteristics of the systems
- Partition the system's functionality into subtasks
- Explore alternative hardware and software tradeoffs
- Decide on interfaces and protocols for data exchange
- Define, model, evaluate, and refine the various subtasks

The final result after the above steps is the **system-level model.**

The next step is the **algorithm design** which is concerned with streamlining computations in view of their implementation in hardware:

- Cut down computational burden and memory requirements
- Find compromises between computational complexity and accuracy
- Contain effects due to finite word-length computation
- Quantify the minimum required computational resources

The result of the algorithm design is the **bit-true software model.**

The **architecture design** takes important high-level decisions:

- Partition a computational task in view of hardware realization
- Organize the interplay of the various subtasks
- Define datapaths and controllers
- Decide on a circuit style and fabrication process

The result of the architecture design is a **high-level block diagram and preliminary floorplan.**

The **physical design** consists of the following steps:

1. Floorplanning
2. Padframe generation and power distribution
3. Initial placement of cells
4. Clock tree insertion
5. Detailed routing
6. Chip assembly
7. Substitution of detailed layout for cell abstracts

The result is *polygon layout data for mask preparation.*

Prior to fabrication, all layout data needs to be checked to protect against fatal mishaps. The set of available instruments includes:

- Check conformity of layout with geometric rules
- Search for patters likely to be detrimental to yield
- Layout extraction to obtain the actual circuit netlist
- Post-layout simulation

1.3.3 FPGA

Reconfigurable computing describes computer architecture combining some flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like **field-programmable gate array (FPGAs).**

- The main difference compared to *ordinary microprocessors* is the ability to make substantial changes to the datapath itself in addition to the control flow.

- The main difference compared to *custom hardware* is the possibility to adapt new hardware during runtime by "loading" a new circuit on the reconfigurable fabric.

The basic structure of an FPGA is composed of the following components:

- Look-up table (LUT): This element performs logic operations.
- Flip-Flop (FF): This register element stores the result of the LUT.
- Wires: These elements connect elements to one another, both logic and clock.
- Input/Output (I/O) pads: These physically available ports get signals in and out of the FPGA.

How can we implement any circuit in an FPGA? Combinational logic is represented by a truth table. Truth tables are implemented in small memories (LUTs). The input values are used to address the LUT and retrieve the value of the function corresponding to the input values.

A LUT is basically a multiplexer that evaluates the truth table stored in the configuration SRAM cells. Sequential logic is handled by adding a flip-flop to the output LUT, i.e. a clocked storage element. This is called a *basic logic element (BLE).*

Example: Consider this simplified Xilinx Configurable Logic Block (CLB):

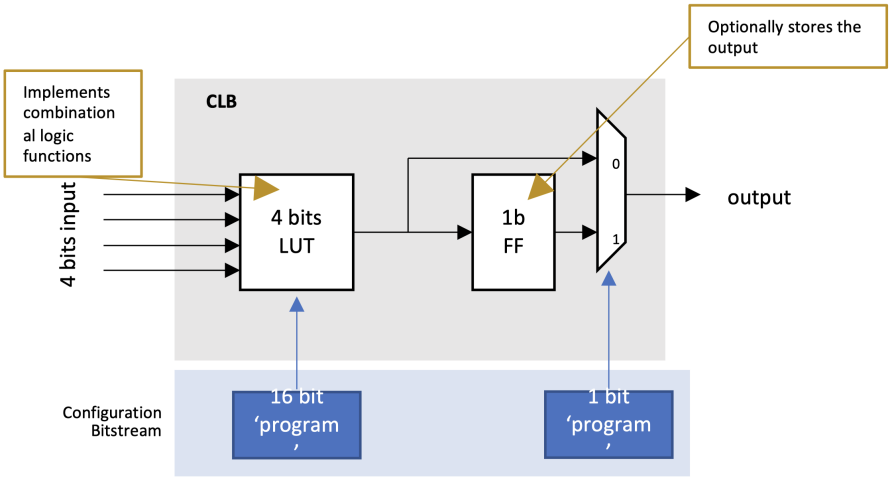

Figure 2: Xilinix Configurable Logic Block

Before an FPGA is programmed, it doesn't know which CLBs will be connected. Connections are designed dependent, so there are wires everywhere, both for data and the clock! CLBs are typically arranged in a grid, with wires on all sides. To connect CLBs to wires, *connection boxes* are used: these blocks allow inputs and outputs of CLBs to connect to different wires. Inside a connection box, there are configurable switches.

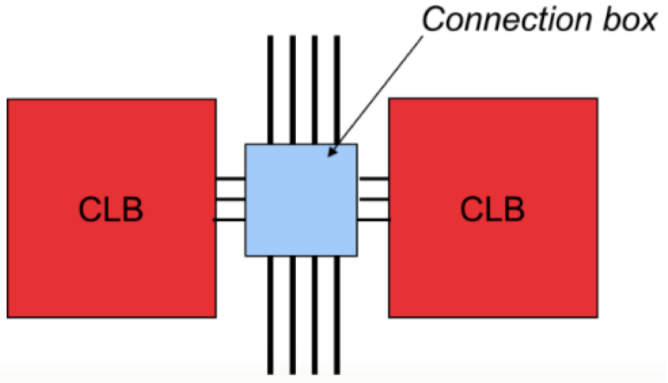

Figure 3: Connection Box

In those switch boxes (connection boxes) there are *short channels* (useful for connecting adjacent CLBs) and *long channels* (useful for connecting separated CLBS), this reduces routing delay for non-adjacent CLBs.