Going from VHDL to implementation

Using Xilinx Vivado

Examples

- LEDs - Simple design
  - No clock
  - No ARM
- Microphone - Slightly more complex
  - Clock different from ARM
  - ARM

What is Xilinx Vivado?

- Synthesis, place and route tool for Xilinx FPGAs
- Abysmal space requirement of ~36 GB
- Does not patch
- Slow
- The more intuitive tool

The three ways of FPGA design

- HDLs (VHDL, Verilog etc)
- HLS (pragmated C or OpenCL)
  - HDL
  - IP core
  - Designated ‘compute’ area (GPGPU like)
- IP cores

1. Create project

- Choose type
  - RTL
  - Post synthesis
  - IO planner
- Choose part / board
  - Vivado is board aware - choose board if present

2. Add sources (if any)

- HDL files
  - For SME projects - VHDL files
- Constraints
  - For the Pynq boards - vendor supplied constraints file

3. Create block design

- Open block design
- Add IP cores (if needed)
  - For SME add module -> choose _export module
- Make IP ports external (if needed)
- Add Zynq IP core
- Auto configure Zynq block
- Save block design
- Create HDL wrapper
- Set wrapper as top
- File -> export block design

4. Open elaborated design

- Parses HDL files
- Can produce estimation reports
- View the parsed and inferred schematic
- Do I/O planning

5. Synthesis

- Maps high level HDL to low level HDL
- Can give additional estimation reports
  - Power consumption
  - Utilization
- Reports on what Vivado was able to infer
  - Note: especially DSP, RAM, ROM, State machines, PLL/MMCM and Flip flops are important
- View schematic of low level HDL
- Clocking wizard
  - Specify clocks and their target clock rate
  - I/O delay
- Synthesis can be time consuming if inference failed

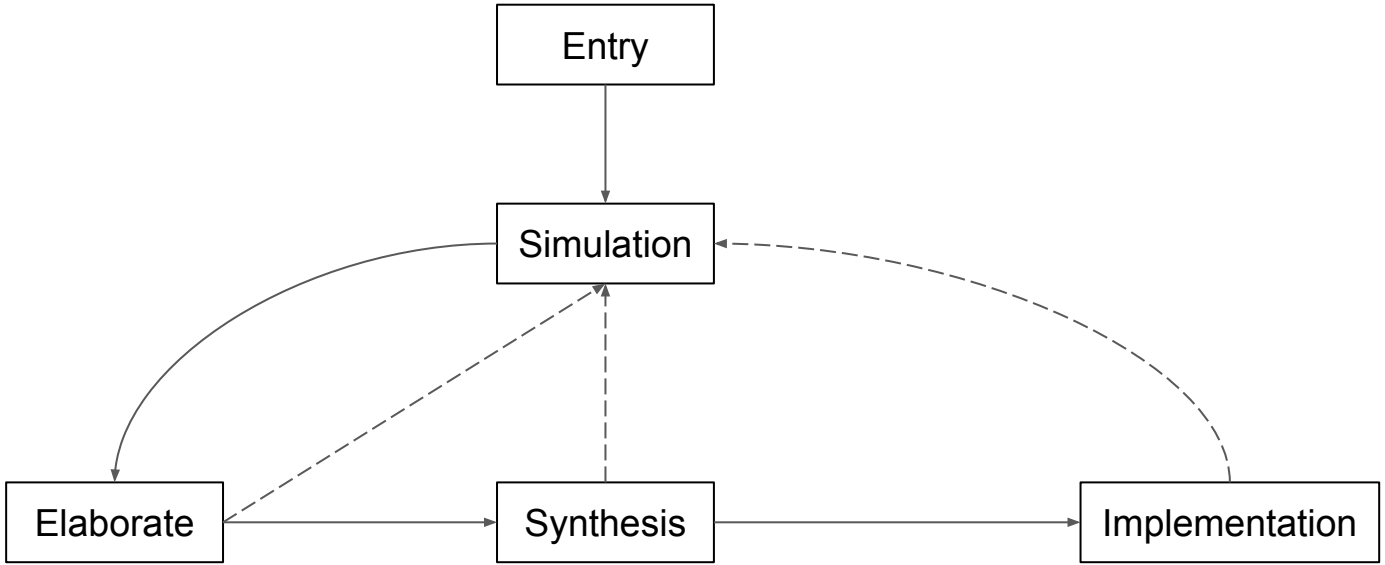
6. Implementation

- Maps the low level HDL to on board components (NP hard)
- Routes the connections between the components (NP hard)
- Can produce final reports on the actual implementation
  - Timing
  - Utilization
  - Power
- View design schematic
- View board schematic

0-6. Simulation

- This is generally done before, during and after the entire process
- There are three points of simulation
  - Behavorial (this is what GHDL does)
  - Post synthesis functional / timing
  - Post implementation functional / timing
- Each stage increases complexity
- Simulations are done by either
  - Forcing signal drivers by ‘hand’
  - Writing testbenches (which SME produces)

Design flowchart



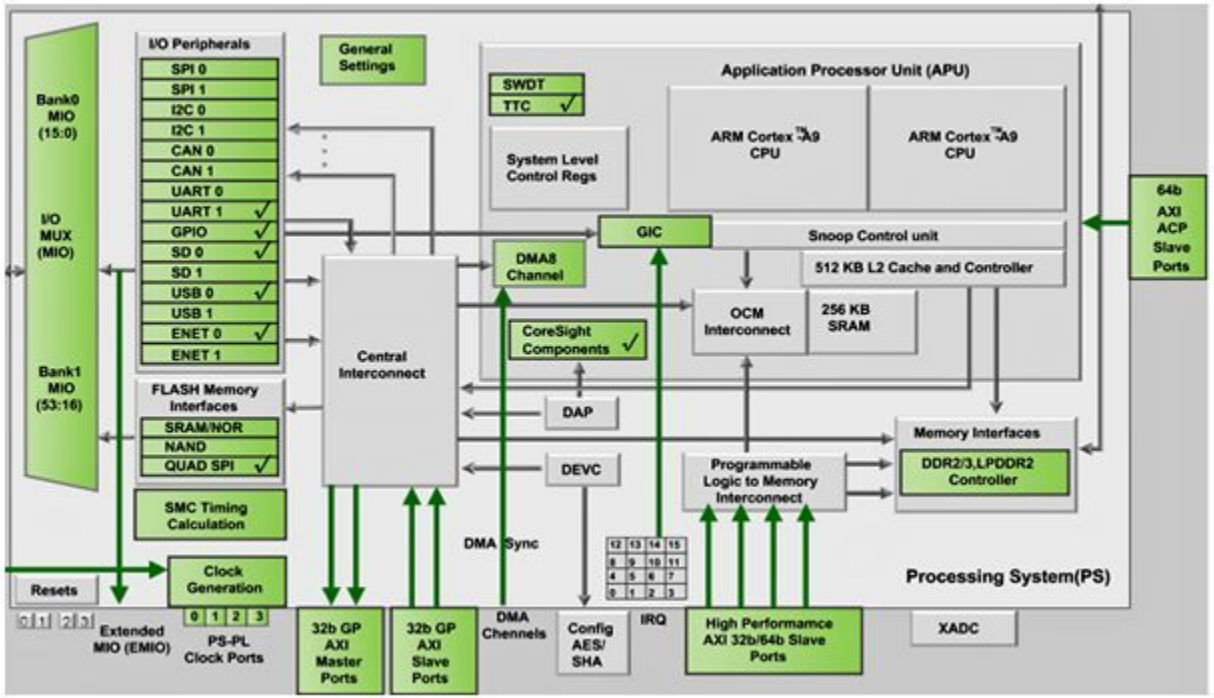
7. Write bitstream

- Produces the binary file
- They can be written to the FPGA in three ways:
  - Using a JTAG cable (micro USB on the Pynq)
  - At runtime from the operating system (as the Pynq does)
  - At powerup from an SD card
- To use the bitstream on the Pynq, the .bit file should be transferred along with the exported block design

Communicating with the ARM

- Additional steps are required:
- AXI interface
- Clocking
- Reset synchronization

Zynq block diagram



AXI interface

- AXI is an now open source protocol for communication
- Comes in two variants
  - Memory mapped
  - Streaming
- Vivado can produce a template project for creating an AXI IP core
  - It has the AXI interface implemented in HDLs
  - We just need to add our logic behind it
    - Note: dual drivers is not allowed in hardware, only one can write at the time

Creating the entire design

- Add the custom AXI IP core
- Add the Zynq IP core
- Auto configure the Zynq IP core
- Auto connect the custom AXI IP core to the Zynq IP core
  - Note: if your design requires a clock different from the 100 MHz supplied by the Zynq, select a new clock wizard as clock source for the AXI slave
  - Note Note: this is only if your design shares the same clock as the AXI interface
  - This produces two additional IP cores
    - AXI Interconnect
    - Reset synchronization
- Make all the pins which the ARM does not use external (E.g. the PMOD)

Final steps

- Same as before
  - Save block design
  - Export block design
  - Create HDL wrapper
  - Map I/O ports
  - Synthesis
  - Implementation
  - Write bitstream
- Note: for designs using the ARM processor, the exported block design contains the address where the AXI interface is mapped

Final notes

- Not all FPGAs carry a ‘host’ CPU
- The ‘host’ CPU does not always run an operating system
  - It can still be programmed by injecting programs directly into the L1 cache