Observations, recommendations…

… and personal opinions …

Tools and things

- It’s the tools ! Most often. In the past two decades anyway…
  - Various flavours (ISE/PlanAhead/Vivado)
  - Versions (10.x-14.x),
  - 32bit / 64bit (not all run on all computers)
  - Bugs

- Read that document !
  - Xilinx document navigator !!!
  - Explore the web page
  - Community pages

- Don’t ignore that message !
  - Plenty of warnings, errors
  - Sometimes misguiding, but…

System level / board level considerations

Partition your system and boards appropriately:

- Asynchronous, simple code with predictable I/O timing
- Circuitry that needs to be operational just after power-up
  → consider use of a CPLD

- Slow control code
  → consider use of a micro controller, or…
  → see embedded course

- Highly parallel, low latency, pipelined code ideally suited for implementation in FPGAs

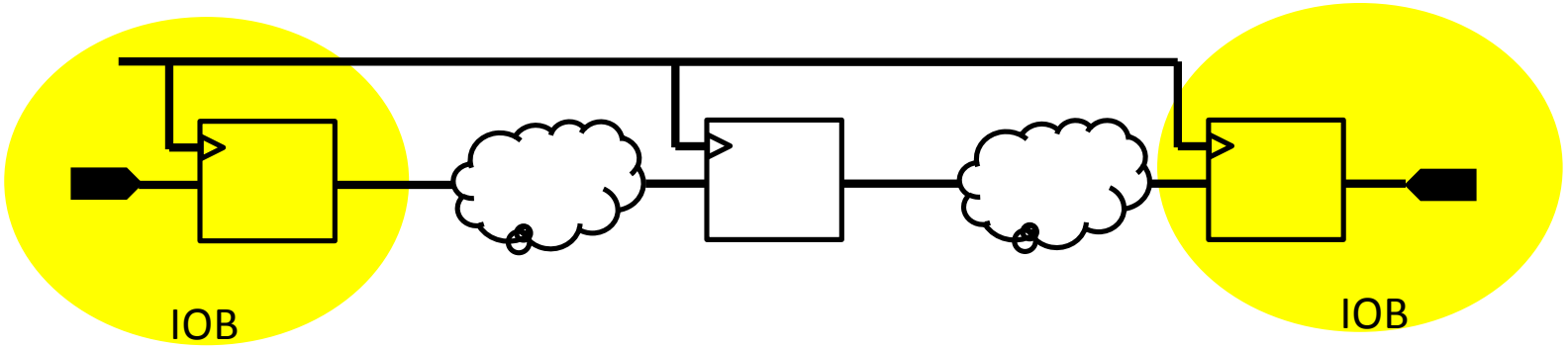
- And occasionally, fixed function off-the-shelf components might still be the most effective choice…

VHDL coding for FPGAs

- Code any test bench as you like
- Not all constructs are synthesizable
- May wish to initialize signals and variables to allow for simulation
- Use of concurrent code allows better control of synthesis results
- Particularly important for low latency designs
- If you want to minimize risk of generating inefficient, dysfunctional, not synthesizable code, prefer clocked processes
- For synchronous code (see below) functional simulation will be very effective for validating your design (see below)
- Some designs might require careful timing simulation (post place and route)

Synchronous designs

- FPGAs generally unsuitable for asynchronous designs (with few exceptions / Achronix)
- Make sure designs are fully synchronous, pipelined (avoid even asynchronous set/reset of flip-flops, if possible)
- Have all data paths starting and ending in an I/O flip-flop
- Intersperse combinatorial logic with flip-flops as needed, to arrive at required clock frequency
- Try to keep # of clocks to minimum
- Require synchronization scheme where crossing clock domains
- For latency critical designs
  - Keep # pipeline stages as low as possible
  - Make sure the design tool is not inferring shift registers rather than flip-flops



Attributes

- The implementation flow can be controlled by global settings (in the ISE GUI) and by attributes
- Attributes are set within the VHDL code

- Two examples of attributes attached to signals that were declared somewhere above:

  - Force a flip-flop into an IOB:

```
attribute IOB : string;
attribute IOB of output_signal : signal is "true";
```

  - Prevent shift registers from being inferred:

```
attribute shreg_extract : string;
attribute shreg_extract of internal_signal: signal is "no";
```

Constraints

.ucf files for Xilinx ISE tools

.xdc for Vivado

Timing constraints

- For fully synchronous designs specification of clock frequency is generally all that's needed
- With a design verified at functional level, and with correct timing constraints, your implementation will generally yield predictable results

Pin constraints

- Make sure all pins found in port of top level entity are constrained
- Make sure I/O bank voltages and logic level of signals are properly constrained

- Constraints can alternatively be set via VHDL attributes