Intel® FPGA Technical Training

Designing for High Productivity with Intel FPGAs

Course Description

FPGA engineers face many challenges while developing with Intel Stratix, Arria families and even with Cyclone family with high utilization (80-90%), such as long compilation time, performance variation due to small/medium changes in the design and resource utilization.

This course provides all practical know-how needed to achieve higher productivity in Intel FPGAs.

The course provides practical tools and design methods for engineers in order to increase their productivity by finish their projects quicker with best results.

Course Duration

2 days

Goals

1. Understand resource utilization reports for Intel FPGAs families
2. Design efficient HDL code for the best resource utilization
3. Decrease Quartus compilation time
4. Use incremental compilation methodology to maintain performance, reduce compilation time, and solve timing issues

Intended Users

Digital hardware engineers and FPGA team leaders, who would like to enhance their FPGA skills and achieve higher productivity in their FPGA design

Previous Knowledge

Quartus Prime software

VHDL/Verilog

TimeQuest

FPGA design experience

Course Material

1. Synthesizer and Place & Route: Quartus Prime
2. Course book (including labs)

Table of Contents

Day #1

- **Introduction to high productivity**
  - Project Management Phases
  - Project goals challenges
    - Resources utilization
    - I/O timing
    - Inter-chip timing
    - Debug
    - Power consumption
    - Reuse
    - Third party IP
    - Engineering resources
    - Communication between teams
    - Design tools version
  - Reverse engineering and security

- **Understanding Resource Utilization Report**
  - Synthesis resource utilization report
  - Utilization by entity report
  - Fitter resource utilization report
  - Resource utilization in terms of ALMs

        - ALMs needed
        - ALMs used in final placement
        - Estimate of ALMs recoverable by dense packing
        - Estimate of ALMs unavailable
        - ALMs used for memory
    - Resource utilization challenges
    - Resource utilization optimization reports
        - Registers removed during synthesis
        - General register statistics
        - Inverted register statistics
        - Synthesis attributes effect on optimization reports
    - Resource utilization use cases
    - Resource optimization advisor
    - I/O assignment analysis
    - Using I/O Flip Flops
    - Using useioff attribute
    - Optimize source code guidelines
    - Optimize synthesis for area globally
    - Optimize synthesis for area using the assignment editor
    - When to use Restructure multiplexers optimization
    - When to use Register packing optimization
    - Preserve register and keep logic attributes
    - Maximum fanout attribute
    - Remove fitter constraints
    - Viewing routing congestion
    - Flatten the hierarchy during synthesis
    - Retargeting memory blocks
    - How to use efficiently physical synthesis to reduce area
    - Retargeting/balancing DSP blocks
    - Limiting the number of DSP blocks
    - FSM processing and safe FSM

- **Designing for Best Area Utilization**
    - Derivation of efficient HDL description
    - Resource sharing definition
    - Operator sharing
    - Operator sharing examples
    - Operator sharing in Quartus Prime
    - Automatic operator sharing limitation

- Balancing operators
- Multipliers balancing
- Counter efficient design
- Functionality sharing definition
- Functionality sharing examples

❖ **Lab #1 : Apply resource optimization techniques to achieve the smallest design area**

- **Reducing Compilation Time**
  - Compilation time challenges
  - Compilation time advisor
  - Rapid recompilation
  - Smart compilation
  - Parallel compilation with multicore host
  - Incremental compilation concept
  - Tips to reduce synthesis time
  - Tips to reduce placement time
  - Tips to reduce routing time
  - Tips to reduce static timing analysis time

❖ **Lab #2 : Apply Synthesis Attributes to Control Synthesis Results and Reduce Compilation Time**

Day #2

- **An Introduction to Incremental Compilation**
  - Top-down design flow
  - Single project design flow issues
  - Team based design flow issues
  - Incremental compilation definition
  - Compilation flow
  - How incremental compilation works
  - Considering FPGA design trade-offs
  - Setting expectations
  - When not to use incremental compilation
  - Planning considerations

- **Design Partitions**
  - What are design partitions?
  - Partition recommendations
  - Design planning
  - Design guidelines

- **Design Partition Tools & Interface**
  - Creating design partitions
  - Design partitions window
  - Partition netlist types
  - Fitter preservation levels
  - Design partition properties
  - Assessing partition quality
  - Design partition planner
  - Chip planner

- **Design Partition Tips and Techniques**
  - Quick multi-partition top-level file
  - Creating black-box wrapper files
  - Use empty partition resources for debugging
  - Fast compiles with SignalTap II
  - Partition I/O interfaces (DDR3, PCIe,RapidIO)
  - Timing closure with incremental compilation
  - Hierarchy isolation
  - Strategy for maximizing performance

❖ **Lab#3: Get familiar with design partitions and incremental compilation tools**
❖ **Lab#4: Use incremental compilation to improve design performance**