

Trigger design engineering tools

# Data flow analysis

- Data flow analysis through the entire Trigger Processor allow us to refine the optimal architecture.
- The optimal architecture is constrained by
  - the Pixel Detector architecture and readout
  - the segment finder algorithm
  - the track and vertex finder algorithm
- Type of constrains
  - physical layout
  - data flow
  - electrical
  - monetary

# Data flow analysis tools

- Queuing theory: provides a measure of data throughput and efficiency in the system. Allow us to calculate buffer sizes and eliminate bottlenecks.
  - 1st two statistical moments of the input distributions
  - computing time of processing nodes
    - algorithms
    - communication nets

# Data flow analysis tools (cont.)

- Data flow simulation (Matlab) : performs verification of the parameters estimated in the queuing analysis. Runs “real” events instead of probabilistic moments
  - Simulation files come from physic simulations.
  - Event processing and communication times are the same as in queuing analysis.
- Simulation results provide
  - a measure of the bandwidth
  - a measure of the efficiency
  - buffer sizes
  - bottleneck detection
- Down side of data flow simulations
  - still to far away from implementation. (Too high level behavioral)

# Data flow analysis tools (cont.)

- Why Matlab?
  - Very good data analysis package.
  - Excellent GUI.
  - communicates with C and C++.
  - Most widely math oriented simulation package in the university world.
  - ESE/CD has one license.
- Matlab output files can be used with other data analysis packages.

# Implementation tools

- The main components in the implementation of the Trigger Processor are FPGAs and DSPs.
- The main tools to be used are
  - FPGAs:
    - VHDL simulators and compilers.
    - Synthesis tools.
    - Fitters and routers (FPGA specific).
  - DSPs:
    - Simulators.
    - Compilers.
    - Assemblers.
    - Profilers
    - Profiled based compilation (TI compiler feature).

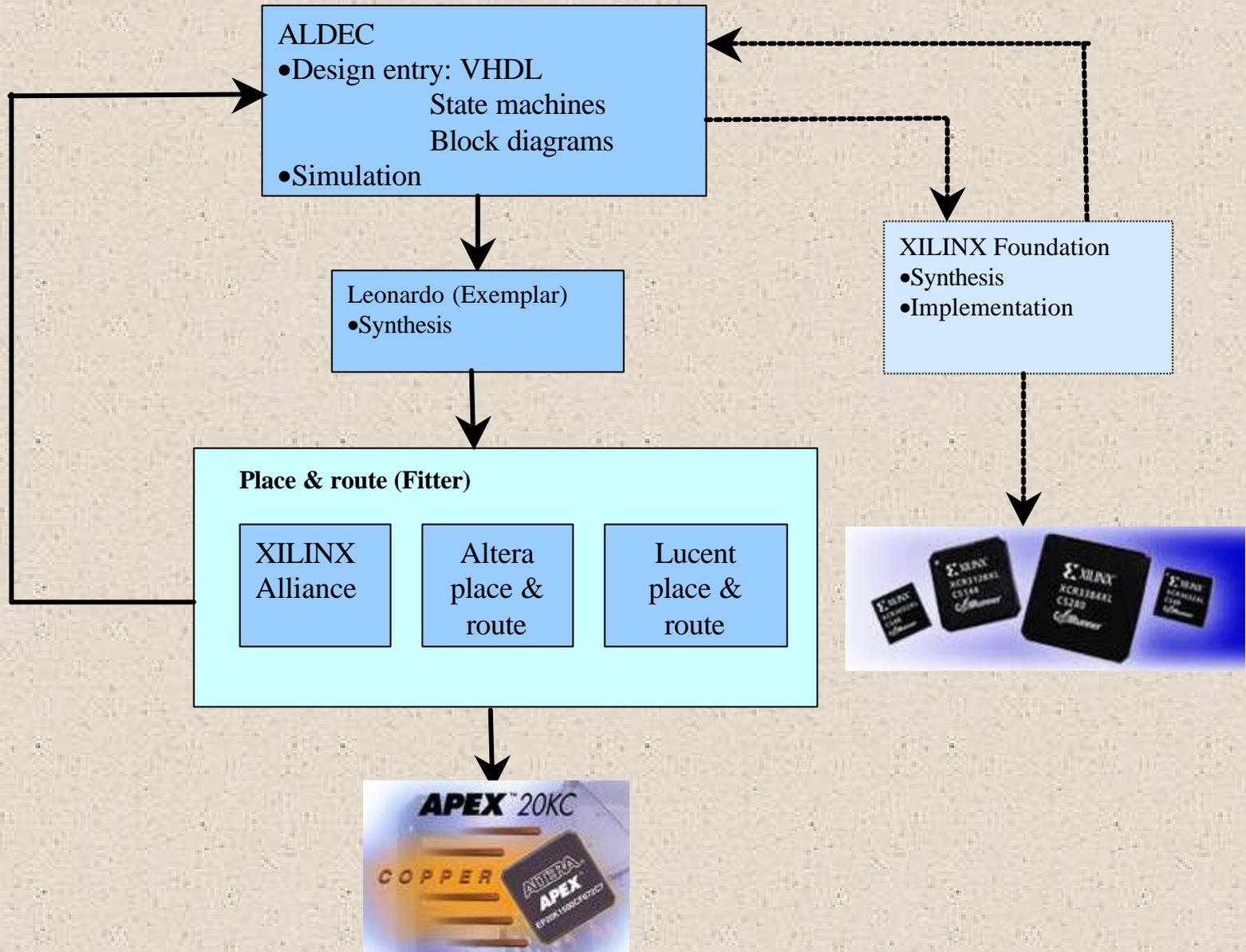
# VHDL simulators and compilers

- VHDL is a hardware description language.
  - The VHDL compiler generates a netlist of logic equations that will be minimized by the synthesizer and later on converted to a “fusemap” by the router and fitter.
- Lower level behavioral simulation
  - good to cross validate parameters in the queuing analysis and behavior simulation.
  - starting point to generate the FPGA firmware.
- VHDL simulation.
  - VHDL simulators allow the use of typical gate and register delays
  - timing parameters can be feedback from FPGA fitters.
- VHDL implementation
  - ALTERA, XILINX, Lucent, etc

# VHDL simulators and compilers (cont.)

- **Brand-specific software**
  - Quartus (AHDL for Altera parts)
    - VHDL and Verilog is supported
  - Xilinx Foundation
    - VHDL and Verilog is supported
- **Brand-independent VHDL software**
  - Many sources. There are commercial and public domain packages.
  - VHDL code is very flexible.
  - No code is 100% portable.
  - Code may contain synthesis directives.
  - VHDL code independent of target device.
  - ALDEC: Currently used VHDL package at ESE/CD

# VHDL design



# DSP's

- Texas Instrument DSP's
  - announcing new series TMS320C64x
    - 8 times the performance of TMS320C62x/7x
    - 600-1100 MHz
    - 4800-8800 MIPs
    - 2400-4400 16bit MACs
    - Code size reduction of ~25%
- Analog Devices
  - SHARC
    - Tiger SHARC:
      - 5000 MIPs
      - 120 MFLOPs
  - New Analog Devices/Intel DSP