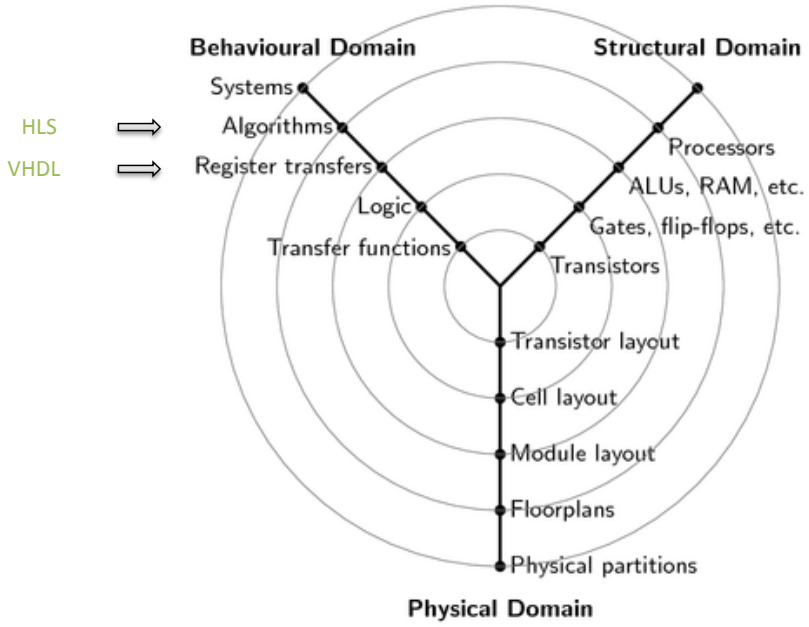
ECE3270

Information for these slides taken from Altera OpenCL documentation and learning materials

INTRODUCTION TO OPENCL AND LIBRARIES

Gajski-Kuhn Y-chart



High Level Synthesis (HLS)

- **High Level Synthesis** is a more productive method for implementing algorithms in hardware (FPGAs in our case), as compared to **RTL** design.
- Even with HLS, it requires more effort to implement algorithms on FPGAs than on a traditional CPU.
- So, then why would we want to implement our algorithms on an FPGA ?

Heterogeneous Computing Systems

CLEMSON UNIVERSITY

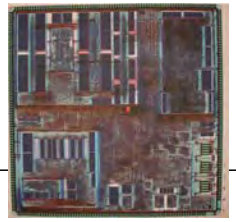
- Heterogeneous computing refers to systems that use more than one kind of processor

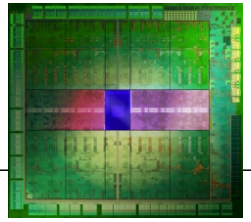
- These are systems that gain performance not just by adding the same type of processors, but by adding dissimilar processors
  - Usually **incorporating specialized processing capabilities to handle particular tasks** or **applications**

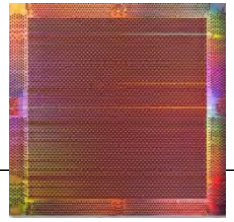
Example Heterogeneous Computing Devices

- Multi-core, general purpose, central processing units (CPUs)
  - include multiple execution units ("cores") on the same chip
- Digital Signal Processing (DSPs) processors
  - optimized for the operational needs of digital signal processing
- Graphics Processing units (GPUs)
  - heavily optimized for computer graphics processing
- Field Programmable Gate Arrays (FPGAs)
  - custom architectures for each problem being solved.


CPUs


DSPs


GPUs

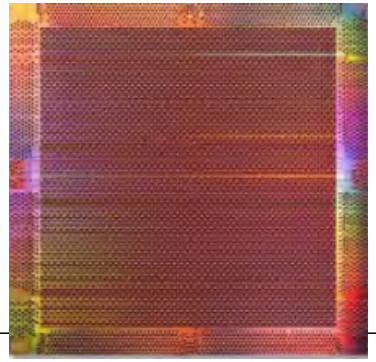

FPGAs

Traditional Approach to Heterogeneous Computing

- Write sequential software for CPU and DSP
  - Advanced superscalar processors could achieve instruction-level parallelism

and

- Design parallel hardware for FPGA
  - Write HDL
  - Fine-grained parallelism
  - Vectorization



Programmers Dilemma

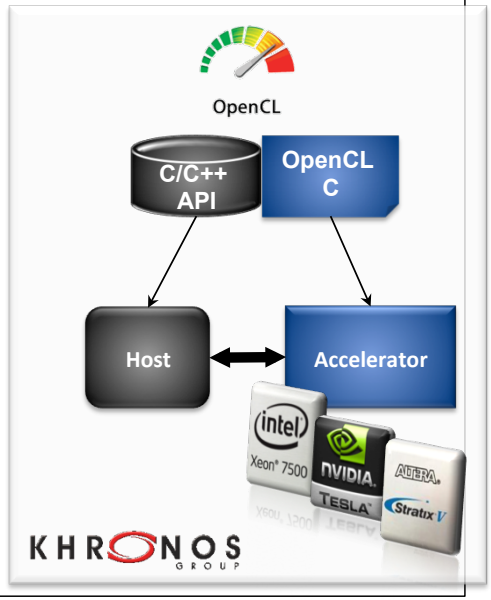
CLEMSON UNIVERSITY

"The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things. I mean, two yeah; four not really; eight, forget it."

~ Steve Jobs, 1955-2011

What is OpenCL?

- A software programming model for software engineers and a software methodology for system architects
  - First industry standard for heterogeneous computing
- Provides increased performance with hardware acceleration
  - Low Level Programming language
  - Based on ANSI C99
- Open, royalty-free, standard
  - Managed by Khronos Group
  - Altera active member
  - Conformance requirements
    - V1.0 is current reference
    - V2.0 is current release
  - http://www.khronos.org



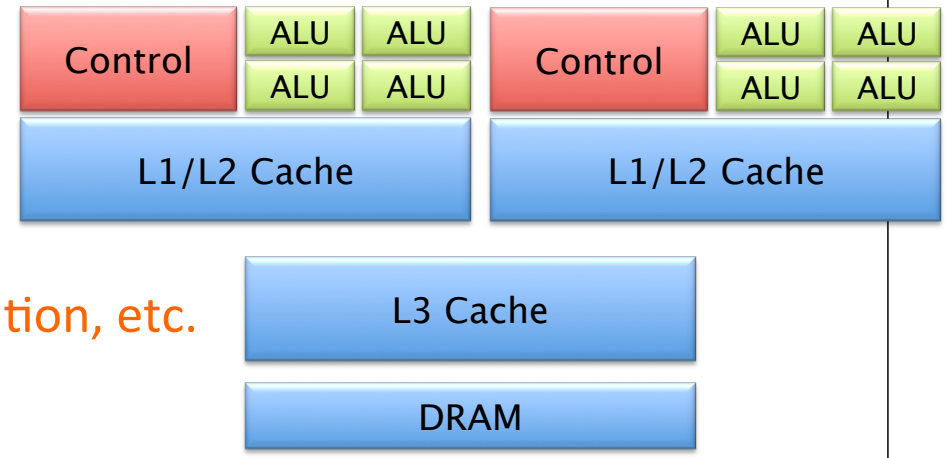
OpenCL Constructs

CLEMSON UNIVERSITY

- OpenCL standard provides abstract models
  - Generic: able to be mapped on to significantly different architectures
  - Flexible: able to extract high performance from every architecture
  - Portable: vendor and device independent

High-Level CPU / DSP Architectures

- Optimized for latency
  - Large caches, HW prefetch
- Complicated control
  - Superscalar, out-of-order execution, etc.
- Comparatively few execution units
  - Vector units (e.g. Intel SSE/AVX, ARM® NEON)



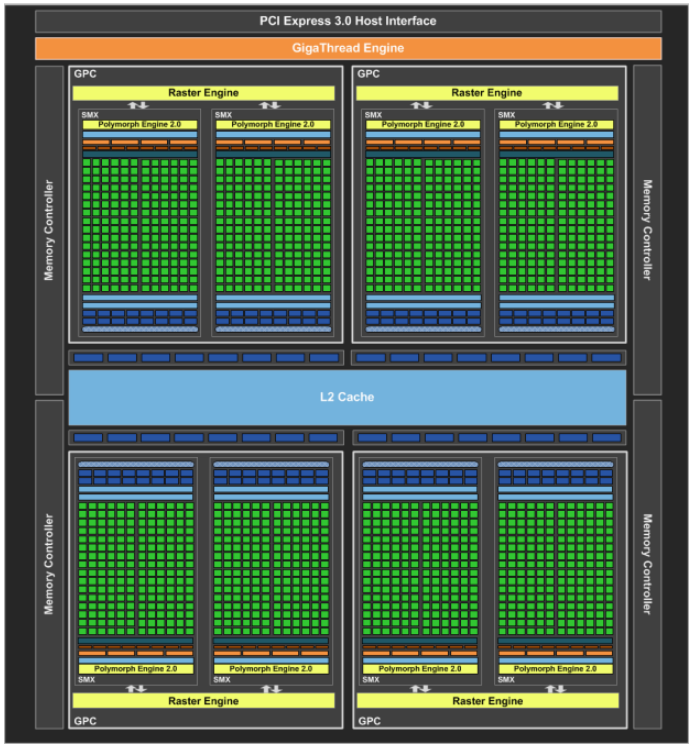
Compiling OpenCL Standard To CPUs

CLEMSON UNIVERSITY

- Execute different workgroups across cores
- Potentially 'fuse' work-items together to execute on vector units
  - Or 'vectorize' kernels to work with explicit vector types
- Synchronization between work-items is handled entirely in software
- No dedicated hardware for sharing data between work-items
  - Rely on caches instead

High-Level GPU Architectures

- Multiple Compute Units
  - "Cores"
  - Vector-like execution
  - Each cores with dedicated resources
    - Registers
    - Local memory/L1 cache
    - Hardware synchronization
- Wide memory bus
  - High bandwidth
  - High latency (800 cycles!)
- Small read/write caches
- PCIe® board



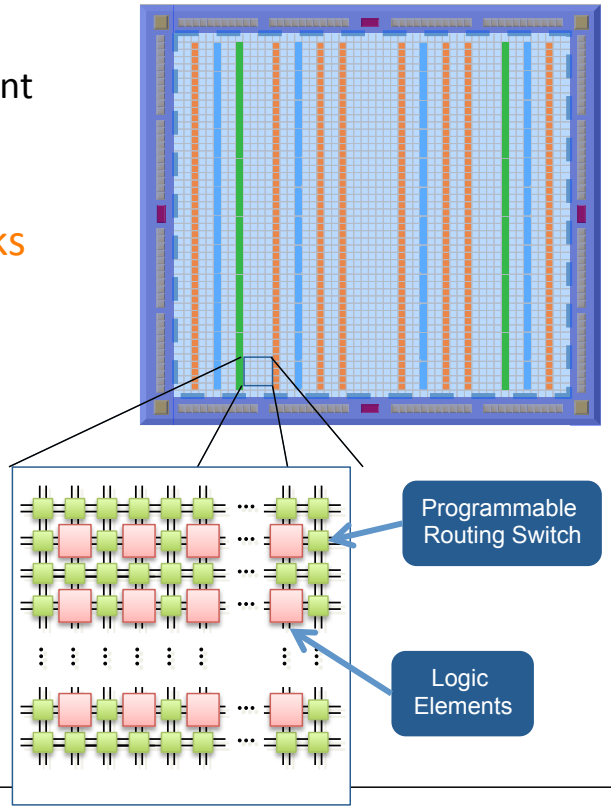
Compiling OpenCL Standard to GPUs

CLEMSON UNIVERSITY

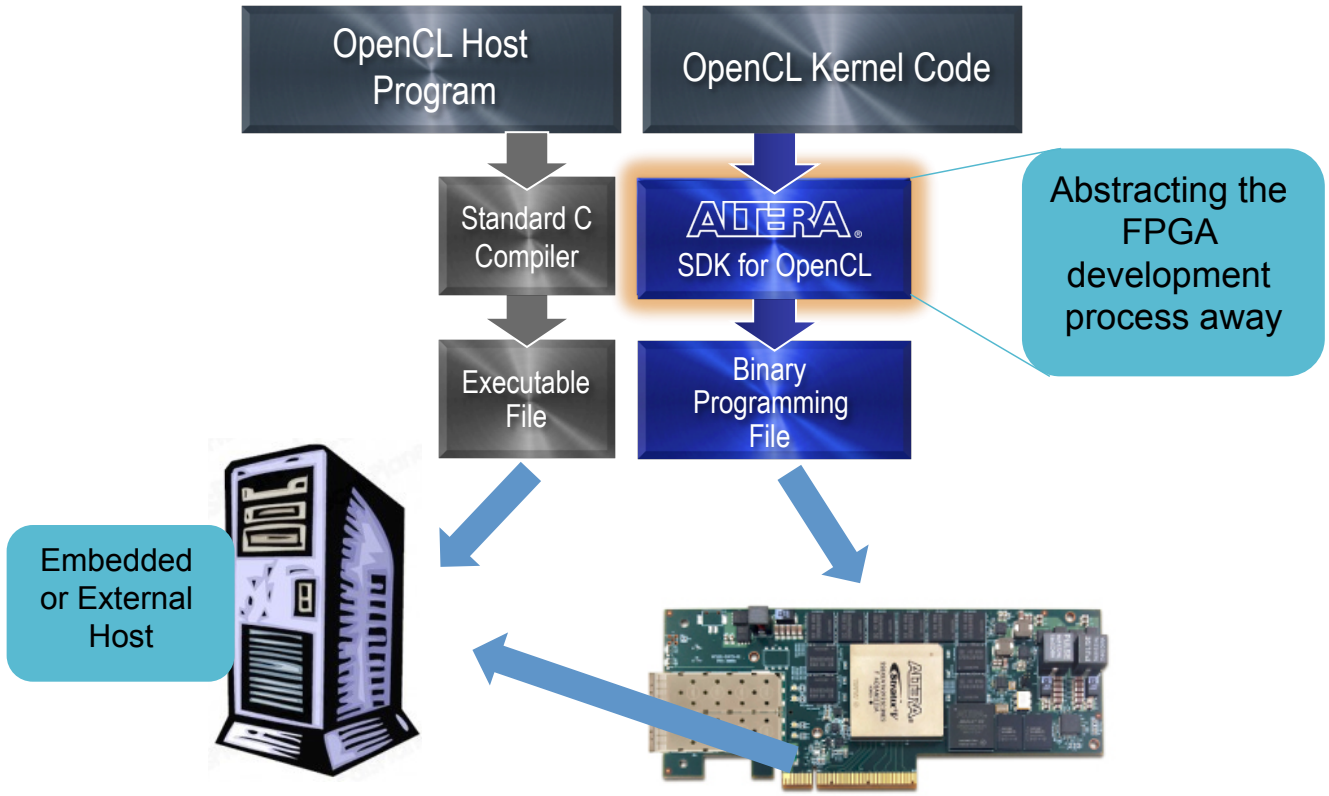
- Distribute workgroups across compute-units
- Work-items execute in parallel on the vector-like cores
- Use dedicated hardware resources for efficient synchronization of work-items and sharing data between work-items
- Run 'enough' work-items per compute-unit to mask latencies
  - Work-items contend for fixed resources (registers, local memory)
  - Hardware limits too!
    - Number of work-items/workgroups per compute unit
  - Translates into a requirement for thousands of work-items!

FPGA Architecture

- Massive Parallelism
  - Millions of logic elements
    - Each include a lookup table able to implement a custom function and multiple registers
  - Thousands of 20Kb memory blocks
  - Thousands of Variable Precision DSP blocks
  - Dozens of High-speed transceivers
  - Various built-in hardened IP
- FPGA Advantages
  - Custom hardware
  - Efficient processing
  - Low power
  - Ability to reconfigure
  - Fast time-to-market

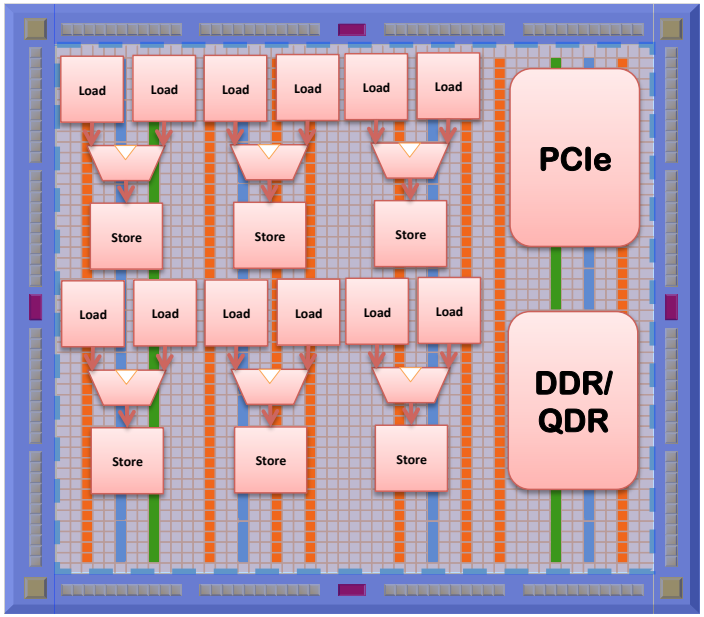


Altera SDK for OpenCL

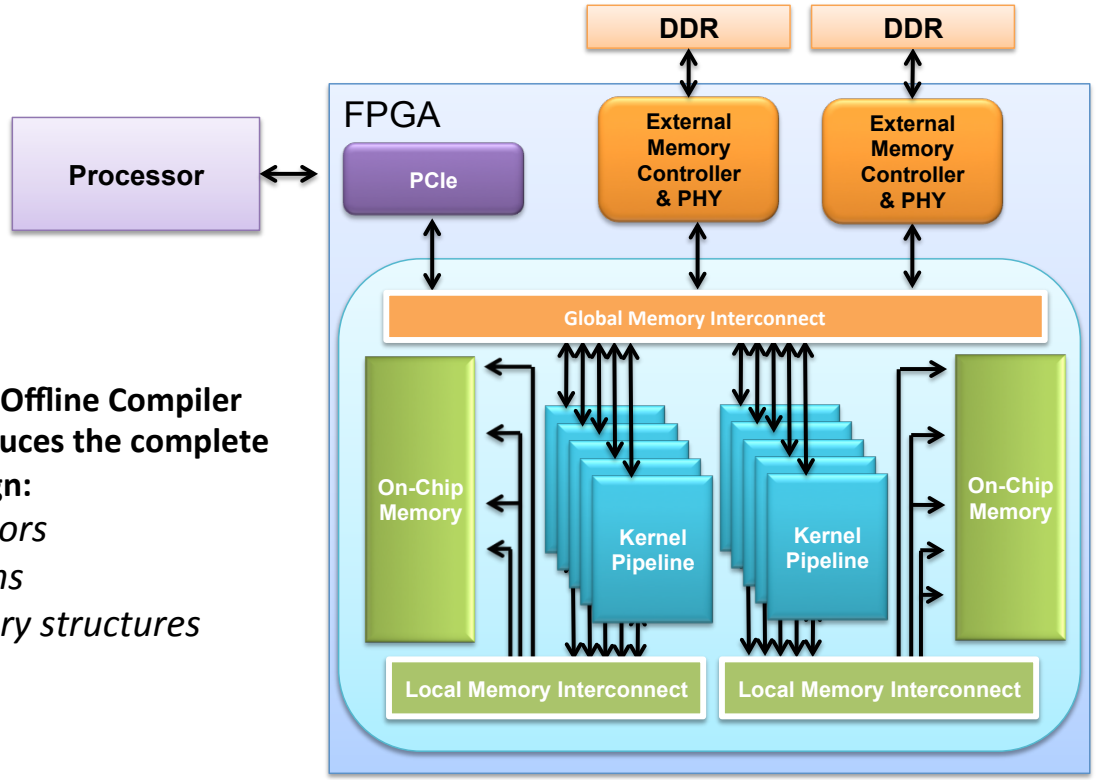


Compiling OpenCL to Altera FPGA

- Custom hardware generated automatically for each kernel
  - Get the advantages of the FPGA without the lengthy design process
- Organized into functional units based on operation
- Able to execute work-items and workgroups concurrently



OpenCL Compiler Builds Complete HDL



The Altera Offline Compiler (aoc) produces the complete FPGA design:

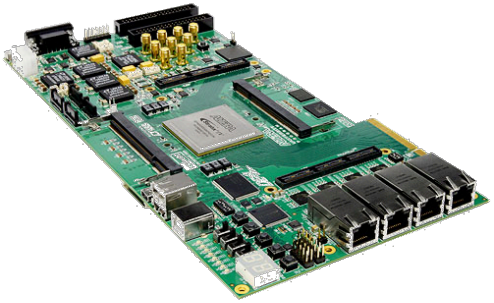
-accelerators

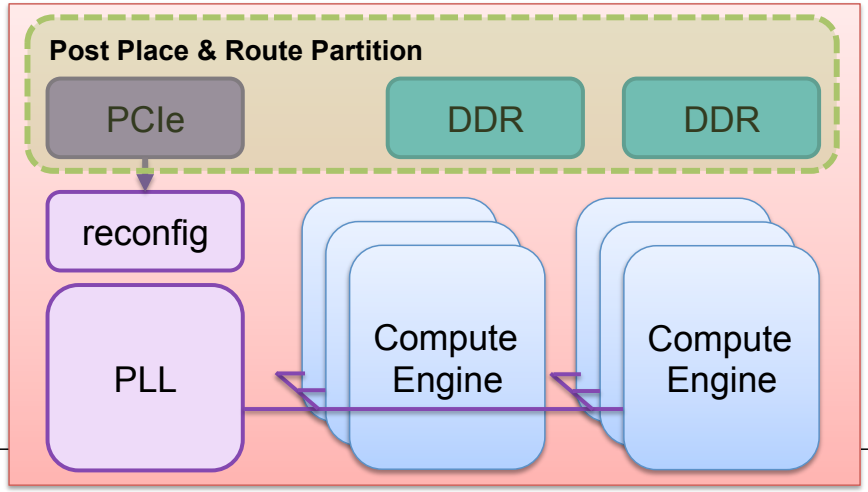
-data paths

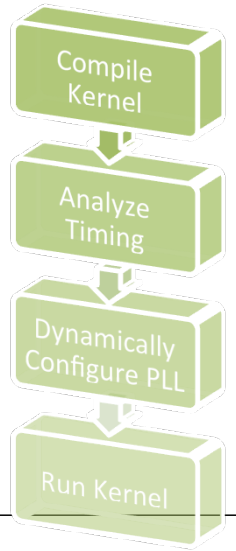
-all memory structures

Guaranteed Timing Closure

- Some interfaces have required clock frequencies
  - **PCIe** 125 MHz / 250 MHz
  - **DDR3-1600** 800 MHz
  - **Kernel** ??







OpenCL vs VHDL?

CLEMSON UNIVERSITY

- Which is better?
- Which is faster?

OpenCL Libraries with Avalon Streaming Interface

CLEMSON UNIVERSITY

- Allows you to write HDL code and interface cleanly with OpenCL
- Gives advantages of both HDL and OpenCL code
- Requires signals in HDL to communicate with OpenCL kernel
- Compiler builds a library from HDL and references this library when the function is called in OpenCL kernel

Avalon Streaming Interface

- Uses many signals to explicitly define communication to OpenCL kernels
  - datain, dataout signals defined as std_logic_vector
    - Must be as wide as a valid OpenCL datatype (8, 16, 32, etc. bits)
  - clock is used to synchronize circuits and is defined as std_logic
  - xvalid states data is valid on the source channel
    - ivalid is for the data input to the library, o is for data output
  - xready states the system is ready to receive data
    - This is called backpressure, necessary to stall applications that can only operate on one datum at a time

Libraries Requirements

- HDL description
  - Must include Avalon Interface signals
- XML configuration file
  - Describes parameters for the HDL file and is used when building the library
- Need to build library and then link it during compilation of OpenCL kernel

XML Configuration File

```
<EFI_SPEC>
    <FUNCTION name="parity"
        module="parity">
        <ATTRIBUTES>
            <IS_STALL_FREE value="yes"/>
            <IS_FIXED_LATENCY value="yes"/>
            <EXPECTED_LATENCY value="1"/>
            <CAPACITY value="1"/>
            <HAS_SIDE_EFFECTS value="yes"/>
            <ALLOW_MERGING value="no"/>
        </ATTRIBUTES>
```

XML Configuration File continued

```
<INTERFACE>
        <AVALON port="clock" type="clock"/>
        <AVALON port="resetn" type="resetn"/>
        <AVALON port="ivalid" type="ivalid"/>
        <AVALON port="iready" type="iready"/>
        <AVALON port="ovalid" type="ovalid"/>
        <AVALON port="oready" type="oready"/>
        <INPUT port="datain" width="8"/>
        <OUTPUT port="dataout" width="8"/>
 </INTERFACE>
```

XML Configuration File continued

```
        <C_MODEL>
            <FILE name="parity_c.cl" />
        </C_MODEL>
        <REQUIREMENTS>
            <FILE name="parity.vhd" />
        </REQUIREMENTS>
    </FUNCTION>
</EFI_SPEC>
```