Project XML Document

Specifications

PSU Engineering (Cerebrum) Team

Created: 17 May 2010

Updated: 11 February 2011

# CHANGE LOG

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.0 | 17 May 2010 | M. Cotter | Created the document |
| 3.0 | 11 Feb 2011 | M. Cotter | Added the “CHANGE LOG” section to document  Updated documentation for Feb 2011 release. |

Table of Contents

[1 Cerebrum Project Files 3](#_Toc270585778)

[1.1 Project Paths File 3](#_Toc270585779)

[1.1.1 Defined Path Descriptions 3](#_Toc270585780)

[1.2 Platform Specification 6](#_Toc270585781)

[1.3 Design Specification File 6](#_Toc270585782)

[1.3.1 Design-Logic Specification 6](#_Toc270585783)

[1.3.2 Design-Connections Specification 7](#_Toc270585784)

[1.3.3 Design-Groups Specification 9](#_Toc270585785)

[1.3.4 Design-Processors Specification 9](#_Toc270585786)

[1.3.5 Design-Programming Specification 10](#_Toc270585787)

[1.4 Communications Specification File 11](#_Toc270585788)

[1.4.1 Ethernet Interfaces 11](#_Toc270585789)

[1.4.2 Inter-FPGA Communications Links 12](#_Toc270585790)

[1.5 Servers (Synthesis and Programming) Specification File 12](#_Toc270585791)

[2 Component Mapping 13](#_Toc270585792)

[2.1 System Specification File (Input and Output) 13](#_Toc270585793)

[2.1.1 System Specification File <Objects> 13](#_Toc270585794)

[2.1.2 System Specification File <Components> 13](#_Toc270585795)

[2.1.3 System Specification File <FPGAs> 14](#_Toc270585796)

[2.1.4 System Specification File <Groups> 14](#_Toc270585797)

[2.1.5 System Specification File <Clusters> 15](#_Toc270585798)

[2.1.6 System Specification File <Connectivity> 16](#_Toc270585799)

[2.1.7 System Specification File <Connections> 16](#_Toc270585800)

[2.1.8 System Specification File <Links> 16](#_Toc270585801)

[2.2 Mapping Specification File (Input and Output) 17](#_Toc270585802)

[2.3 Routing Table Generation File (Output) 17](#_Toc270585803)

[2.3.1 Routing Table Generation File <ComponentConnections> 17](#_Toc270585804)

[2.3.2 Routing Table Generation File <Nodes> 18](#_Toc270585805)

[2.3.3 Routing Table Generation File <FPGALinks> 18](#_Toc270585806)

[2.4 Individual Component Map File (Output for XPS Project Generation) 18](#_Toc270585807)

[2.4.1 Individual Component Map File <ComponentMap> 19](#_Toc270585808)

[2.4.2 Individual Component Map File <FPGAs> 19](#_Toc270585809)

[2.5 Individual FPGA Map File (Output for XPS Project Generation) 19](#_Toc270585810)

[2.6 Address Map File (Input for Core Server Compiler) 20](#_Toc270585811)

[3 References 21](#_Toc270585812)

# Cerebrum Project Files

## Project Paths File

The project paths file enumerates all important project paths and files that are used for traversing the design flow. The format of the file is simple and detailed below. Each path is enumerated on its own line within a <Path> element consisting of two attributes: Name and Value. The project path manager interprets the path names as all lower case, and so is case-insensitive. It also supports indirect paths, allowing one path to reference another by using ${Name} syntax.

The example file shown here defines all paths that MUST be defined in order for the Cerebrum tools to function correctly. See the samples/paths.xml file for more detailed information about each path and what it represents in the tool.

<Paths>

<Path Name="CerebrumRoot" Value="C:\Cerebrum\v1\_00\_a" />

<Path Name="RemoteCerebrumRoot" Value="/export/home/Cerebrum/v1\_00\_a" />

<Path Name="BinDirectory" Value="${CerebrumRoot}\Tool\bin" />

<Path Name="Platforms" Value="${CerebrumRoot}\Platforms" />

<Path Name="ProjectPlatform" Value="BEE3\_Platform" />

<Path Name="LocalProjectRoot" Value="${CerebrumRoot}\Projects\DemoProject" />

<Path Name="LocalProject" Value="${LocalProjectRoot}\Project" />

<Path Name="LocalOutput" Value="${LocalProjectRoot}\output" />

<Path Name="ProjectTemp" Value="${LocalProjectRoot}\temp" />

<Path Name="RemoteProject" Value="${RemoteCerebrumRoot}/Synthesis" />

<Path Name="ProgrammingPath" Value="${RemoteCerebrumRoot}/Programming" />

<Path Name=“CoreSearchPaths” Value="/path1/cores;/path2;/path3/folder/etc" />

<Path Name="CerebrumCores" Value="${CerebrumRoot}\Cores" />

<Path Name="LocalXilinxEDKDirectory" Value="C:\Xilinx\11.1\EDK" />

<Path Name="XilinxEDKDirectory" Value="/home/mdl/Xilinx64/11.1/EDK" />

<Path Name="GlobalSynthPCores" Value="${XilinxEDKDirectory}/hw/XilinxProcessorIPLib/pcores" />

<Path Name="LinuxKernelLocation" Value="${RemoteCerebrumRoot}/linux-2.6-xlnx" />

<Path Name="DeviceTreeLocation" Value="${RemoteCerebrumRoot}/device-tree\_v0\_00\_x" />

<Path Name="ELDKLocation" Value="/export/home/opt/eldk" />

<Path Name="MicroblazeGNUTools" Value="/export/home/opt/eldk" />

<Path Name="CoreServerSource" Value="${RemoteCerebrumRoot}/Global\_Embedded\_Linux/server" />

<Path Name="LinuxBootMount" Value="/export/home/linux\_mnt" />

<Path Name="OnBoardMount" Value="/mnt/develop" />

</Paths>

### Defined Path Descriptions

#### CerebrumRoot

CerebrumRoot defines the location in which the Cerebrum Design Tools have been installed. While not strictly required, it is useful in defining other paths in the file, relative to this path.

#### RemoteCerebrumRoot

RemoteCerebrumRoot defines the location on the remote synthesis/programming servers under which remote synthesis and programming can be done. While not strictly required, it is useful in defining other paths in the file, relative to this path.

#### BinDirectory

This path defines the location in which all of the Cerebrum tool executable files have been installed.

#### Platforms

This path defines the location in which the Cerebrum design platform specifications and corresponding board specifications can be found.

#### ProjectPlatform

This entry is a full path, but corresponds to the name of a Platform folder (and matching definition) that is defined under the path specified in the [*Platforms*](#_Platforms) entry.

#### LocalProjectRoot

This path defines the location in which the active project will be located. All project files, cores, etc should be located within this folder, and any such paths should be specified relative to this folder.

#### LocalProject

This path defines the location in which each FPGA’s XPS Project Folder will be built and stored.

#### LocalOutput

This path defines the location where the BIT and ELF files produced by synthesis and compilation are stored together locally.

#### ProjectTemp

This path defines a temporary working location in which all Cerebrum tools may store temporary files.

#### RemoteProject

This path defines the location on the synthesis server where the XPS Projects are to be located and synthesized.

#### ProgrammingPath

This path defines the location on the programming server where the BIT and ELF files are to be transferred for programming.

#### CoreSearchPaths

This path defines a semicolon-delimited list of paths that should be searched for cores and their respective definition files during XPS Project Building. The paths are searched in-order, with the first match being chosen.

#### CerebrumCores

This path defines the location, defined by Cerebrum, which specifies where standard, Cerebrum-provided cores are located for use in **any** Cerebrum project. This path is searched for core files during XPS Project build only after no matches are found in any of the paths listed in [*CoreSearchPaths*](#_CoreSearchPaths).

#### LocalXilinxEDKDirectory

This path defines a location of the Xilinx EDK Tools Directory, if it is installed locally. When searching for cores, if this path is defined, the Xilinx IP Core Lib in this location will be searched for pcores, if a match is not found in [*CoreSearchPaths*](#_CoreSearchPaths) or [*CerebrumCores*](#_CerebrumCores). If this path is not defined, it is ignored.

#### XilinxEDKDirectory

This path defines a location of the Xilinx EDK Tools Directory that must be installed on the synthesis/programming servers. When searching for cores, the Xilinx IP Core Lib in this location will be searched for pcores, if a match is not found in any previous core repositories. This path must be defined.

#### GlobalSynthPCores

This path defines a semicolon-delimited list of paths that should be included as module search paths during synthesis. These paths are expected to exist on the synthesis server and must terminate 2 directory levels above the location of the pcores. For example, if these cores are located in “/home/cerebrum/hw/ipcorelib/pcores”, the path in this entry should include “/home/cerebrum/hw”.

#### LinuxKernelLocation

This path defines the location to the root of the Linux source tree that will be used for compiling the Embedded Linux kernel (currently supported ver. 2.6.31) for operation on the FPGA(s). This value may be overridden on a per-processor basis by specifying the [*LinuxKernelSource*](#_LinuxKernelSource) parameter in the [*Processor*](#_Design-Processors_Specification) block of the [*Design*](#_Design_Specification_File) file.

#### DeviceTreeLocation

This path defines location of the device-tree drivers that are to be used in compiling the Embedded Linux kernel.

#### ELDKLocation

This path defines the full path to the location of the ELDK cross compilation tool, used in compiling the Embedded Linux kernel.

#### MicroblazeGNUTools

This path defines the full path to the location of the Microblaze GNU Tools, used in compiling the Embedded Linux kernel for Microblaze processors.

#### CoreServerSource

This path defines the full path to the location of the source root for the Cerebrum-standard core server applications and drivers.

#### LinuxBootMount

This path defines the location of the path that is boot-mounted by the Linux RAMDisk when started on the FPGAs. Once compiled, Core server applications and drivers will be copied to this location to be accessed by the FPGA when it starts.

#### OnBoardMount

This path defines the path to [*LinuxBootMount*](#_LinuxBootMount), as seen by the Linux running on the FPGA. This is the mount point on the FPGA, to which the location specified by [*LinuxBootMount*](#_LinuxBootMount) is mounted at startup.

## Platform Specification

The hardware platform that will be used in the design is defined by the [*ProjectPlatform*](#_ProjectPlatform) entry in the Project Paths file. The exact specification of the Platform can be found in the Platform XML Specifications File [[1]](#_References).

## Design Specification File

The design specification file currently consists of sections: Logic, Connections, Groups, Processors, and Programming.

### Design-Logic Specification

The logic section details the attributes of design cores that have been added to the project. These attributes include a unique ID, a generic display name, the location of the core’s source files, the owner of the core, the version, whether it requires server application support (for the core server compiler) and a list of FPGA architectures (virtex4, virtex5, etc) supported by the core. Each core also has a set of sub-elements detailing the resources required for the core.

<Design>

<Logic>

<Core

ID="component0"

Name="cerebrum\_component"

Location=""

OwnerName="Penn State"

Version="1.00.a"

Server="false"

SupportedArch="">

<Resource Name="slices" Value="1000" />

</Core>

…

</Logic>

…

</Design>

#### ID

This is the instance name of the design core in the project. This ID must be unique among all cores in the design.

#### Name

This is the core’s type name. This value need not be unique among cores in the design. This value, combined with the [Version](#_Version), is used to locate the core during XPS Project Generation and Synthesis.

#### Location

*This value is currently unused.* This value indicates the location of the core definition files on the disk.

#### OwnerName

*This value is currently unused.* This value indicates the owner of the IP defined by the core.

#### Version

This value indicates the version number of the core. This value, combined with the [Name](#_Name), is used to locate the core during XPS Project Generation and Synthesis.

#### Server

This value is a Boolean, “true” or “false” indicating whether the Core Server Applications/Drivers should be compiled for this core.

#### SupportedArch

This is a comma-delimited list of FPGA architectures that are supported by this core. If none are specified, then it is assumed that all architectures are supported.

#### Resource

A core element may contain multiple Resource elements. Each element defines the name and value/amount of a resource that is required by the design core.

### Design-Connections Specification

The connections section describes how the logic cores are interconnected within the design. Each connection is assigned a unique identifier and indicates both the source and sink components for the connection

<Design>

…

<Connections>

<Connection

ID="conn\_0"

Source="component0"

SourceInstance=" core0"

Sink="component1"

SinkInstance="core0" />

…

</Connections>

…

</Design>

#### ID

This is the ID of the connection between cores in the project. This ID must be unique among all connections in the design.

#### Source

This is the ID of the component that provides the input data to this connection—the source of the data flow.

#### SourceInstance

This is the internal instance name of the core inside the component that provides the input data to this connection—the source of the data flow.

#### Sink

This is the ID of the component that receives the output data from this connection—the sink of the data flow.

#### SinkInstance

This is the internal instance name of the core inside the component that receives the output data to this connection—the sink of the data flow.

### Design-Groups Specification

The groups section describes how the logic cores are to be physically grouped together within the design. This allows the user to specify a minimum connection cost between components as grouped components will be collocated on the same FPGA.

<Design>

…

<Groups>

<Group Name="autogroup\_0" ID=" autogroup\_0">

<ComponentMember ID="Hard\_Ethernet\_MAC" />

<ComponentMember ID="gabor\_filter\_11x11\_0" />

</Group>

…

</Groups>

…

</Design>

#### Name

This is the display-name of the group of cores in the project.

#### ID

This is the ID of the group of cores in the project. This ID must be unique among all groups in the design.

#### ComponentMember

A core element may contain multiple ComponentMember elements. Each element defines the ID of a core that is to be considered a member of the group.

### Design-Processors Specification

The processors section describes all of the processors that are to be synthesized on the FPGAs within the system. Each processor specification indicates which FPGA it will be located on, its instance name on the FPGA, the software type that it will run (Linux OS is the only type supported so far), the type (PowerPC or Microblaze), and the console device that it will use for I/O. Additionally, each processor requires a CPU number to be specified for programming.

<Design>

…

<Processors>

<Processor OnDevice="510\_R.fpga\_0">

<Instance>ppc440\_0</Instance>

<OS>Linux</OS>

<Type>PowerPC</Type>

<ConsoleDevice>RS232\_Uart\_1</ConsoleDevice>

<DTS>virtex440-ml510</DTS>

<MakeConfig>virtex44x/defconfig</MakeConfig>

<CompilerArgs>ppc\_4xx</CompilerArgs>

<CPUNumber>1</CPUNumber>

<LinuxKernelSource>1</LinuxKernelSource>

<CerebrumID>B93CE865-5C83-4e49-AD52-E224FD3E5D14</CerebrumID>

</Processor>

…

</Processors>

…

</Design>

#### Processor.OnDevice

The Device attribute of each Processor element indicates the Cerebrum ID of the FPGA on which the processor will be instantiated.

#### Instance

This is the instance name of the processor.

#### OS

Currently, this value must be set to Linux.

#### Type

This value indicates the type of processor: PowerPC or Microblaze.

#### DTS

This is the name of the DTS file that is used for compiling the Linux kernel for the processor.

#### MakeConfig

This is the argument passed to make, in order to copy the appropriate configuration file just prior to compiling the Linux kernel.

#### CompilerArgs (PowerPC Only)

This specifies the argument(s) that are passed to eldk\_init in order to configure the cross compiler for compiling the Linux kernel for PowerPC.

#### CPUNumber

This is the CPU Number used for programming. This value is calculated by the XPS Builder.

#### LinuxKernelSource

This value specifies the path to the Linux Kernel Source tree that should be used for this processor. If this value is not specified, the default value is the [*LinuxKernelLocation*](#_LinuxKernelLocation) path from the Project Paths file.

#### CerebrumID

This value specifies a globally-unique identifier (GUID) that is to be assigned to this processor. This GUID will be inserted into the Linux Kernel compiled for the processor and allow it to identify itself on bootup. If this value is not specified, a GUID will be generated by the Synthesis tool.

### Design-Programming Specification

Finally, the programming section describes how each device will be programmed on the target FPGA. Each program specification indicates the device it applies to, as well as the cable type and cable port that are to be used for programming.

<Design>

…

<Programming>

<Program Device="510\_R.fpga\_0">

<CableType>xilinx\_platformusb</CableType>

<CablePort>usb21</CablePort>

</Program>

…

</Programming>

</Design>

#### Program.Device

The Device attribute of each Program element indicates the Cerebrum ID of the FPGA to which the programming information applies.

#### CableType

This specifies the type of cable to use for JTAG programming. Legal values here are “xilinx\_parallel”, “xilinx\_parallel3”, and “xilinx\_platformusb”.

#### CablePort

This value specifies the cable port ID to be used for connecting to the programming cable. Examples of this value may be “usb21”, “paport11”, etc. The exact value of the port ID depends on the connection order and cable type.

## Communications Specification File

The communications file specifies the communication interfaces for each FPGA in the platform. Currently, the focus is on Ethernet-style communication, but can be expanded to include other types. The specification defines an interface for each device, and within it, the parameters required to enforce that interface on the FPGA device.

### Ethernet Interfaces

<Interfaces>

<Interface Device="board\_0.fpga\_0">

<DHCP>true</DHCP>

<EthernetDevice>Hard\_Ethernet\_MAC</EthernetDevice>

<EthernetMAC>00-00-FF-00-00-FF</EthernetMAC>

<IPAddress>192.168.1.4</IPAddress>

</Interface>

…

</Interfaces>

#### Interface.Device

The Device attribute of each Interface element indicates the Cerebrum ID of the FPGA to which the interface information applies.

#### DHCP

This value is a Boolean, “true” or “false”, indicating whether the interface should be configured to use DHCP to obtain an IP address.

#### EthernetDevice

This value specifies the instance name of the core (on the FPGA) to which this interface applies.

#### EthernetMAC

This value specifies the Ethernet MAC address to be coded into the Ethernet interface.

#### EthernetDevice

This value specifies the IP Address that will be assigned to the device. If DHCP is false, this value will be configured as a static IP in the Linux kernel. If DHCP is true, this value indicates the IP address that is expected to be assigned to the Ethernet interface (likely based on the MAC address). In either case, this value also is used to configure any Core Server Applications/Drivers for cores assigned to the FPGA.

## Servers (Synthesis and Programming) Specification File

Both the synthesis server and programming server specification files use the same format. The files are distinct however, to easily support both using the same servers, or different servers for the two procedures. The file allows for a list of servers so that synthesis and/or programming may be done in parallel across multiple servers.

<Servers>

<Server>

<ID>0</ID>

<UserName>username</UserName>

<Address>server\_address</Address>

<LinuxHost>true</LinuxHost>

<Port>22</Port>

</Server>

</Servers>

# Component Mapping

The component mapping system accepts 2 input specifications that are provided in XML format—the system specification and the mapping specification. The system specification is required, and must be contained entirely within a single file. The mapping specification is optional and must be contained entirely within a single file. If desired, both of these specifications may be contained together in the same file. The IDs used to uniquely identify objects and connections in the mapping system are case-insensitive, as all IDs are converted to and treated in lower case, so uniqueness of IDs must be maintained without regard to case.

In either case, each specification file read in, system or mapping, must define <MappingSystem> as the top-level tag.

## System Specification File (Input and Output)

The system specification file describes all of the objects within the system under two classification tags: <Objects> and <Connectivity>. <Objects> defines the key system objects, and their groupings within the logical and physical design. <Connectivity> defines any interconnections, both logical and physical, between objects in the design.

It is possible during use of the mapping system to change values, names, IDs or any other property of any object within the system. The resulting system, reflecting these changes, may also be created upon completion. The most notable change between the input and output system specifications will be the generation and/or removal of component groups. If the resulting system mapping specification is to be used as a future input, this resulting system specification must also be used, or any changes made for the mapping may not be accurately reflected.

### System Specification File <Objects>

<Objects> describes the logical <Components>, or processing elements, of the design; the physical hardware <FPGAs> on which those Components are to be implemented; the logical <Groups> of Components that are defined for mapping; and logical (or physical) <Clusters> of FPGAs in the hardware.

### System Specification File <Components>

This subsection describes the set of logical components in the design. Each component is described using the following format:

<Component ID=”component\_id” Name=”component\_name”>

<Resource Name=”resource\_name0” Value=”resource\_amount0” />

<Resource Name=”resource\_name1” Value=”resource\_amount1” />

…

</Component>

The value of Component.ID is only used to uniquely identify that component within the mapping system, and therefore must not be shared with ANY other component defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation. The value of Component.Name is to be used for descriptive purposes, and is not required to be unique.

The hardware resources required by each component are also defined as sub-elements using the <Resource> tag. The value of Resource.Name is the string matching the name of the required resource on the FPGA. As of now (5/17/2010) this value is not enumerated by any specific set of possible values, and so care must be taken to correctly populate and modify this value. The strings used for resource names are also case-insensitive. (i.e. “BRAMs” is treated the same as “brams”) Resource.Value indicates the integer-valued amount of the resource that is required by the component. If the same resource is defined multiple times within a single component definition, the last value is used—the values are added NOT cumulatively.

### System Specification File <FPGAs>

This subsection describes the set of logical components in the design. Each component is described using the following format:

<FPGA ID=”fpga\_id” Name=”fpga\_name”>

<Resource Name=”resource\_name0” Value=”resource\_amount0” />

<Resource Name=”resource\_name1” Value=”resource\_amount1” />

…

</FPGA>

The value of FPGA.ID is only used to uniquely identify that FPGA within the mapping system, and therefore must not be shared with ANY other FPGA defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation. The value of FPGA.Name is to be used for descriptive purposes, and is not required to be unique.

The total hardware resources available on each FPGA are also defined as sub-elements using the <Resource> tag. The value of Resource.Name is the string matching the name of the resource available on the FPGA. As of now (5/17/2010) this value is not enumerated by any specific set of possible values, and so care must be taken to correctly populate and modify this value. The strings used for resource names are also case-insensitive. (i.e. “BRAMs” is treated the same as “brams”) Resource.Value indicates the integer-valued amount of the resource that is available for components on that FPGA. If the same resource is defined multiple times within a single FPGA definition, the last value is used—the values are NOT added cumulatively.

### System Specification File <Groups>

This subsection describes any groups defined for mapping and if no such groups are defined, is entirely optional. If defined, each group is described using the following format:

<Group ID=”group\_id” Name=”group\_name”>

<ComponentMember ID = “component\_id0” />

<ComponentMember ID = “component\_id1” />

…

</Group>

The value of Group.ID is only used to uniquely identify that Group within the mapping system, and therefore must not be shared with ANY other Group defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation. The value of Group.Name is to be used for descriptive purposes, and is not required to be unique.

The definition of groups in the input file is optional. Upon execution of the algorithm, each group mapped to an FPGA, rather than each component. If any component is found to not be contained with a group, a new group is automatically generated, and that component becomes the sole member. If there are no groups defined, each component will have a group generated automatically during mapping.

Each defined group may have multiple component members, each specified using the <ComponentMember/> tag within the <Group> definition. The value of ComponentMember.ID must match an ID assigned to a component defined in <Components>. If no matching value is found, an error will occur. If a single component ID appears in a ComponentMember.ID field of multiple groups, an error will occur.

If any groups are defined that have no members, they will be ignored and purged by the algorithm.

### System Specification File <Clusters>

This subsection defines any clusters defined for logical display by the mapping system. Clusters are defined purely for visual (GUI) purposes and have no direct impact on the mapping algorithm. If no such clusters are defined, this section is entirely optional. If defined, each cluster is described using the following format:

<Cluster ID=”cluster\_id” Name=” cluster \_name”>

<FPGAMember ID = “fpga\_id0” />

<FPGAMember ID = “fpga\_id1” />

…

</Cluster>

The value of Cluster.ID is only used to uniquely identify that Cluster within the mapping system, and therefore must not be shared with ANY other Cluster defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation. The value of Cluster.Name is to be used for descriptive purposes, and is not required to be unique.

The definition of clusters in the input file is optional. Their presence or lack thereof, has no direct impact on the mapping algorithm. They are implemented purely to provide a visual perspective of complex hardware platforms in which grouping FPGAs together may allow for a clearer picture of the design and/or data flow.

Each defined cluster may have multiple FPGA members, each specified using the <FPGAMember/> tag within the <Cluster> definition. The value of FPGA.ID must match an ID assigned to a component defined in <FPGAs>. If no matching value is found, an error will occur. If a single FPGA ID appears in an FPGAMember.ID field of multiple groups, an error will occur.

If any clusters are defined that have no members, they will be ignored and purged by the algorithm.

### System Specification File <Connectivity>

<Connectivity> describes the logical <Connections> between Components in the design, and the physical <Links> between FPGAs in the hardware platform.

### System Specification File <Connections>

This subsection defines the flow of data between logical components in the design by detailing the connections between them. Each connection is described using the following format:

<Connection ID=”connection\_id” Input=”source\_component\_id”

Output=”sink\_component\_id” DataWeight=”data\_weight\_value” />

The value of Connection.ID is only used to uniquely identify that Connection within the mapping system, and therefore must not be shared with ANY other Connection defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation.

The value of Connection.Input is the ID of the (source) component that generates the data transmitted across this connection—the input to the connection. The value of Connection.Output is the ID of the (sink) component that receives the data transmitted across this connection—the output of the connection. Finally, Connection.DataWeight is supported to provide a way of describing how much data is sent by the source component (Connection.Input). If IDs specified by Connection.Input or Connection.Output do not correspond to IDs of components specified in <Components>, an error will occur.

### System Specification File <Links>

This subsection defines the flow of data between physical FPGAs in the hardware platform by detailing the links between them. Each link is described using the following format:

<Link ID=”link\_id” Input=”source\_fpga\_id”

Output=”sink\_fpga\_id” LinkSpeed=”link\_speed\_value”

Bidirectional=”true/false” />

The value of Link.ID is only used to uniquely identify that Link within the mapping system, and therefore must not be shared with ANY other Link defined in the file. Outside of this purpose, the ID has no significance to the design or the implementation.

The value of Link.Input is the ID of the (source) component that generates the data transmitted across this connection—the input to the connection. The value of Link.Output is the ID of the (sink) component that receives the data transmitted across this connection—the output of the connection. Link.LinkSpeed is supported to provide a way of describing how fast data can be sent by the source FPGA (Link.Input) to the sink FPGA (Link.Output). Finally, Link.Bidirectional is a Boolean whose value indicates whether the link is unidirectional (“false”) or bidirectional (“true”). If IDs specified by Link.Input or Link.Output do not correspond to IDs of components specified in <FPGAs>, an error will occur.

## Mapping Specification File (Input and Output)

The mapping specification file defines the mapping of component-groups to target FPGAs. When provided as input, the system assumes that any mappings read correspond to user-defined pre-mappings of the system. If these mappings would cause any invalid assignments to occur, due to over-allocation of resources, an error will occur prior to mapping proceeding. In order to support a before-and-after style of mapping, the specification for mapping is broken into 2 sections: <PreMapping> and <PostMapping>. The format of each section is the same and is described using the following format:

<PreMapping> <!-- or <PostMapping> -->

<MappedGroup ID=”group\_id0” TargetFPGA=”fpga\_id0” />

<MappedGroup ID=”group\_id1” TargetFPGA=”fpga\_id1” />

…

</PreMapping> <!-- or </PostMapping> -->

Each MappedGroup specifies two IDs: the ID of the group it represents, and the ID of the FPGA to which that group is mapped. If a group with an ID matching MappedGroup.ID does not exist in the set of groups defined in the system specification <Groups> section, or if no FPGA exists in <FPGAs> with an ID matching MappedGroup.TargetFPGA, the mapping is ignored without error as invalid.

The <PreMapping> and <PostMapping> tags generated by the mapping system will also have a Date=”MM/dd/YYYY hh:mm AMPM” attribute indicating the time it was generated. This attribute is not required, but is generated for accounting purposes.

## Routing Table Generation File (Output)

Once the component mapping has been completed, the routing tables for the hardware on-FPGA routers must be generated so that the flow of data can be routed from one component (on one FPGA) to the next component in sequence (on possibly another FPGA). This file is currently generated by the mapping system is it has access to all of the required information.

The format of the XML file is different from the system mapping, however, as it is more focused on the connections rather than the objects. The top-level tag for this file is defined as <SystemRouting>, and is divided into 3 types of information required to calculate the routing tables: <ComponentConnections>, <FPGAs>, and <FPGALinks>.

### Routing Table Generation File <ComponentConnections>

This subsection defines each connection in the system, post-mapping, detailing the source/sink component IDs (used by the mapping system) as well as the corresponding mapped source/sink FPGA IDs (used by the mapping system). The normalized data weight is also provided as generated by the mapping system. Each connection is described using the following format:

<ConnectionInfo SourceComponent=”source\_component\_id”

SinkComponent=”sink\_component\_id” SourceFPGA=”source\_fpga\_id”

SinkFPGA=”sink\_fpga\_id” Weight=”weight\_value” />

…

The values of ConnectionInfo.SourceComponent and ConnectionInfo.SinkComponent represent the IDs of the components (from the mapping system) connected by the connection. The values of ConnectionInfo.SourceFPGA and ConnectionInfo.SinkFPGA then correspond to the IDs of the FPGAs to which SourceComponent and SinkComponent are mapped, respectively. For routing purposes, this indicates that the data output by SourceComponent on SourceFPGA must be routed to SinkComponent on SinkFPGA through the routing tables. The IDs specified in SourceFPGA and SinkFPGA correspond to FPGA IDs defined in the <FPGAs> section of this file.

The value of ConnectionInfo.Weight is the normalized weight of the data, based on the metric provided by the user to the mapping system. The meaningfulness of this value depends on how it was obtained for the system specification of the mapping system input.

### Routing Table Generation File <Nodes>

This section is simply a list of the Nodes (FPGAs) enumerated in the mapping system and their corresponding IDs. This section is provided simply to ensure that all FPGAs are accounted for when creating routing tables. This is required for the case in which an FPGA has no components mapped to it (and therefore no connections to components on it), but still may be used for routing of data between other components. Each FPGA is enumerated using the following format:

<Node ID="fpga\_id" />

…

The value of Node.ID is the ID of the FPGA from the mapping system, and relates the IDs specified in ConnectionInfo for SourceNode and SinkNode to physical FPGAs for routing.

### Routing Table Generation File <FPGALinks>

This section enumerates the set of physical connections or links between FPGAs on the platform. Each link is described using the following format:

<LinkInfo SourceFPGA="source\_fpga\_id" SinkFPGA="sink\_fpga\_id"

LinkSpeed="link\_speed" Bidirectional="True/False" />

The values of LinkInfo.SourceFPGA and LinkInfo.SinkFPGA represent the IDs of the FPGAs (from the mapping system) connected by the link. These IDs correspond to the FPGAs defined in <FPGAs> as well. The LinkInfo.LinkSpeed value is an indicator of the relative speed of the link. This value is based on the link speed specified as input to the mapping system. The meaningfulness of this value depends on how it was obtained for the system specification of the mapping system input. Finally, LinkInfo.Bidirectional is a Boolean whose value indicates whether the link is unidirectional (“false”) or bidirectional (“true”).

## Individual Component Map File (Output for XPS Project Generation)

This XML file, also generated by the mapping system, enumerates all components in the system, indicating the FPGA to which each component is mapped to. The top-level tag of this file is <SystemMap> and it contains two child tags: <ComponentMap> and <FPGAs>.

### Individual Component Map File <ComponentMap>

This section lists each component and the FPGA it’s mapped to using the following format:

<Component ID="11" FPGA="2" />

…

The value of Component.ID indicates the ID of the component (from the mapping system). The value of Component.FPGA indicates the ID of the FPGA the component is mapped to. The value of Component.FPGA also corresponds to an ID in the enumeration of IDs in <FPGAs>

### Individual Component Map File <FPGAs>

This section is simply a list of the FPGAs enumerated in the mapping system and their corresponding IDs. This section is provided simply to ensure that all FPGAs are accounted for when creating routing tables. This is required for the case in which an FPGA has no components mapped to it (and therefore no connections to components on it), but still may be used for routing of data between other components. Each FPGA is enumerated using the following format:

<FPGA ID="fpga\_id" />

…

The value of FPGA.ID is the ID of the FPGA from the mapping system, and relates the IDs specified in ConnectionInfo for SourceFPGA and SinkFPGA to physical FPGAs for routing.

## Individual FPGA Map File (Output for XPS Project Generation)

This XML file, also generated by the mapping system, enumerates all FPGAs in the system, listing all components that are mapped to it. The top-level tag of this file is <SystemMap> and each FPGA in the system is defined as a direct child of this tag. Each FPGA is defined using the following format:

<FPGA ID=”fpga\_id”>

<Component ID=”component\_id0” />

<Component ID=”component\_id1” />

<Component ID=”component\_id2” />

…

</FPGA>

The value of FPGA.ID indicates the ID of the FPGA (from the mapping system) being described. The value of each Component.ID within the <FPGA> definition indicates the ID of a component (also from the mapping system) that is mapped to the FPGA.

## Address Map File (Input for Core Server Compiler)

This XML file is generated by the Mapping Algorithm, and subsequently updated by the XPS Project Builder after address validation. It contains information used by the Core Server Compiler to compile the On-FPGA core application and drivers, specific to each core and includes an IP address, Port number, along with the address space and size assigned to each core.

This information is also used in generating scripts that allow the FPGA to attempt to identify itself upon boot up and execute the scripts to load the drivers and servers required for the cores assigned to it.

<AddressMap>

<Component

ID="component\_0"

Name="demo\_component"

FPGA="board\_0.fpga\_0"

IP="192.168.1.4"

MAC="00-FF-FF-00-FF-FF"

Port="1234"

BASEADDR="0xC0A00000"

HIGHADDR="0xC0A0FFFF"

ADDRSIZE="0x0000FFFF"

TARGETS="mb;ppc;" />

…

</AddressMap>

#### ID

This is the ID / Instance Name of the component in the design.

#### Name

This is the display name of the component.

#### FPGA

This is the ID of the FPGA to which the component was mapped.

#### IP

This is the IP Address that is (or will be) assigned to the Ethernet interface of the FPGA.

#### MAC

This is the MAC Address that is assigned to the Ethernet interface of the FPGA.

#### Port

This is the TCP port assigned to the component server running on the FPGA.

#### BASEADDR

This the base address of the component on the processor PLB bus, used for processor-component communication.

#### HIGHADDR

This the high address of the component on the processor PLB bus, used for processor-component communication.

#### ADDRSIZE

This the size of the address space reserved for the component on the processor PLB bus, used for processor-component communication.

#### TARGETS

This is a list of processor architectures for which the core server application and drivers are to be compiled for. Legal values are “powerpc” or “ppc” for PowerPC processors and “microblaze” or “mb” for Microblaze processors.

# References

1. Platform XML Specifications:

<https://www.cse.psu.edu/svn/mdl/falcon_repository/trunk/Software/Cerebrum/Documentation/platform_xml.docx>