connectal Documentation

Release 14.11.6

Jamey Hicks, Myron King, John Ankcorn

December 11, 2014

Contents:

CHAPTER

ONE

INTRODUCTION

Introduction goes here.

CHAPTER

TWO

CONNECTAL BSV LIBRARIES

2.1 HostInterface Package

The HostInterface package provides host-specific typedefs and interfaces.

`HostInterface::`**`DataBusWidth`**
    Width in bits of the data bus connected to host shared memory.

`HostInterface::`**`PhysAddrWidth`**
    Width in bits of physical addresses on the data bus connected to host shared memory.

`HostInterface::`**`NumberOfMasters`**
    Number of memory interfaces used for connecting to host shared memory.

interface `HostInterface::`**`BsimHost`**
    Host interface for the bluesim platform

interface `HostInterface::`**`PcieHost`**
    Host interface for PCIe-attached FPGAs such as vc707 and kc705

interface `HostInterface::`**`ZynqHost`**
    Host interface for Zynq FPGAs such as zedboard, zc702, zc706, and zybo.

    The Zc706 is a ZynqHost even when it is plugged into a PCIe slot.

2.2 Leds Package

interface `Leds::`**`LEDS`**

`Leds::`**`LedsWidth`**
    Defined to be the number of default LEDs on the FPGA board.

    The Zedboard has 8, Zc706 has 4, ...

2.3 CtrlMux Package

`CtrlMux::`**`mkInterruptMux`**
    Used by BsimTop, PcieTop, and ZynqTop

`CtrlMux::`**`mkSlaveMux`** → PhysMemSlave#(addrWidth,dataWidth)
    Takes a vector of MemPortals and returns a PhysMemSlave combining them.

2.4 Portal Package

2.4.1 PipePortal Interface

interface `Portal::`**`PipePortal`**
`:parameter: numeric type numRequests, numeric type numIndications, numeric type slaveDataW`

> **`messageSize`** (Bit#(16) methodNumber) → Bit#(16)
> Returns the message size of the methodNumber method of the portal.

`requests` → Vector#(numRequests, PipeIn#(Bit#(slaveDataWidth)))

`indications` → Vector#(numIndications, PipeOut#(Bit#(slaveDataWidth)))

2.4.2 MemPortal Interface

interface `Portal::`**`MemPortal`**

`slave` → PhysMemSlave#(slaveAddrWidth,slaveDataWidth)

`interrupt` → ReadOnly#(Bool)

`top` → WriteOnly#(Bool)

`Portal::`**`getSlave`** (MemPortal#(_a,_d) p) → PhysMemSlave(_a,_d)

`Portal::`**`getInterrupt`** (MemPortal#(_a,_d) p) → ReadOnly#(Bool)

`Portal::`**`getInterruptVector`** (Vector#(numPortals, MemPortal#(_a,_d)) portals) → Vector#(16, ReadOnly#(Bool))

2.4.3 ShareMemoryPortal Interface

interface `Portal::`**`SharedMemoryPortal`**
Should be in SharedMemoryPortal.bsv

`readClient;`

`writeClient` → MemWriteClient#(dataBusWidth)

interface `cfg` → SharedMemoryPortalConfig

`Portal::`**`interrupt`** → ReadOnly#(Bool)

CHAPTER

THREE

CONNECTAL EXAMPLES

3.1 Simple Example

CHAPTER FOUR

PYTHON

AST
bsvgen
cppgen
discover_tcp
makefilegen
util

4.1 AST

Functions

classInfo(item)	
declInfo(mitem)	
dtInfo(arg)	
piInfo(pitem)	
serialize_json(interfaces, globalimports, ...)	Returns json serialized data

Classes

Enum(elements)	Represents a BSV Enum
EnumElement(name, qualifiers, value)	
Function(name, return_type, params)	
Interface(name, params, decls, ...)	Represents a BSV Interface
InterfaceMixin	
Method(name, return_type, params)	
Module(moduleContext, name, params, ...)	represents a BSV Module
Param(name, t)	
Struct(elements)	Represents a BSV Struct
StructMember(t, name)	
Type(name, params)	Represents a BSV Type
TypeDef(tdtype, name, params)	
Typeclass(name)	
TypeclassInstance(name, params, provisos, decl)	
Variable(name, t)	

class `AST.`**`Enum`**(*elements*)
Represents a BSV Enum

`instantiate`(*paramBindings*)

class `AST.`**`EnumElement`**(*name*, *qualifiers*, *value*)

class `AST.`**`Function`**(*name*, *return_type*, *params*)

class `AST.`**`Interface`**(*name*, *params*, *decls*, *subinterfacename*, *packagename*)
Bases: `AST.InterfaceMixin`

Represents a BSV Interface

Create an Interface with name, params, decls, subintfacename, packagename

`instantiate`(*paramBindings*)

`interfaceType`()

class `AST.`**`InterfaceMixin`**

`getSubinterface`(*name*)

`parentClass`(*default*)

class `AST.`**`Method`**(*name*, *return_type*, *params*)

`instantiate`(*paramBindings*)

class `AST.`**`Module`**(*moduleContext*, *name*, *params*, *interface*, *provisos*, *decls*)
represents a BSV Module

instantiates a BSV Module

class `AST.`**`Param`**(*name*, *t*)

`instantiate`(*paramBindings*)

class `AST.`**`Struct`**(*elements*)
Represents a BSV Struct

`instantiate`(*paramBindings*)

class `AST.`**`StructMember`**(*t*, *name*)

`instantiate`(*paramBindings*)

class `AST.`**`Type`**(*name*, *params*)
Represents a BSV Type

`instantiate`(*paramBindings*)

class `AST.`**`TypeDef`**(*tdtype*, *name*, *params*)

class `AST.`**`Typeclass`**(*name*)

class `AST.`**`TypeclassInstance`**(*name*, *params*, *provisos*, *decl*)

class `AST.`**`Variable`**(*name*, *t*)

`AST.`**`classInfo`**(*item*)

`AST.`**`declInfo`**(*mitem*)

`AST.`**`dtInfo`**(*arg*)

`AST.`**`piInfo`**(*pitem*)

`AST.`**`serialize_json`**(*interfaces*, *globalimports*, *dutname*, *interfaceList*)
Returns json serialized data

4.2 bsvgen

Functions

`collectElements`(mlist, workerfn, name)	
`fixupSubsts`(item, suffix)	
`generate_bsv`(project_dir, noisyFlag, jsondata)	
`toBsvType`(titem)	

`bsvgen.`**`collectElements`**(*mlist*, *workerfn*, *name*)

`bsvgen.`**`fixupSubsts`**(*item*, *suffix*)

`bsvgen.`**`generate_bsv`**(*project_dir*, *noisyFlag*, *jsondata*)

`bsvgen.`**`toBsvType`**(*titem*)

4.3 cppgen

Functions

`accumWords`(s, pro, memberList)	
`cName`(x)	
`collectMembers`(scope, pitem)	
`emitCD`(item, generated_hpp, indentation)	
`emitEnum`(item, name, f, indentation)	
`emitMethodDeclaration`(mname, params, f, ...)	
`emitStruct`(item, name, f, indentation)	
`emitStructMember`(item, f, indentation)	
`formalParameters`(params, insertPortal)	
`gatherMethodInfo`(mname, params, itemname)	
`generate_class`(className, declList, parentC, ...)	
`generate_cpp`(project_dir, noisyFlag, jsondata)	
`generate_demarshall`(fmt, w)	
`generate_marshall`(pfmt, w)	
`getNumeric`(item)	
`hasBitWidth`(item)	
`indent`(f, indentation)	
`typeBitWidth`(item)	
`typeCName`(item)	
`typeNumeric`(item)	

Classes

`paramInfo`(name, width, shifted, datatype, ...)	

cppgen.**accumWords** (*s*, *pro*, *memberList*)

cppgen.**cName** (*x*)

cppgen.**collectMembers** (*scope*, *pitem*)

cppgen.**emitCD** (*item*, *generated_hpp*, *indentation*)

cppgen.**emitEnum** (*item*, *name*, *f*, *indentation*)

cppgen.**emitMethodDeclaration** (*mname*, *params*, *f*, *className*)

cppgen.**emitStruct** (*item*, *name*, *f*, *indentation*)

cppgen.**emitStructMember** (*item*, *f*, *indentation*)

cppgen.**formalParameters** (*params*, *insertPortal*)

cppgen.**gatherMethodInfo** (*mname*, *params*, *itemname*)

cppgen.**generate_class** (*className*, *declList*, *parentC*, *parentCC*, *generatedCFiles*, *create_cpp_file*, *generated_hpp*, *generated_cpp*)

cppgen.**generate_cpp** (*project_dir*, *noisyFlag*, *jsondata*)

cppgen.**generate_demarshall** (*fmt*, *w*)

cppgen.**generate_marshall** (*pfmt*, *w*)

cppgen.**getNumeric** (*item*)

cppgen.**hasBitWidth** (*item*)

cppgen.**indent** (*f*, *indentation*)

class cppgen.**paramInfo** (*name*, *width*, *shifted*, *datatype*, *assignOp*)

cppgen.**typeBitWidth** (*item*)

cppgen.**typeCName** (*item*)

cppgen.**typeNumeric** (*item*)

4.4 discover_tcp

Functions

`connect_with_adb`(ipaddr)
`detect_network`()
`do_work`(start, end)
`int2ip`(addr)
`ip2int`(addr)
`open_adb_socket`(dest_addr)

discover_tcp.**connect_with_adb** (*ipaddr*) `discover_tcp.`**`detect_network`**`()`

`discover_tcp.`**`do_work`**`(`*start*, *end*`)`

`discover_tcp.`**`int2ip`**`(`*addr*`)`

`discover_tcp.`**`ip2int`**`(`*addr*`)`

`discover_tcp.`**`open_adb_socket`**`(`*dest_addr*`)`

4.5 makefilegen

4.6 util

Functions

`capitalize`(s)	
`createDirAndOpen`(f, m)	
`decapitalize`(s)	
`foldl`(f, x, l)	
`intersperse`(e, l)	
`splitBinding`(s)	

`util.`**`capitalize`**`(`*s*`)`

`util.`**`createDirAndOpen`**`(`*f*, *m*`)`

`util.`**`decapitalize`**`(`*s*`)`

`util.`**`foldl`**`(`*f*, *x*, *l*`)`

`util.`**`intersperse`**`(`*e*, *l*`)`

`util.`**`splitBinding`**`(`*s*`)`

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *pkgindex*
- *search*

BSV PACKAGE INDEX