Twitter Trivia!

Question: What is the formal IEEE standard name for IP-XACT?

Listen for answer and then tweet to @Synopsys

Get your game card for prize details and more chances to win!

Accellera IP-XACT Users Group Meeting
June 7th 2011

Twitter Trivia!

Question: What is the formal IEEE standard name for IP-XACT?

Listen for answer and then tweet to @Synopsys

Get your game card for prize details and more chances to win!

Today's Agenda

- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - ST Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE P1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC
- Wrap-up with an introduction to IP-XACT tool providers and where to fine more information about them

Today's Agenda

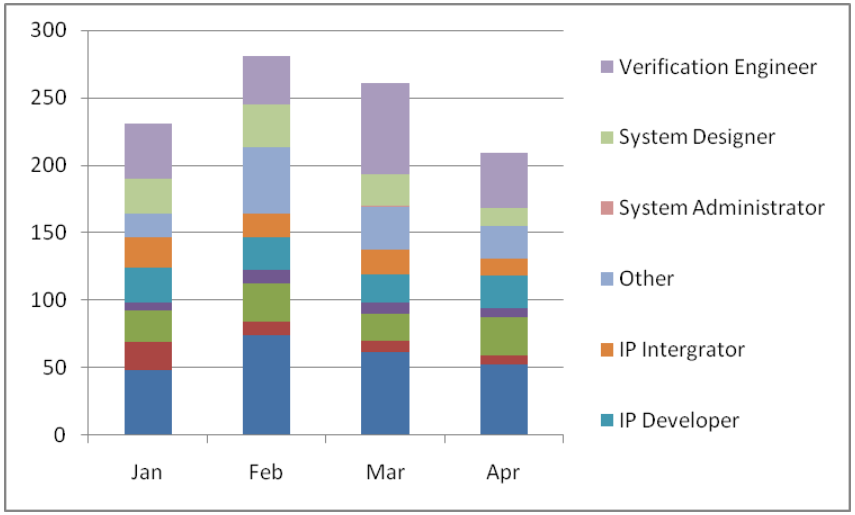
- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE P1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC
- Wrap-up with an introduction to IP-XACT tool providers and where to fine more information about them

What drove the creation of IP-XACT

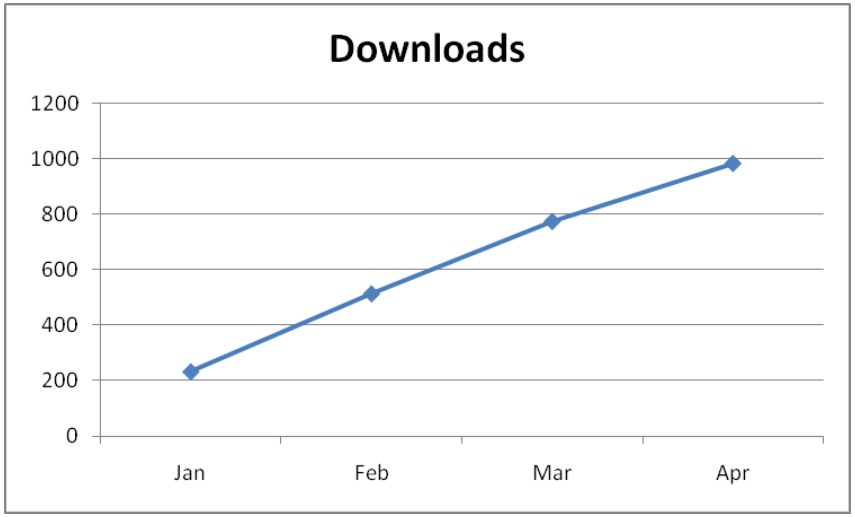
- Time to market requirements
  - First to market makes the most money
- Managing ever increasing design complexity
  - Just look at the designs you are doing now
- Reducing development and verification times / costs
  - Again, first to market makes the most money

Get IEEE 1685 Downloads

Different types of users



Total downloads in 2011

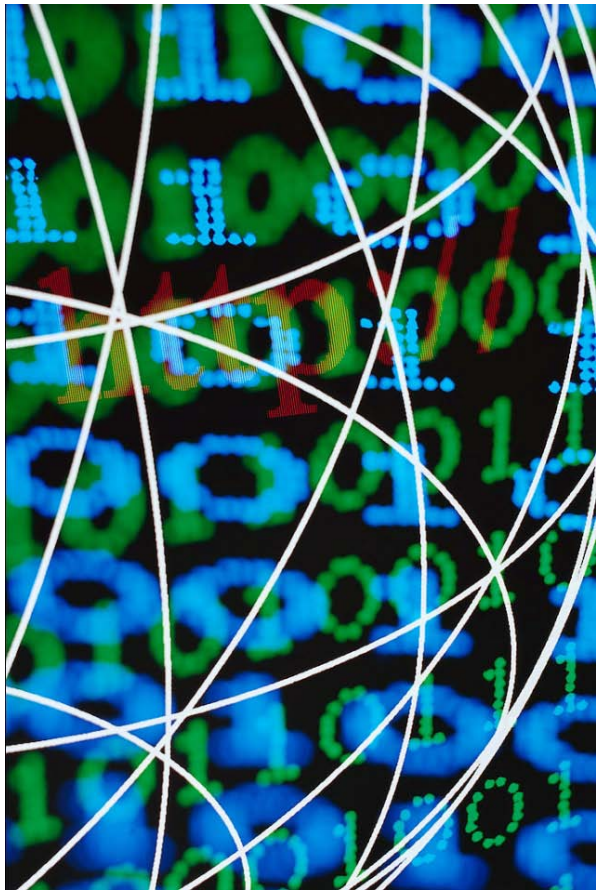


Components of an Efficient Design Flow

- Managing information
  - Organization
  - Content
- Processing information
  - Relate different pieces of information associated with one another to form a new result

Examples: Working with Information

- Documenting attributes of an IP component
  - Interfaces and signals
  - Parameters
  - Memory maps and registers
  - File sets
  - Etc…
- Processing Information
  - Assembly
  - Synthesis
  - Test insertion
  - Verification
  - Etc…



Managing and Processing Information Example

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!DOCTYPE plist (View Source for full doctype...)>
- <plist version="1.0">
  - <dict>
      <key>Major Version</key>
      <integer>1</integer>
      <key>Minor Version</key>
      <integer>1</integer>
      <key>Application Version</key>
      <string>10.2</string>
      <key>Features</key>
      <integer>5<integer>/
      <key>Show Content Ratings</key>
      <true />
      <key>Music Folder</key>
      <string>file://localhost/C:/Music/iTunes/iTunes%20Media/</string>
      <key>Library Persistent ID</key>
      <string>89E022BAE95775BD</string>
      <key>Tracks</key>
      <dict />
      <key>Playlists</key>
     - <array>
       - <dict>
          <key>Name</key>
          <string>Library</string>
          <key>Master</key>
          <true />
          <key>Playlist ID</key>
          <integer>63</integer>
          <key>Playlist Persistent ID</key>
          <string>EDC84D5DB88B6350</string>
          <key>Visible</key>
          <false />
          <key>All Items</key>
          <true />
         </dict>
       - <dict>
...
```

New media info added to XML file

User Interface data and media location

Same XML data used to control features supported in hardware

Today's Agenda

- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - ST Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE P1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC
- Wrap-up with an introduction to IP-XACT tool providers and where to fine more information about them

Today's Agenda

- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - ST Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE 1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC
- Wrap-up with an introduction to IP-XACT tool providers and where to fine more information about them

What is IEEE 1685-2009 (IP-XACT)

- An XML schema for language and vendor-neutral IP descriptions
- Includes a generator interface for “plug-in” functionality
- It has proven:
  - Low adoption costs
  - Value
  - The data needed to expand on for:
    - Verification
    - Software tools
    - Documentation
    - …

IP-XACT provides the information that you would expect to find in a data book in an electronic tool independent format so you can use the data to enhance your companies productivity

The IP-XACT Specification

- Is design language neutral
- Is design tool neutral
- Is efficient
- Is proven
- Is built on the existing XML (W3C) standard
- Includes a standardized API for generator integration (TGI)
- Validated and released in accordance with the IEEE policies

What is an XML Schema?

- The purpose of a schema is to define the legal building blocks of an XML document
  - It defines the document structure with a list of legal elements
- An XML schema defines:
  - Elements and attributes that can appear in a document
  - Which elements are child elements
  - The number and order of child elements
  - Whether an element is empty or can include text
  - Data types for elements and attributes
  - Default and fixed values for elements and attributes

XML 101

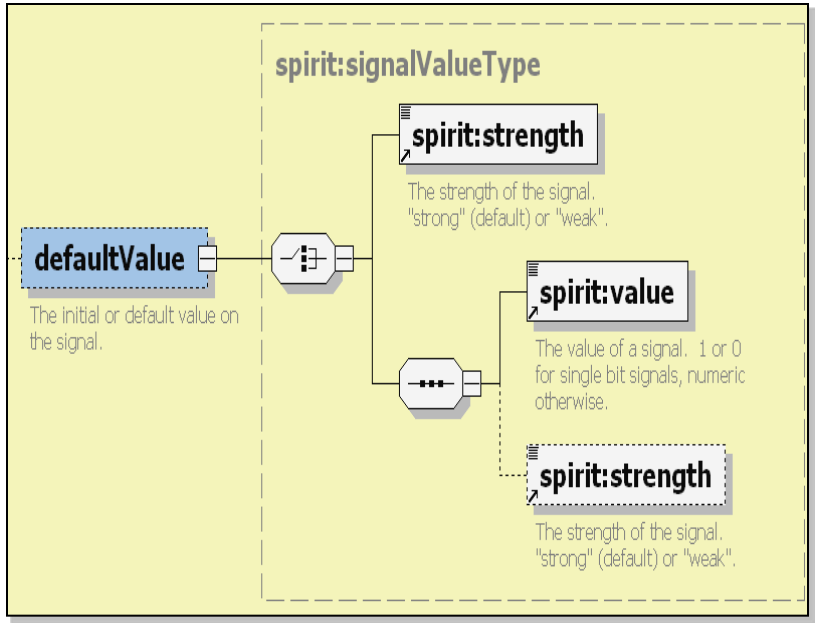
- **XML simplifies data sharing**
  - In the real world, computer systems and databases contain data in incompatible formats
  - XML data is stored in plain text format providing a software and hardware independent way of storing data
  - This makes it much easier to create data that different applications can share
- **XML simplifies data transport**
  - With XML, data can easily be exchanged between incompatible systems
  - One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet
  - Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications
- **XML simplifies platform changes**
  - Upgrading to new systems (hardware or software platforms), is always very time consuming. Large amounts of data must be converted and incompatible data is often lost
  - XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data
- **XML makes your data more available**

XML 101

- XML DOES NOT DO ANYTHING
  - XML was created to structure, store and transport information
- XML is just plain text
  - XML is nothing special. It is just plain text. Software that can handle plain text can also handle XML.
  - However, XML-aware applications can handle the XML tags specially.
  - The functional meaning of the tags depends on the nature of the application
- With XML you invent your own tags
  - XML has no pre-defined tags
  - XML is designed to allow things like… ***IP-XACT, an XML Schema***

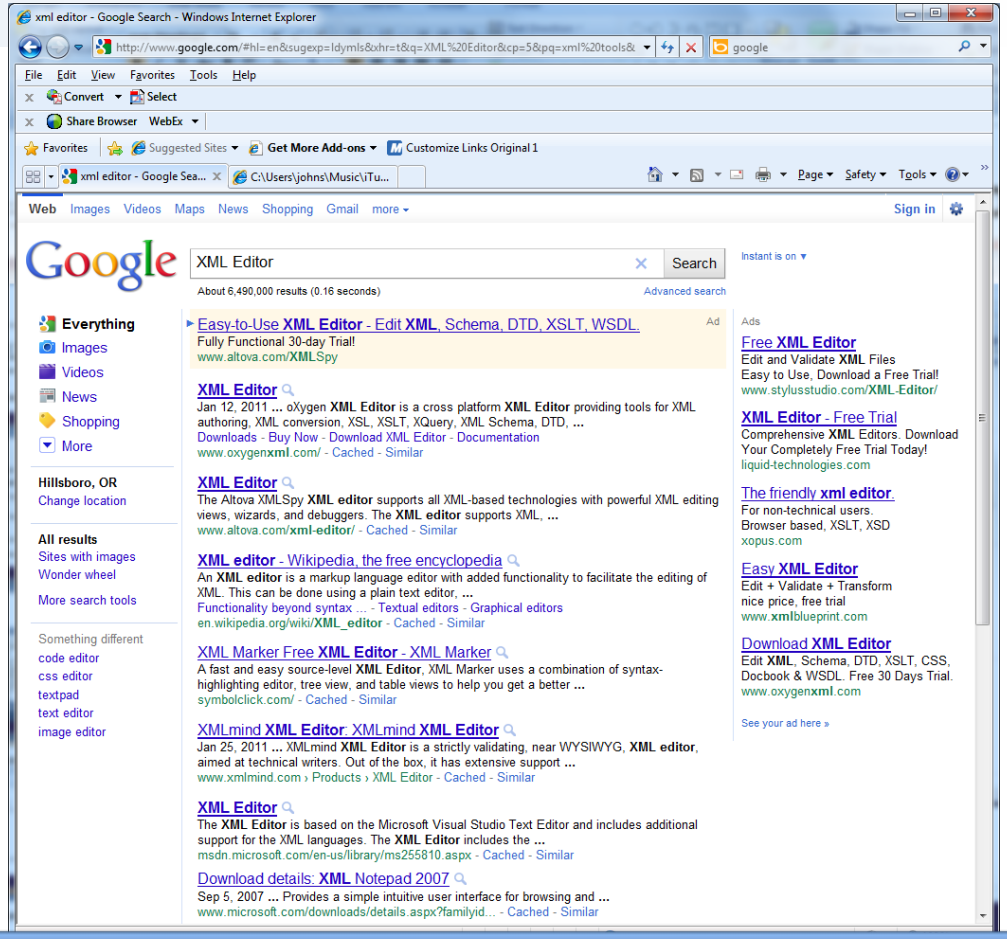
Interpreting the Schema

It's easy with the right tools!



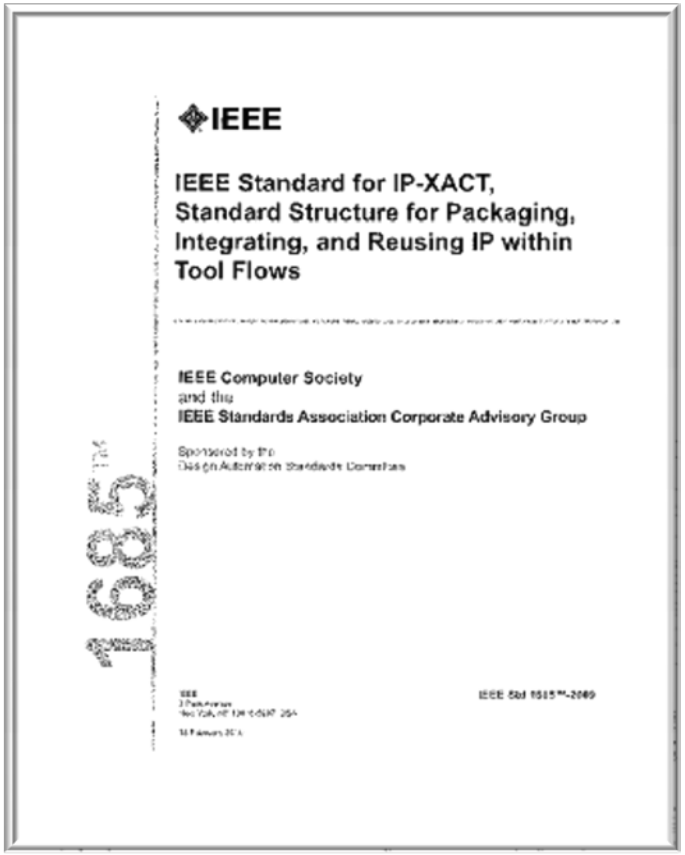

```
<xs:element name="defaultValue" minOccurs="0">
  <xs:annotation>
    <xs:documentation>The initial or default
    value on the signal<xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice>
      <xs:group ref="spirit:strengthGroup"/>
      <xs:sequence>
        <xs:element name="value">
          <xs:annotation>
            <xs:documentation>The value of a
    signal. 1 or 0 for single bit signals,
    unsigned numeric
    otherwise.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:group ref="spirit:strengthGroup"
    minOccurs="0"/>
      </xs:sequence>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

A Quick Google on “XML Editor”



6,490,000 results in 0.16 seconds

You can, of course, write IP-XACT…



IEEE

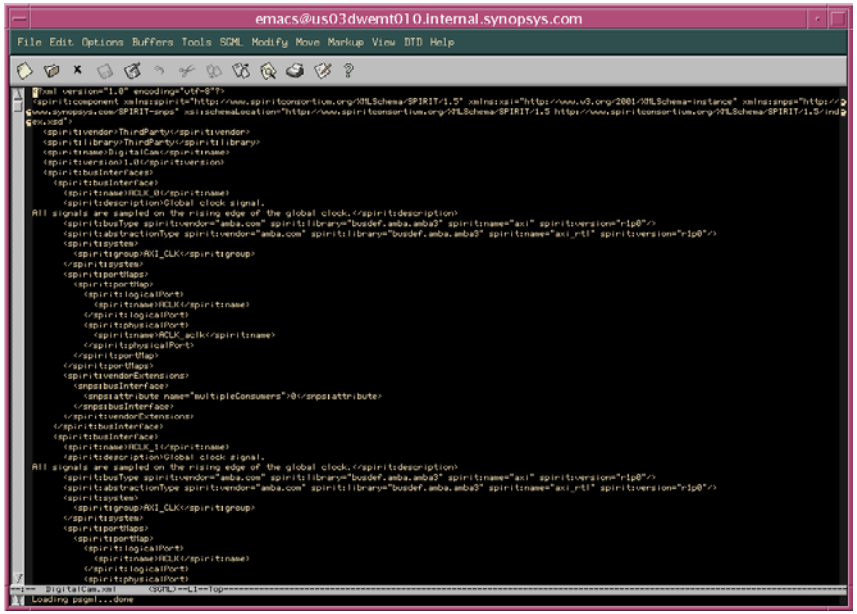
IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

IEEE Computer Society
and the
IEEE Standards Association Corporate Advisory Group

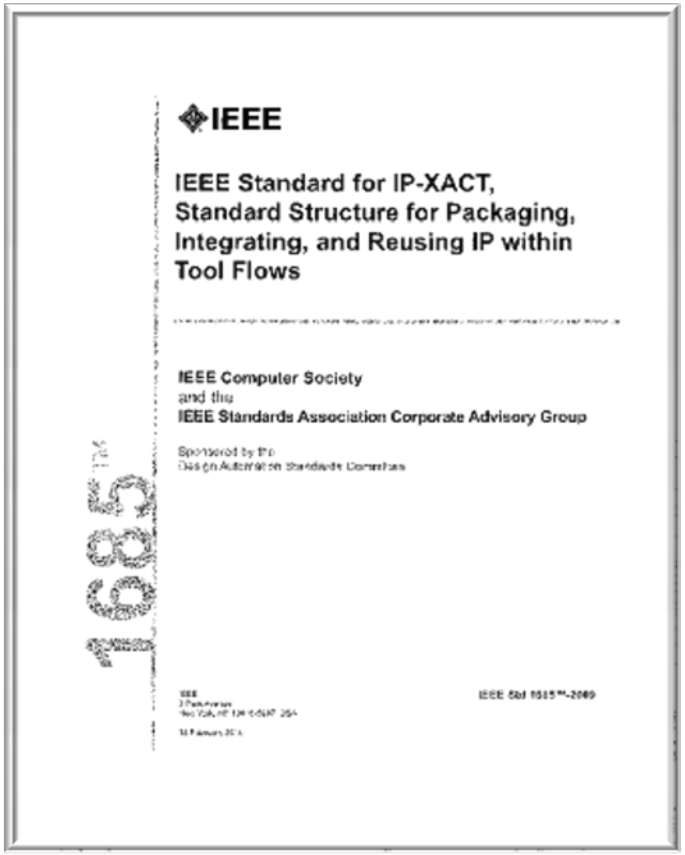
Sponsored by the
Design Automation Standards Committee

1685™

IEEE Std 1685™-2009



Or…
Use a tool that generates the IP-XACT


IEEE

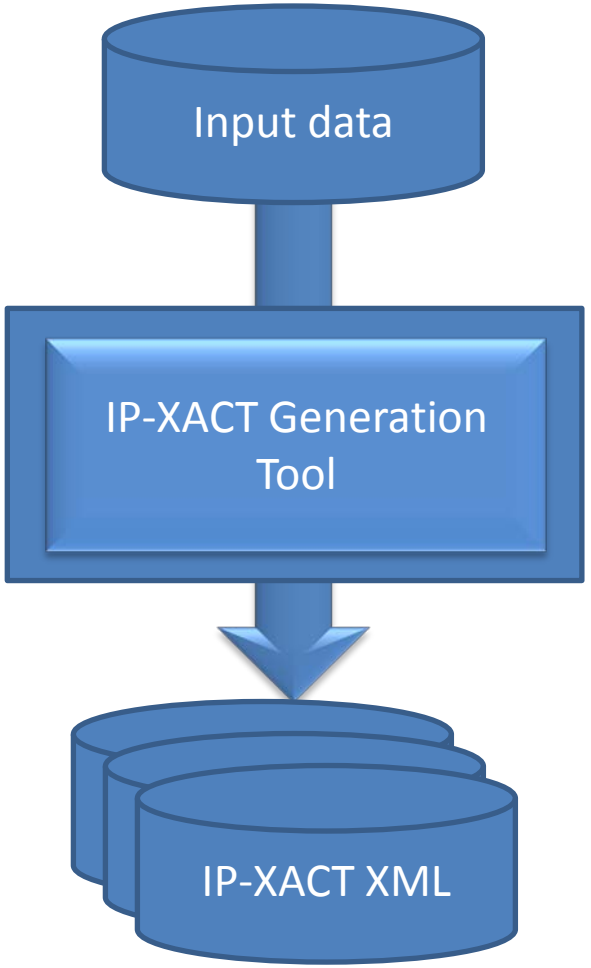
IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

IEEE Computer Society
and the
IEEE Standards Association Corporate Advisory Group

Sponsored by the
Design Automation Standards Committee

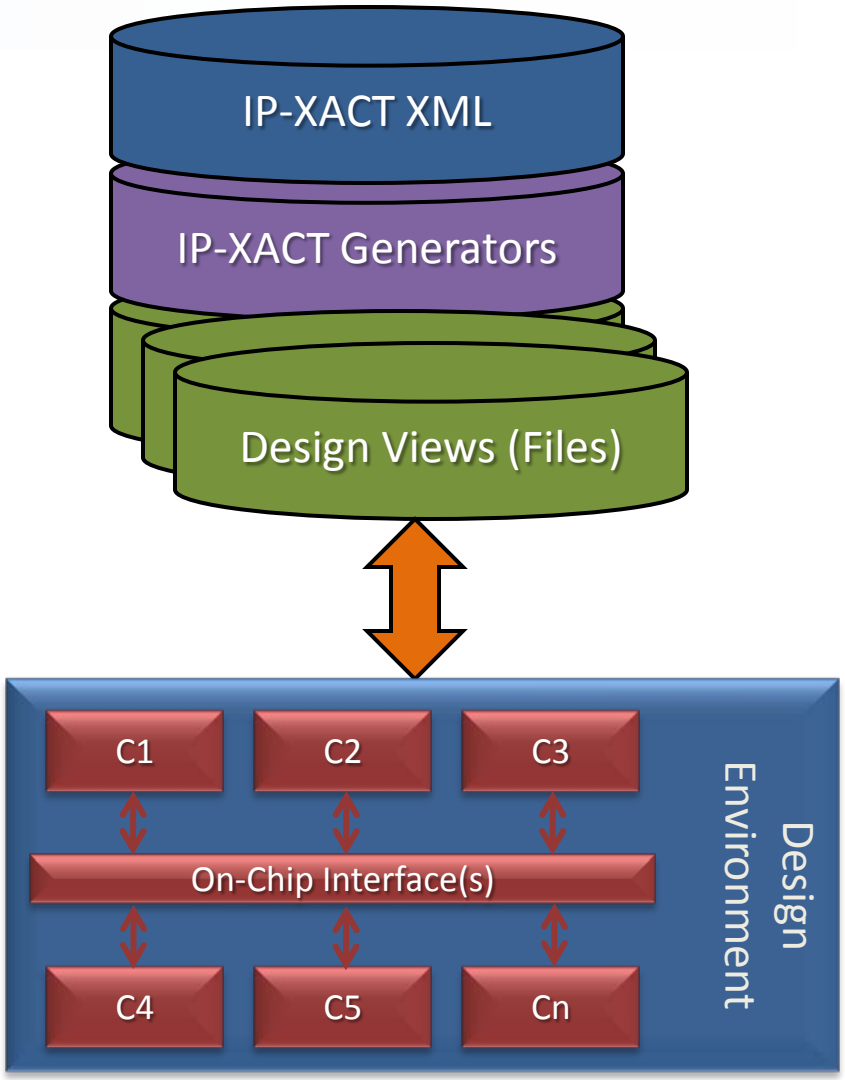
1685

IEEE Std 1685™-2009



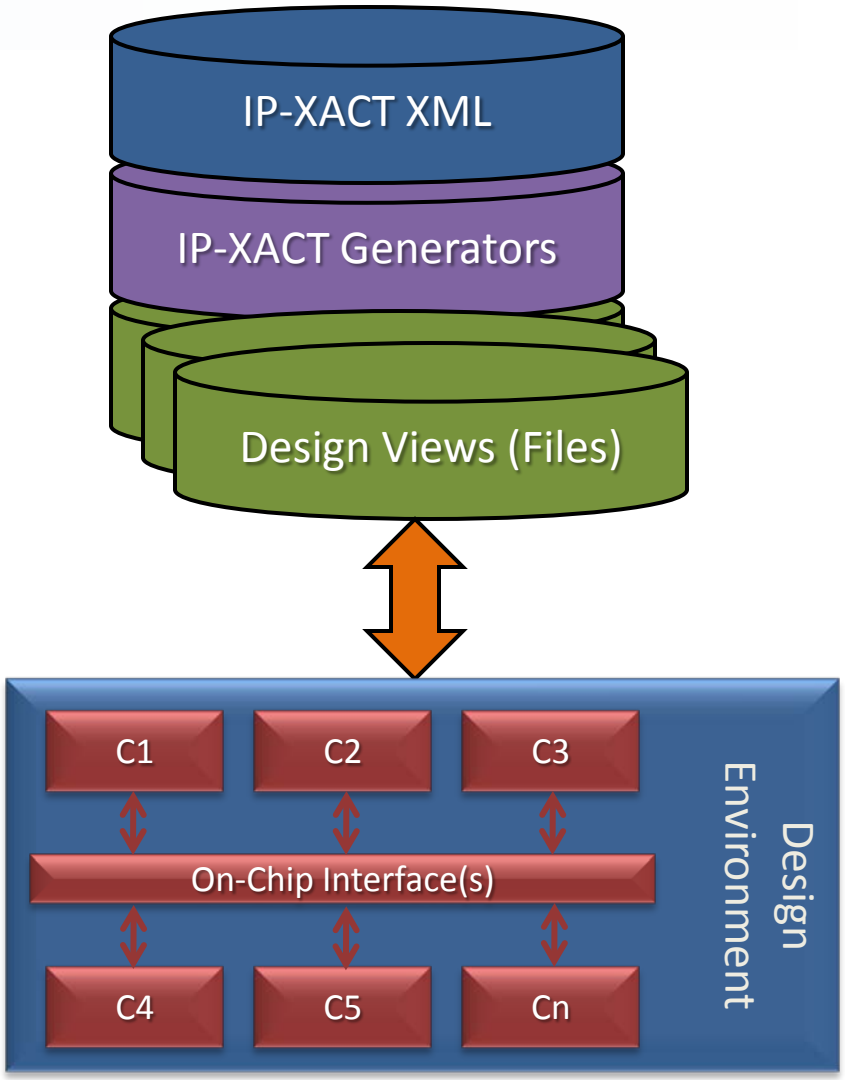
IP-XACT for Component Descriptions

- IP-XACT is an IEEE specification for documenting IP
  - Enables automated design creation and configuration
  - Tool independent
  - Machine readable
- Benefits
  - Documentation of all aspects of IP using XML databook format
  - Documentation of models in a quantifiable and language-independent way
  - Enables designers to deploy specialist knowledge in their design



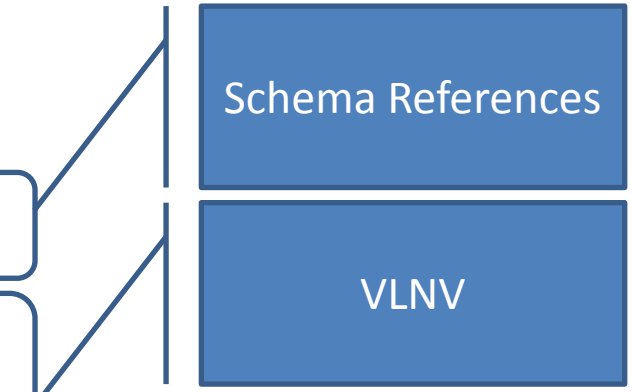
IP-XACT for Component Descriptions

- Component XML describes
  - Memory maps
  - Registers
  - Bus interfaces
  - Ports
  - Views (additional data files)
  - Parameters
  - Generators
  - File sets



UART Example (1)

```
<spirit:component
    xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009"
    xmlns:xsi="http://www/w3.org/2001/SMLSchema-instance"
    xmlns:<vendor>="http://<vendor provided path>"
    <spirit:vendor>IP Provider</spirit:vendor>
    <spirit:library>MyLibrary</spirit:library>
    <spirit:name>uart4APB</spirit:name>
    <spirit:version>1.00</spirit:version>
```



UART Example (2)

```
…
<spirit:busInterfaces>
<spirit:busInterface>
    <spirit:name>APB_Slave</spirit:name>
    <spirit:description>Bus slave side of the APB (Peripherals).</spirit:description>
    <spirit:busType spirit:vendor="amba.com" spirit:library="busdef.amba.amba3" spirit:name="apb" spirit:version="r1p0" />
    <spirit:abstractionType spirit:vendor="amba.com" spirit:library="busdef.amba.amba3" spirit:name="apb_rtl" spirit:version="r1p0" />
    <spirit:slave>
        <spirit:memoryMapRef spirit:memoryMapRef="uart_memory_map" />
    </spirit:slave>
    <spirit:portMaps>
        <spirit:portMap>
            <spirit:logicalPort>
                <spirit:name>paddr</spirit:name>
            </spirit:logicalPort>
            <spirit:physicalPort>
                <spirit:name>paddr</spirit:name>
            </spirit:physicalPort>
        </spirit:portMap>
        <spirit:portMap>
          …
<spirit:vendorExtensions>
    <VENDOR:What>
    …
</spirit:vendorExtensions>
…
```

Vendor Extensions

One or More Component Interfaces Defined

UART Example (3)

```
...
<spirit:memoryMaps>
    <spirit:name>uart_memory_map</spirit:name>
    <spirit:addressBlock>
        <spirit:name>uart_address_block</spirit:name>
        <spirit:baseAddress>0x0</spirit:baseAddress>
        <spirit:range>0x1000</spirit:range>
        <spirit:width>32</spirit:width>
        <spirit:register>
            <spirit:name>RBR_THR_DLL</spirit:name>
            <spirit:description>Receive Buffer Register</spirit:description>
            <spirit:addressOffset>0x0</spirit:addressOffset>
            <spirit:size>32</spirit:size>
            <spirit:volatile>true</spirit:volatile>
            <spirit:access>read-write</spirit:access>
            <spirit:reset>
                <spirit:value>0x0</spirit:value>
            </spirit:reset>
            <spirit:field>
                <spirit:name>rbr_thr_dll</spirit:name>
                <spirit:description><Detailed Description> </spirit:description>
                <spirit:bitOffset>0</spirit:bitOffset>
                <spirit:bitWidth>8</spirit:bitWidth>
                <spirit:access>read-write</spirit:access>
            </spirit:field>
        </spirit:register>
            ...
    </spirit:addressBlock>
      ...
</spirit:memoryMaps>
...
```

Component Memory Map

UART Example (4)

```
<spirit:choices>
    <spirit:choice>
        <spirit:name>APB_DATA_WIDTH</spirit:name>
        <spirit:enumeration spirit:text="8">8</spirit:enumeration>
        <spirit:enumeration spirit:text="16">16</spirit:enumeration>
        <spirit:enumeration spirit:text="32">32</spirit:enumeration>
    </spirit:choice>
        <spirit:choice>
        …
</spirit:choices>
<spirit:fileSets>
    <spirit:fileSet>
        <spirit:name>Hdl</spirit:name>
        <spirit:file>
            <spirit:name><Path to file></spirit:name>
            <spirit:fileType>verilogSource</spirit:fileType>
            <spirit:logicalName>work</spirit:logicalName>
        </spirit:file>
        <spirit:file>
        …
</spirit:fileSets
<spirit:parameters>
    <spirit:parameter>
        <spirit:name>APB_DATA_WIDTH</spirit:name>
        <spirit:description>APB_DATA_WIDTH: <Description>.</spirit:description>
        <spirit:value spirit:choiceRef="APB_DATA_WIDTH" spirit:configGroups="BasicConfig" spirit:format="long"…
    </spirit:parameter>
</spirit:parameters>
<spirit:vendorExtensions>
…
</spirit:vendorExtensions>
```

Enumerations related to the IP Block

File Locations related to the IP block

Parameters and info on the parameters

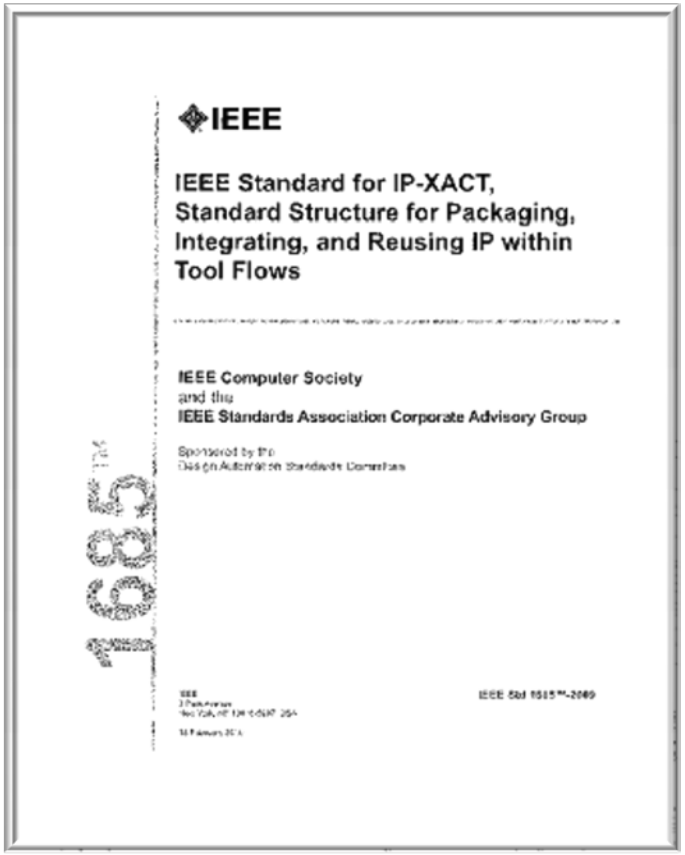
UART Example (5)

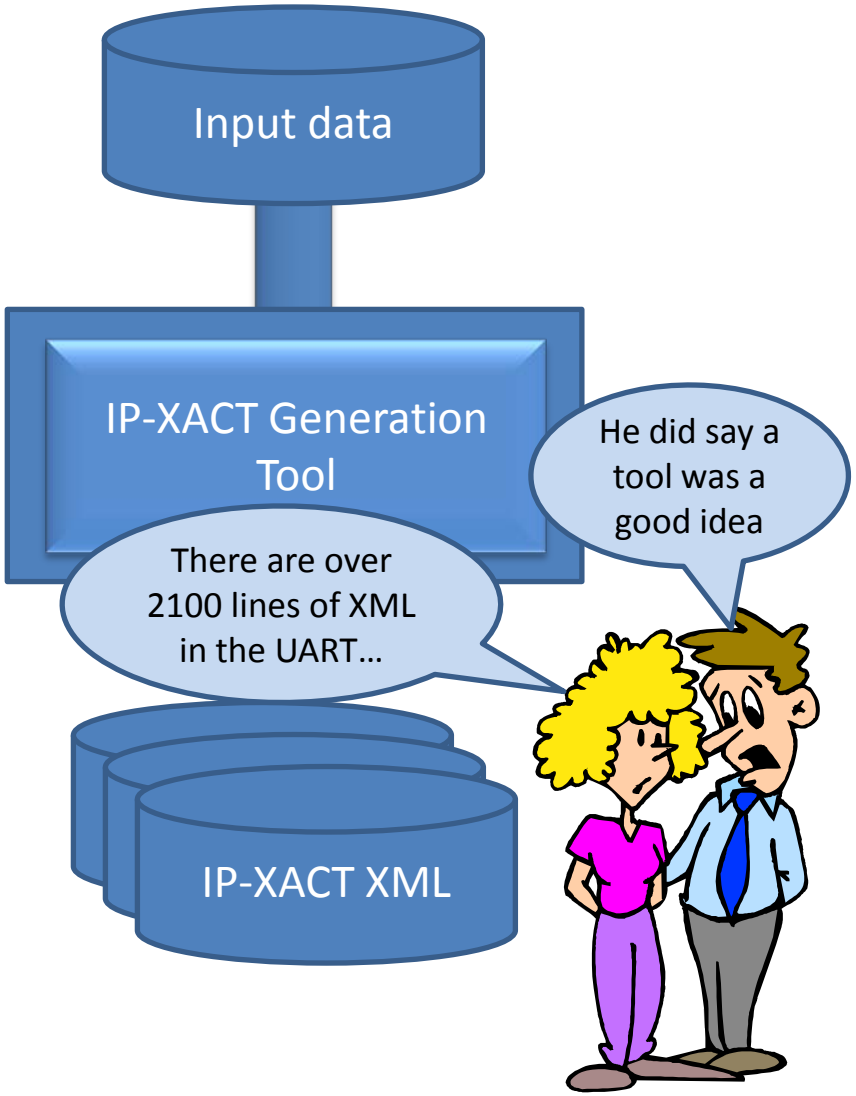
```
<spirit:model>
    <spirit:views>
        <spirit:view>
            <spirit:name>RTL</spirit:name>
            <spirit:envIdentifier>:*Synthesis:</spirit:envIdentifier>
            <spirit:language spirit:strict="true">verilog</spirit:language>
            <spirit:modelName>uart4apb</spirit:modelName>
            <spirit:fileSetRef>
                <spirit:localName>Hdl</spirit:localName>
            </spirit:fileSetRef>
            <spirit:constraintSetRef>default-constraints</spirit:constraintSetRef>
        </spirit:view>
    <spirit:ports>
        <spirit:port>
            <spirit:name>cts_n</spirit:name>
            <spirit:description>Description</spirit:description>
            <spirit:wire>
                <spirit:direction>in</spirit:direction>
                <spirit:constraintSets>
                    <spirit:constraintSet spirit:constraintSetId="default-constraints">
                        <spirit:driveConstraint>
                            <spirit:cellSpecification>
                                <spirit:cellClass>sequential</spirit:cellClass>
                            </spirit:cellSpecification>
                        </spirit:driveConstraint>
                        <spirit:timingConstraint spirit:clockName="pclk">20</spirit:timingConstraint>
                    </spirit:constraintSet>
                </spirit:constraintSets>
            </spirit:wire>
        </spirit:port>
    </spirit:views>
...
```

Different Views Defined (RTL, C/C++, GDSII, …_

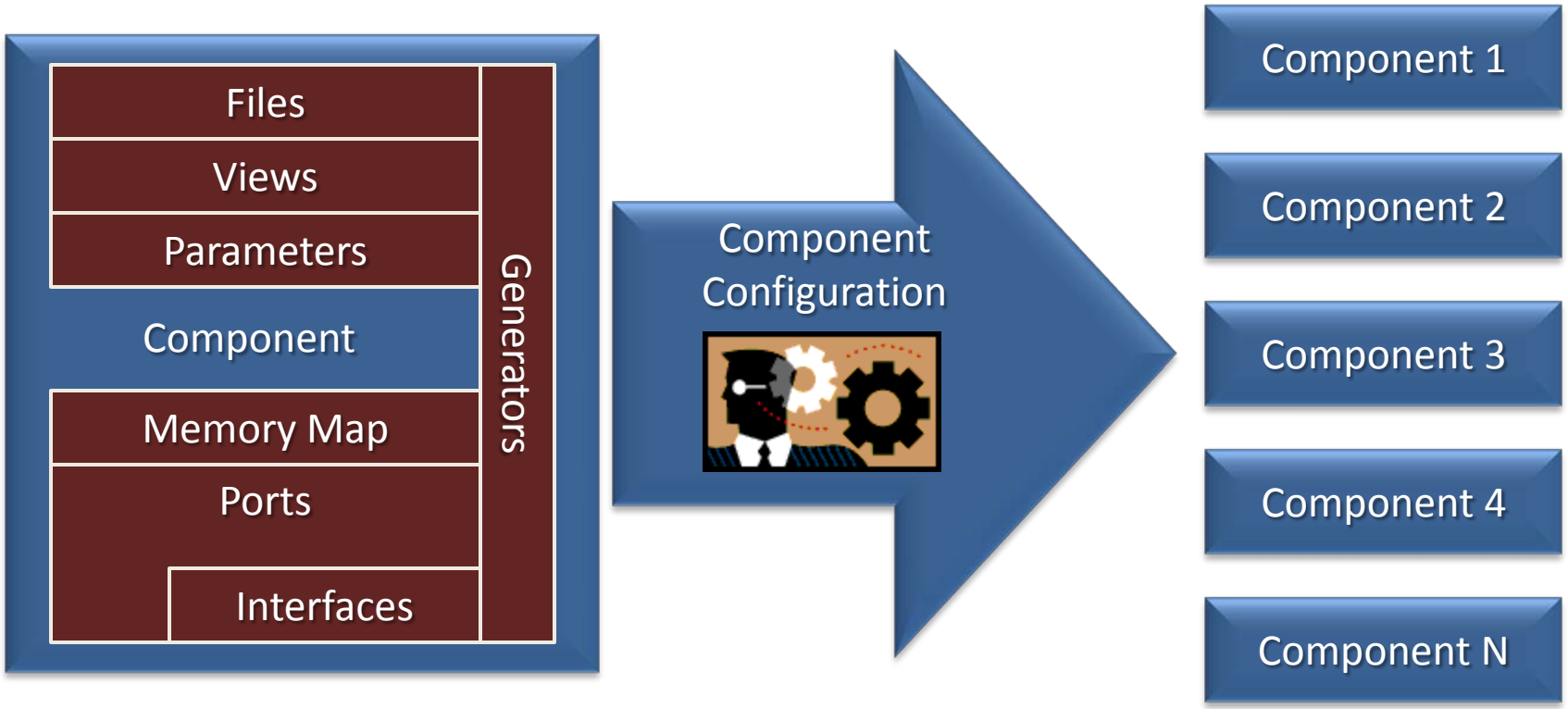
Ports of the component

As you can see
XML is “Ugly” but easy to parse

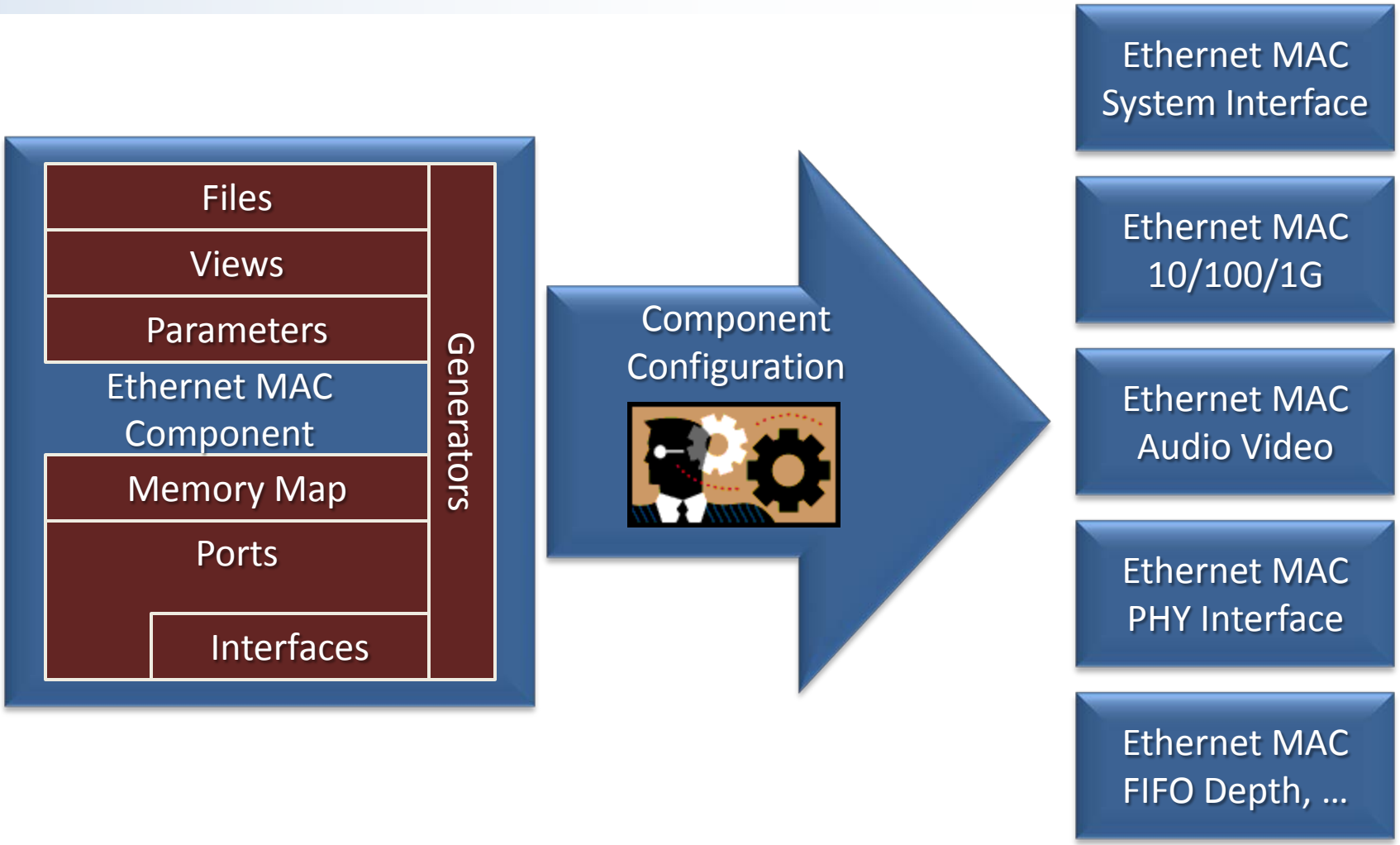




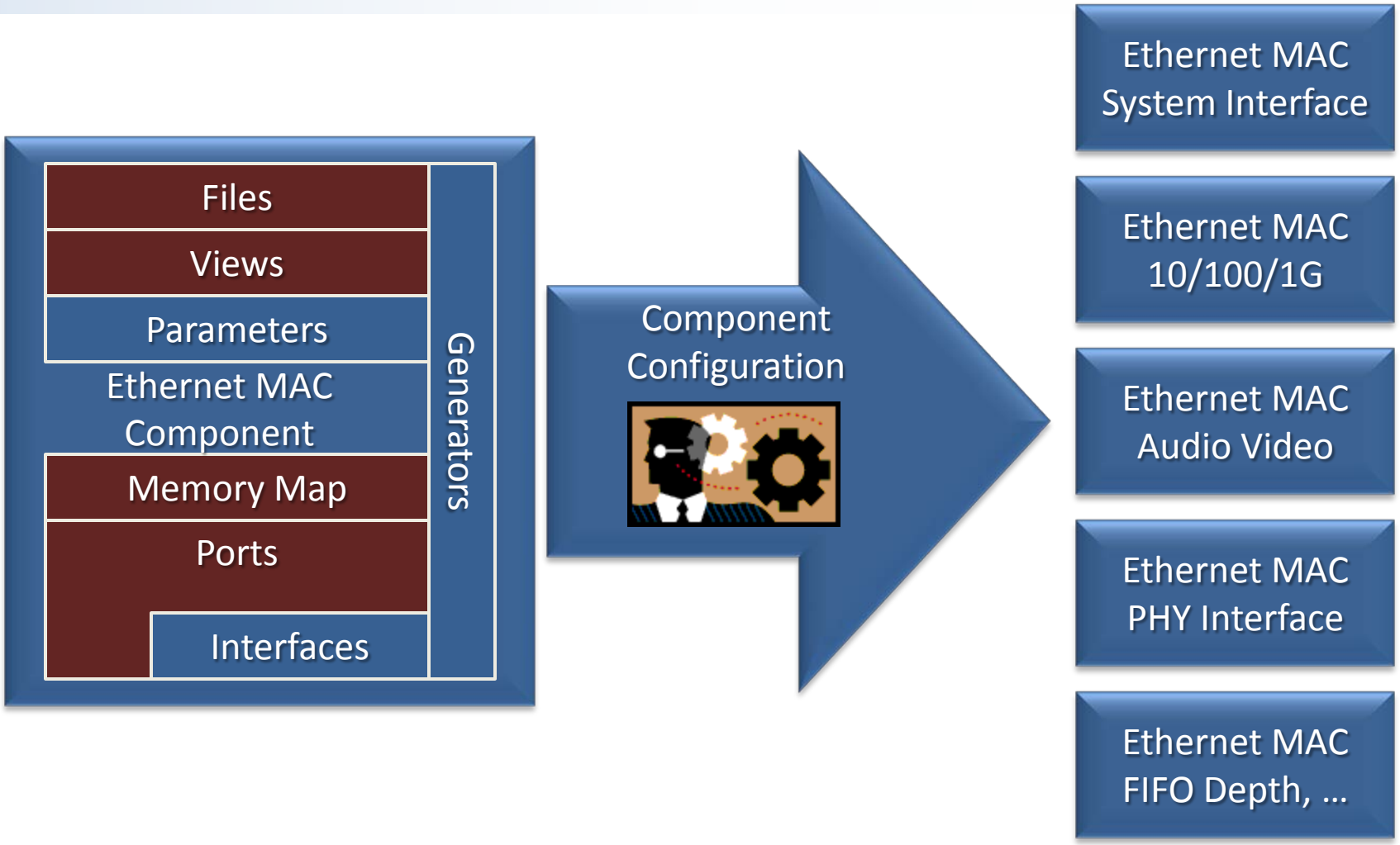
Configurable Components



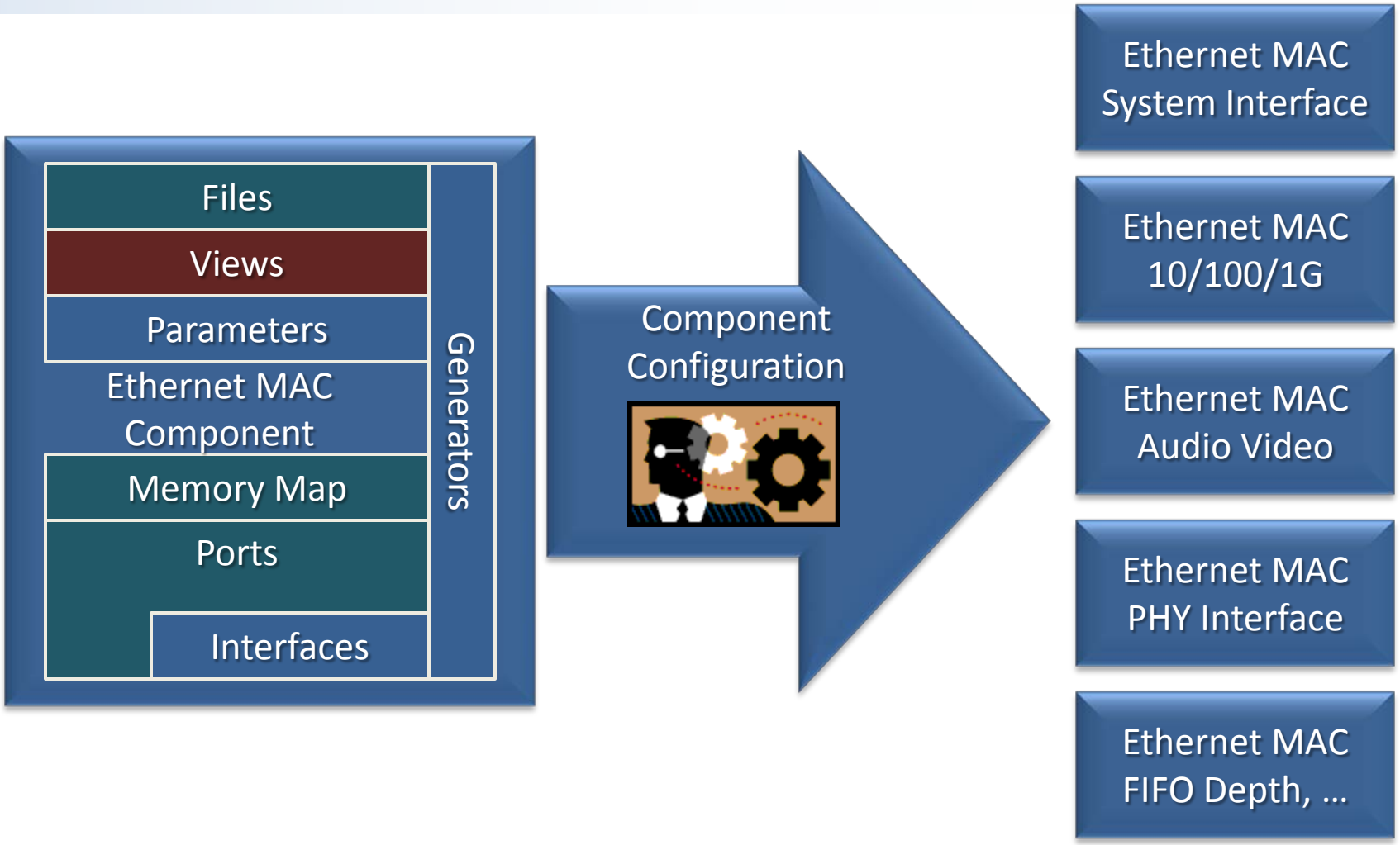
Configurable Components Example



Configurable Components Example

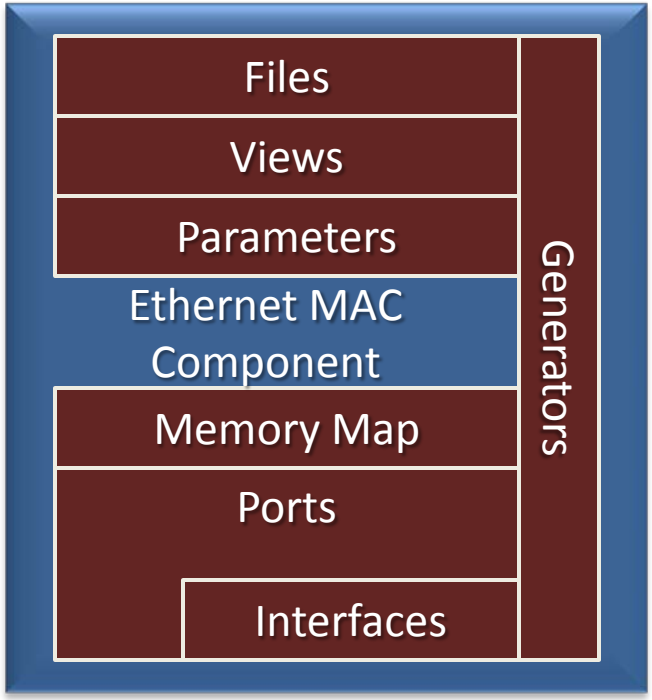


Configurable Components Example

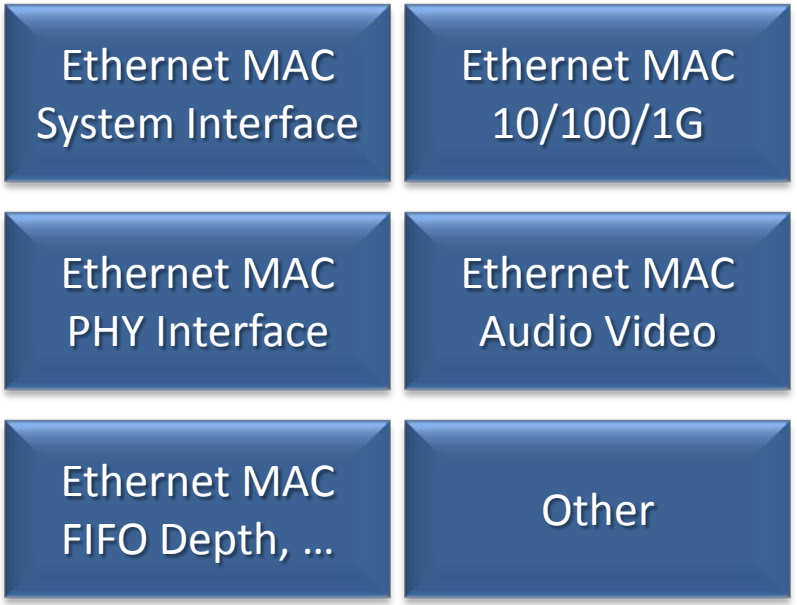


Your Choice Depending on the IP and your environment requirements

Single IP-XACT component with generators (if needed)



Generated or separate IP-XACT components

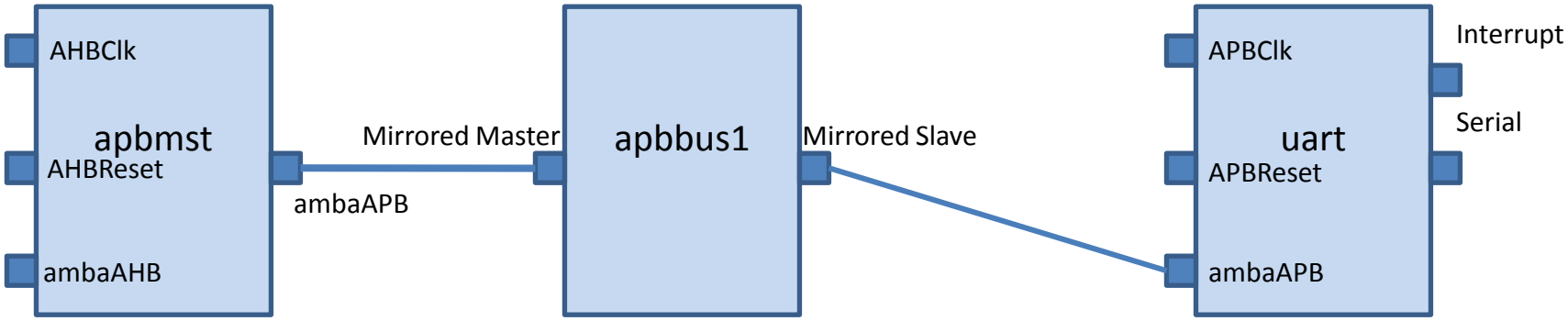


Different Types of IP Different Challenges

- Hard or non-configurable IP
  - Easiest to capture as it is a fixed configuration with a fixed foot-print and registers
    - Yes, register maps can be configured…
    - Yes, ports can have different “modes”
      - IP-XACT today, does not have any concept of dynamic configuration.
      - You can create multiple components
- Configurable IP (IIP or VIP)
  - Some common configuration constructs are not natively supported by the standard
    - Configurable tool and IP flow
    - Dynamic attribute values and complex expressions
    - pragmas or code substitution
  - There are tools that are designed to support this type of IP, some will be generator based, some will embed the functionality into the tool and generate a static IP-XACT view of a configurable component
  - You can write your own generators or restrict how you allow IP to be configured

Building Designs from components

Interfaces Enable Automated Assembly



APB Master → APB Bus → APB Slave

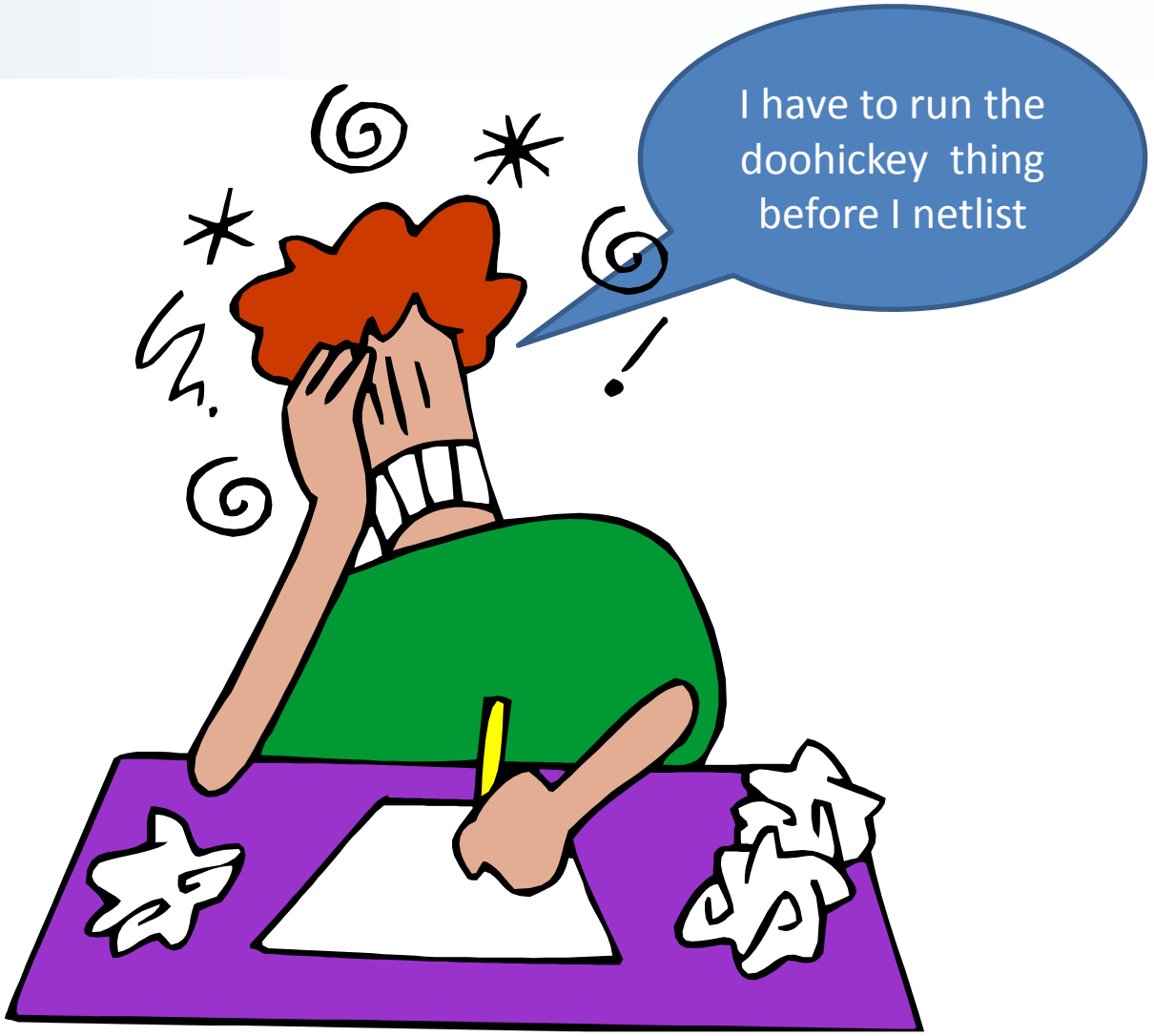
Tell me more about bus interfaces…

- A bus interface is made of:
  - A name (used by the interconnection in design)
  - A bus definition type => defines the protocol
  - An abstraction type => defines a level of abstraction (RTL, systemC, …)
  - A portMap => binds physical port to logical port
- A connection is only valid if it connects 2 interfaces with the same bus type
  - Thanks to the logical binding, individual physical ports connect automatically

Bus connection :Direct or not Direct

- The bus definition defines whether connections of bus interface can be direct :
  - Direct means that a Master bus interface can be connected directly to a Slave interface
    - Example : AXI, CLOCK, INTERRUPT
  - Not direct means an additional decoding or adaption is required
    - Example : AMBA2 AHB protocol with HREQ on Master side, HSEL on Slave side

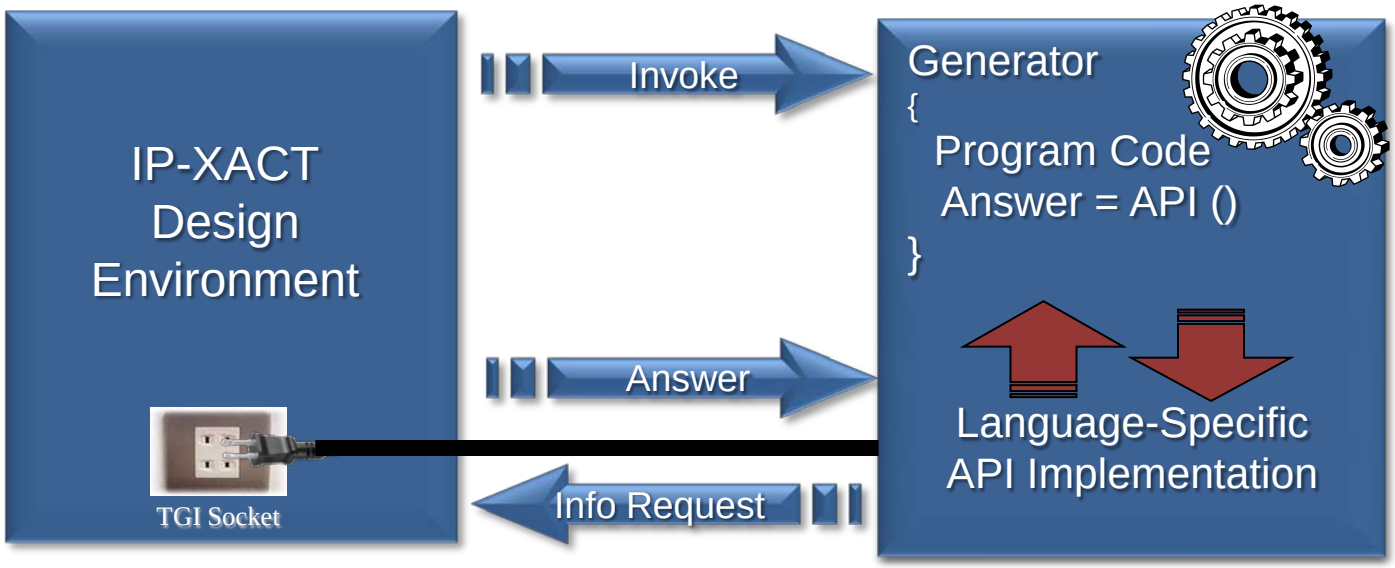
But…



What are Generators?

- Generators are program modules that process IP-XACT XML data into “something useful” for the design
- Examples might include
  - Component configuration
  - Verification hooks
  - Design creation
    - Netlists
    - Software
  - Tool configuration
  - Other things that you want to do unique to your design flow requirements
- To maximize the potential of reuse and simplify support each generator generally does one well defined task

Generators



Generators are program modules that process IP-XACT XML data into 'something useful' for the design

Key portable mechanism for encapsulating specialist design knowledge
Enables designers to deploy specialist knowledge in their design

Where Are Generators Specified

- Generators can be grouped into generator chains and invoked from the design environment
  - Combining individual generators enables the creation of custom functionality
  - Example: A generator chain may combine a generic HDL netlist generator with a simulator specific compilation command generator to build a custom command to run a simulation
- Generators can also be attached to a component
  - The activate only of the component is included in the design
  - Example: Check to see if a more up-to-date version of a component exists in a your companies RCS

A Simple Generator Example Loading a Generator

```
<?xml version="1.0" encoding="UTF-8"?>
<spirit:generatorChain
xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009/generator.xsd">
 <spirit:vendor>TestVendor</spirit:vendor>
 <spirit:library>TestLib</spirit:library>
 <spirit:name>SimpleGenerators</spirit:name>
 <spirit:version>1.0</spirit:version>

 <spirit:generator>
  <spirit:name>basicGenerator</spirit:name>
  <spirit:apiType>TGI</spirit:apiType>
  <spirit:generatorExe>bin/tgi_prog.tcl</spirit:generatorExe>
 </spirit:generator>

 <spirit:generator>
  <spirit:name>quickGenerator</spirit:name>
  <spirit:apiType>TGI</spirit:apiType>
  <spirit:generatorExe>bin/tgi_prog2.tcl</spirit:generatorExe>
 </spirit:generator>

</spirit:generatorChain>
```

A Simple Generator Example Setup

```
#!/bin/sh
# \
exec tclsh "$0" ${1+"$@"}
# Get location of script and tool root
set home [string trimright [file dirname [info script]] ./]
set home [file join [pwd] $home]
#
# Process command line
package require cmdline
set options [list \
            [list url.arg "" {SOAP service url}] \
            [list path.arg "" {Component generation path}] \
            [list nb_ports.arg "" {Number of ports}] \
           ]
array set arg [cmdline::getoptions argv $options]
if {$arg(url) eq ""} {  error "Must specify -url" }
if {$arg(path) eq ""} {  error "Must specify -path" }
if {$arg(nb_ports) eq ""} {  error "Must specify -nb_ports" }
# Load required packages
package require SOAP
package require SOAP::WSDL
# Load pre-defined procedures defined for SOAP communication.
source [file join $home tgidata.tcl]
# Update URL in data loaded from tgidata.tcl
insert_tgi_url $arg(url) ;
```

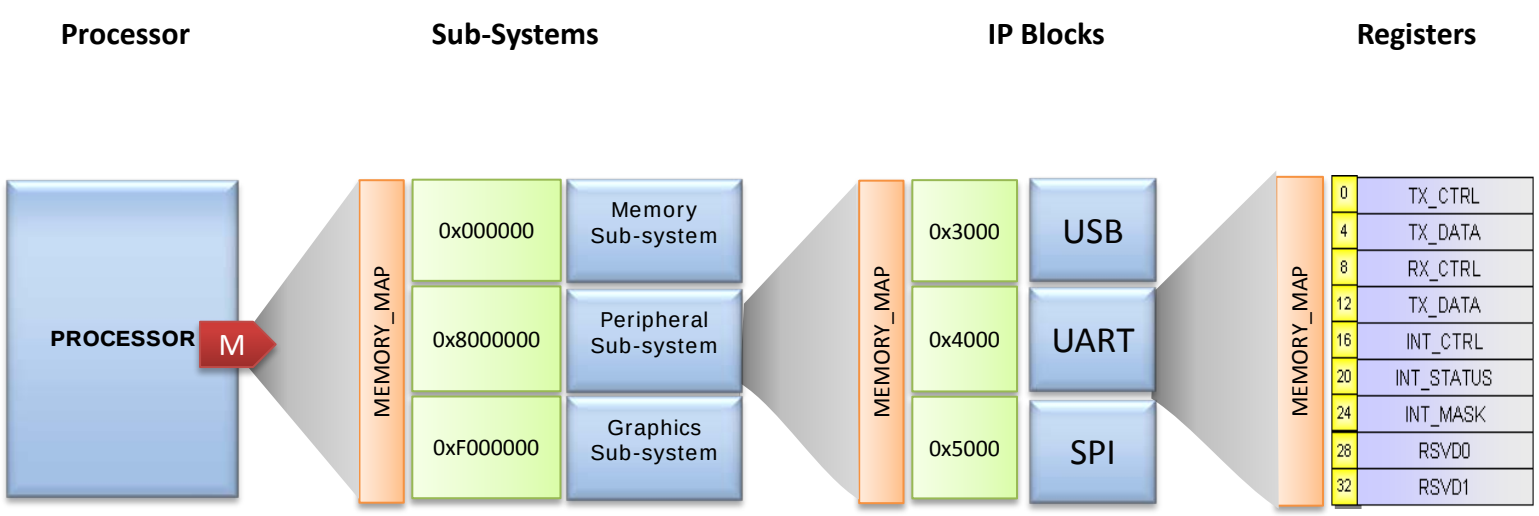
A Simple Generator Example
Do Something

```
namespace eval tgi {
 # Initialize
 init 1.5 fail "Client connected"
 message info "Running example generator"
 message info "Arguments: [array get arg]“
 # What instance invoked me?
 set instID [getGeneratorContextComponentInstanceID [info script]]
 # What is my name and VLNV?
 set name [getComponentInstanceName $instID]
 set vlnv [getComponentInstanceVLNV $instID]
 message info "Processing $name ($vlnv)“
 # Get XML of component.
 set xml [getComponentInstanceXML $instID]
 # Modify XML and write out a file with a new VLNV.
 regsub {PSIP} $xml {newPSIP} xml  ;# Change VLNV
 regsub {add_verify_} $xml {# add_verify_} xml  ;# Prevent inf loop
 set fname $arg(path)/new.xml
 set fid [open $fname "w"]
 puts $fid [set xml]
 close $fid
 regsub {PSIP} $vlnv {newPSIP} newVLNV
 if {![registerVLNV $fname 1]} {
   message error "Could not register $fname."
 }
 # Replace the component.
 set designID [getDesignID true]
 replaceComponentInstance $designID $instID $newVLNV
 # Indicate that we are done.
 end 0 "Client done"
}
```

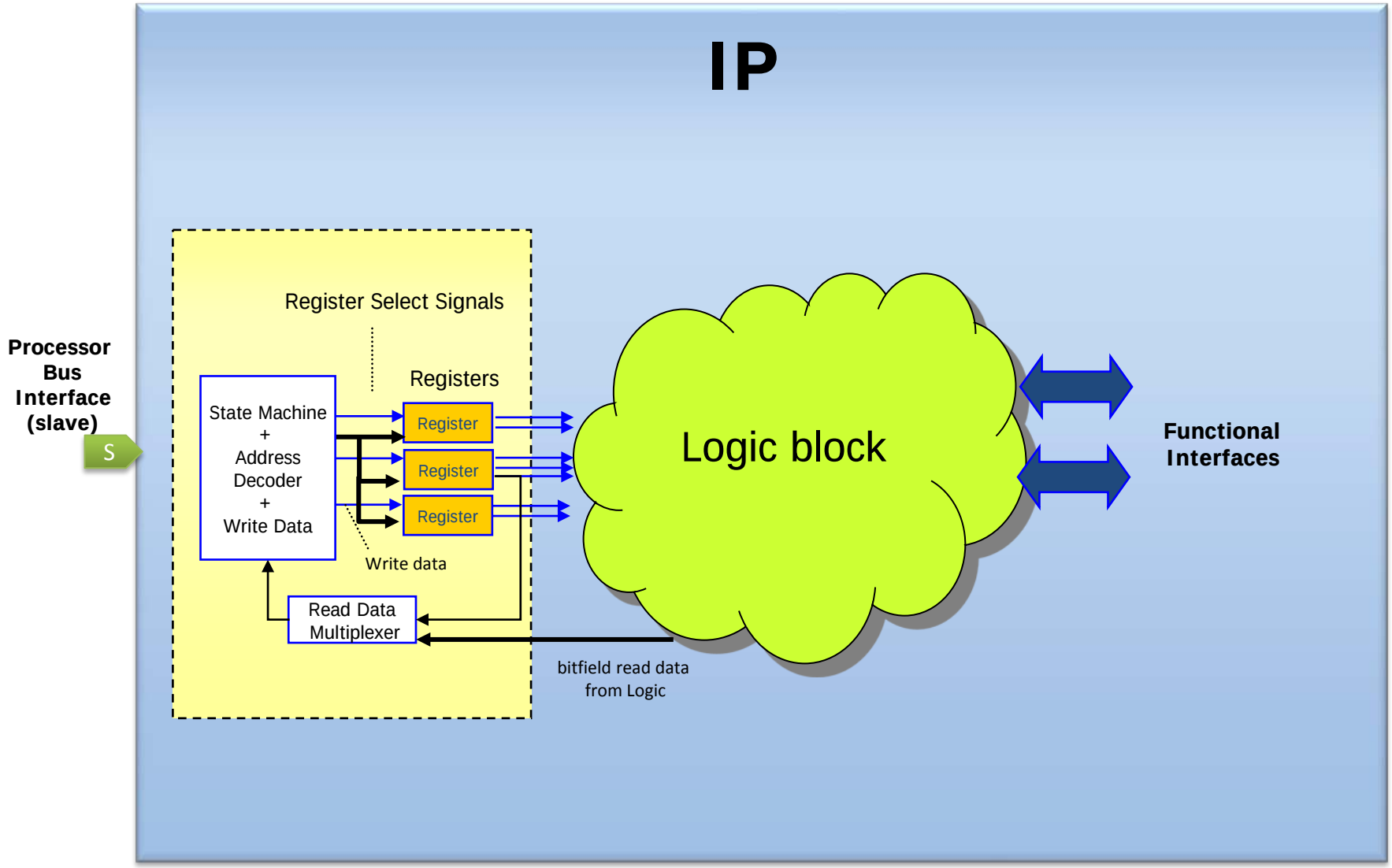
Today's Agenda

- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - ST Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE P1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC
- Wrap-up with an introduction to IP-XACT tool providers and where to fine more information about them

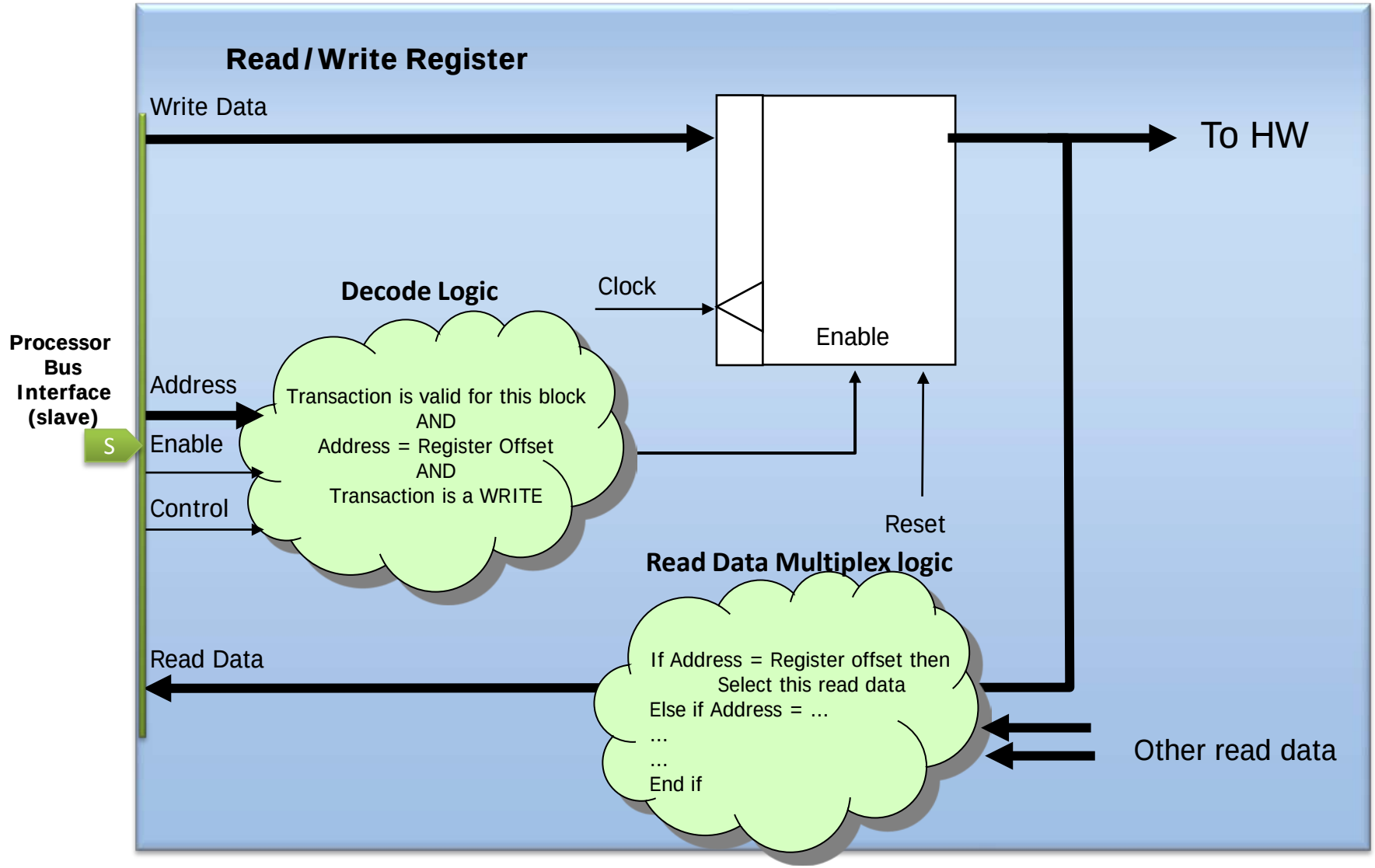
HW/SW interface



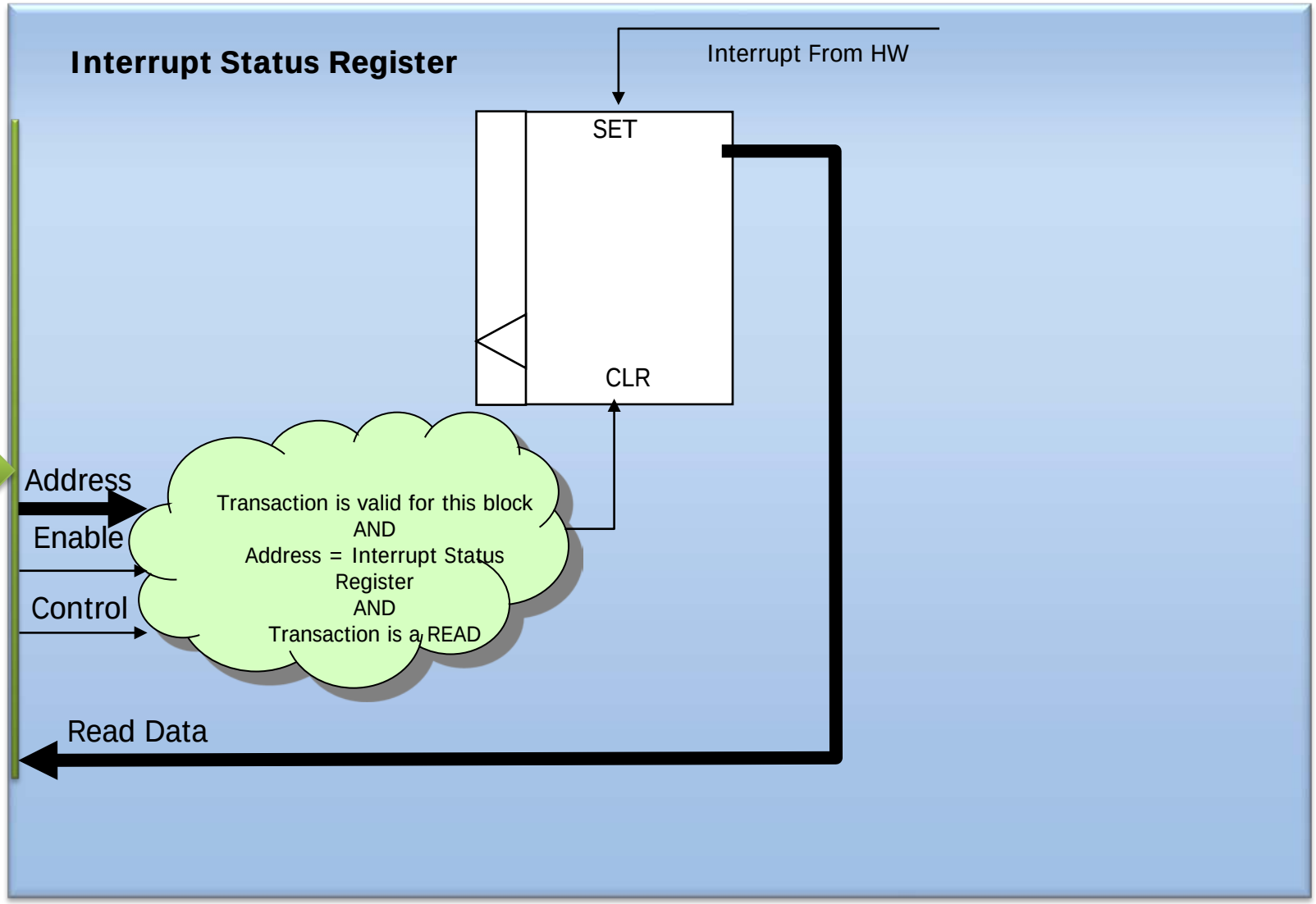
IP/Component



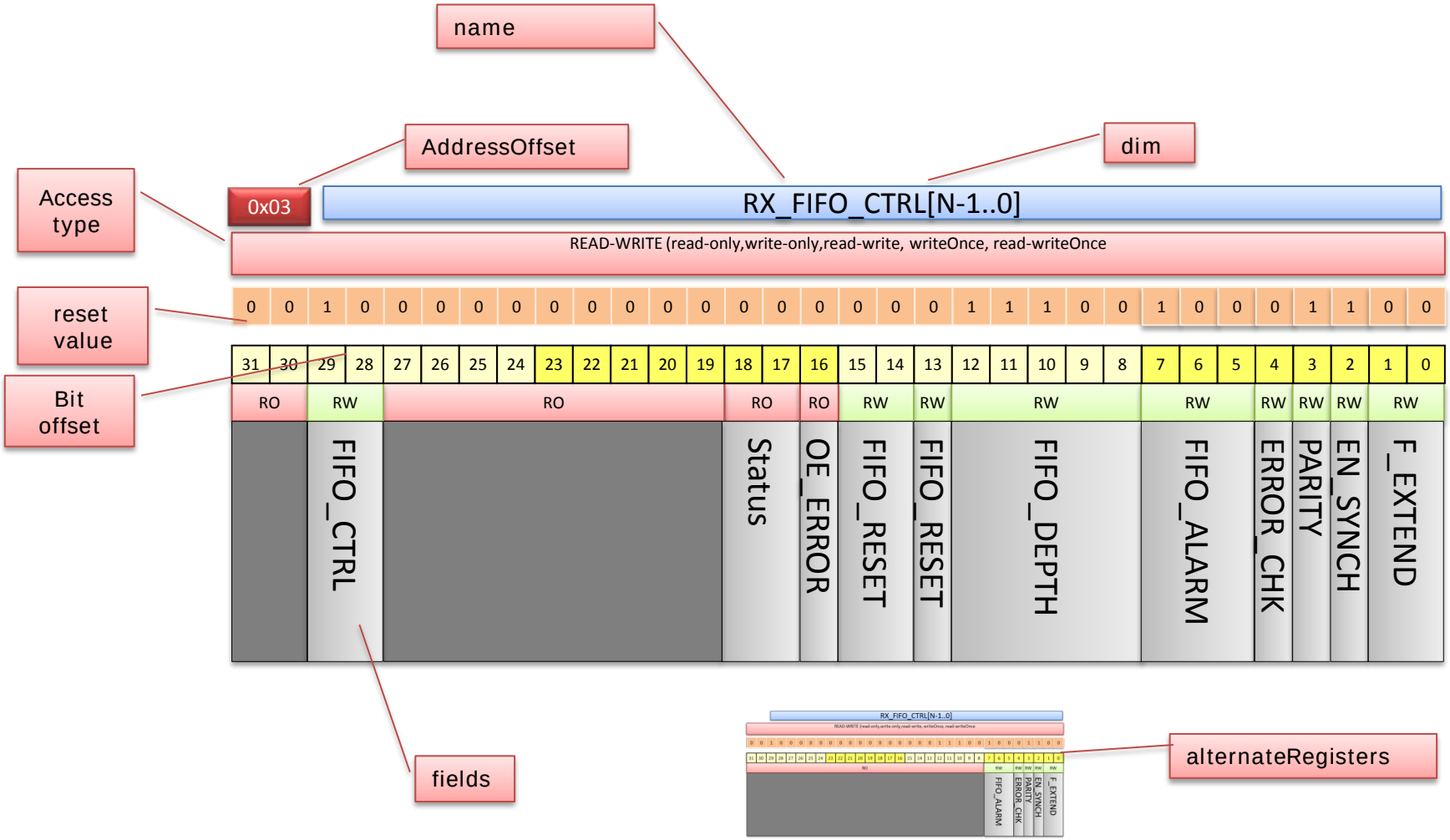
Read/Write Register



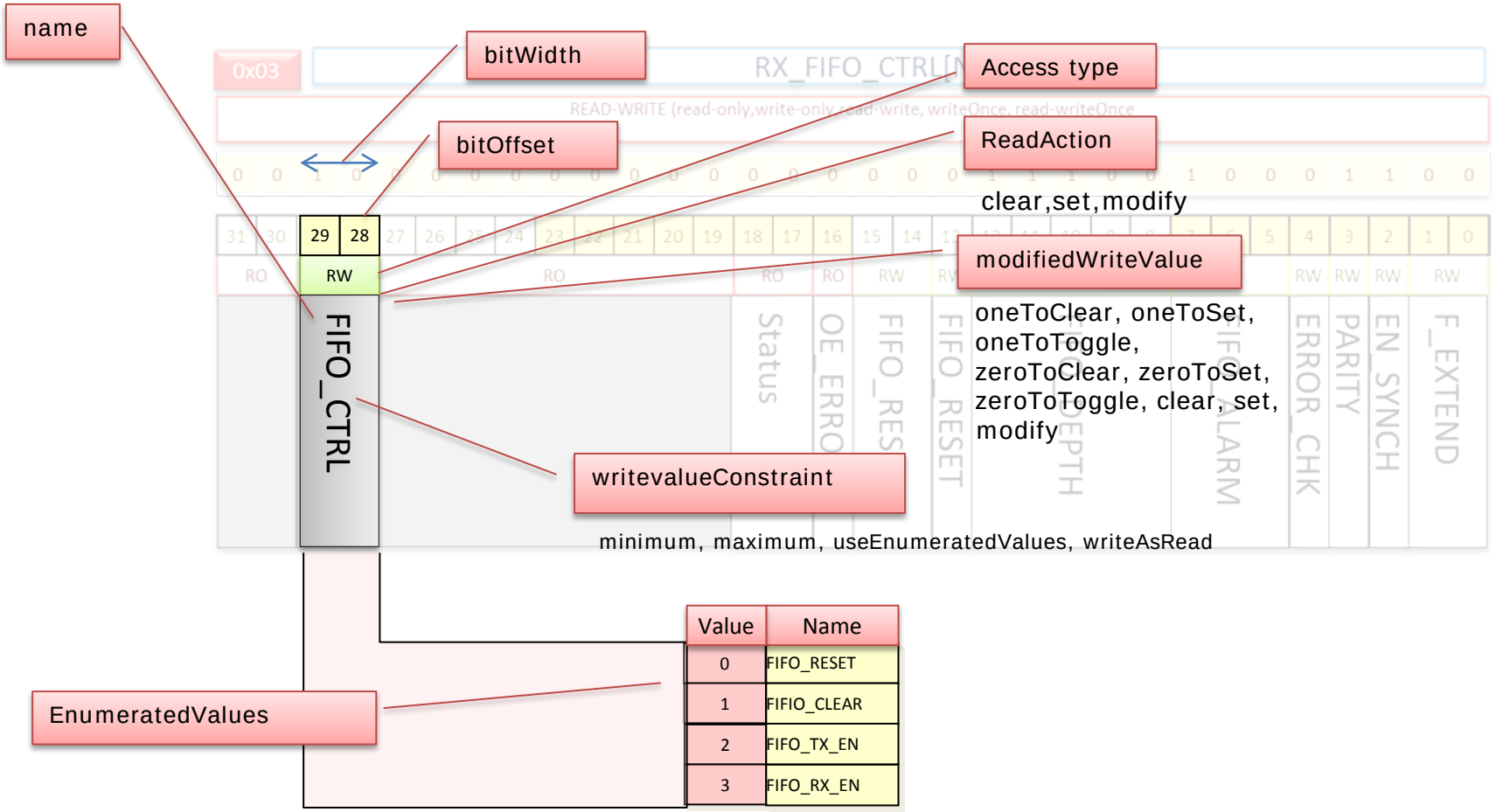
Interrupt Status Register



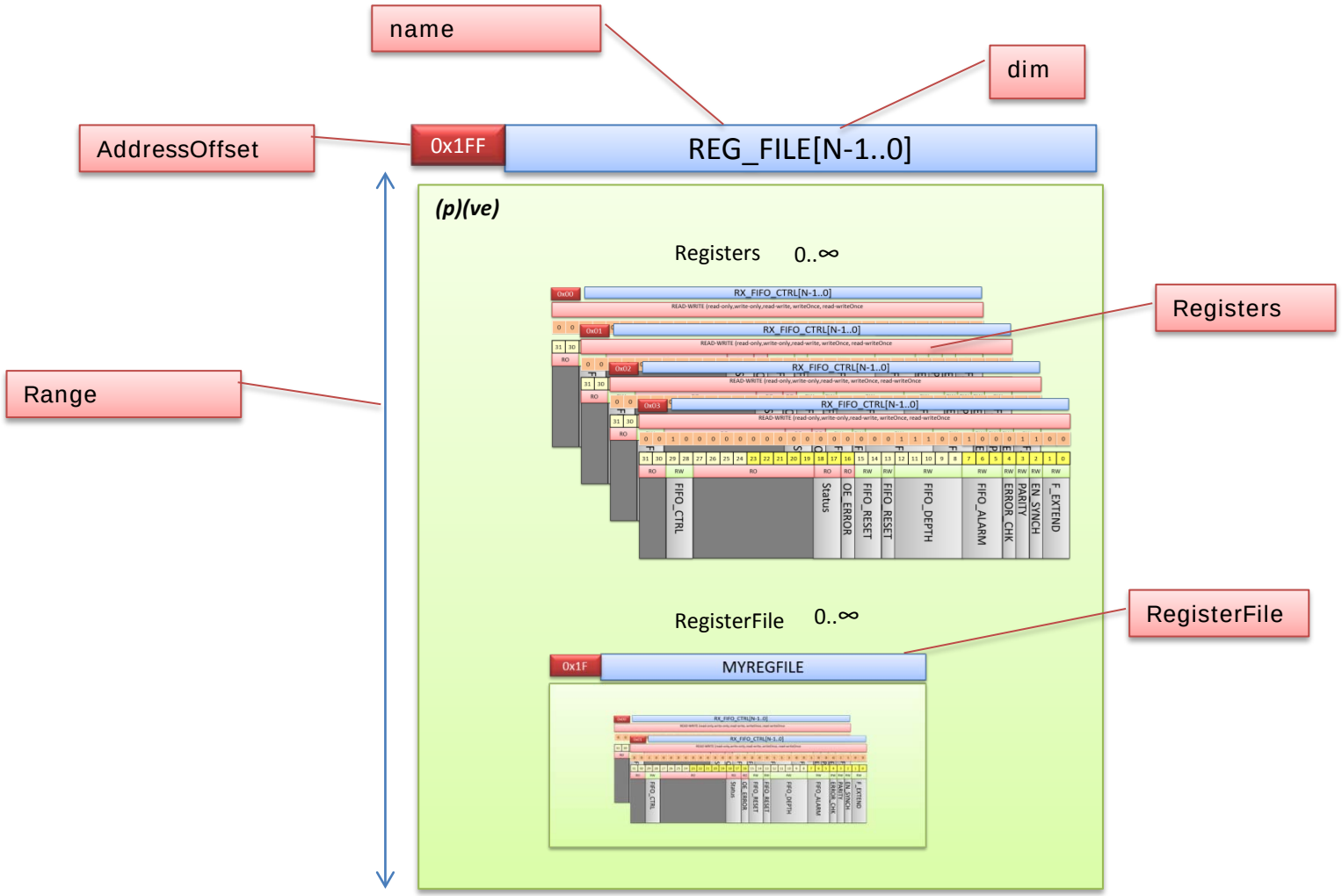
IP-XACT : Register



IP-XACT : Field

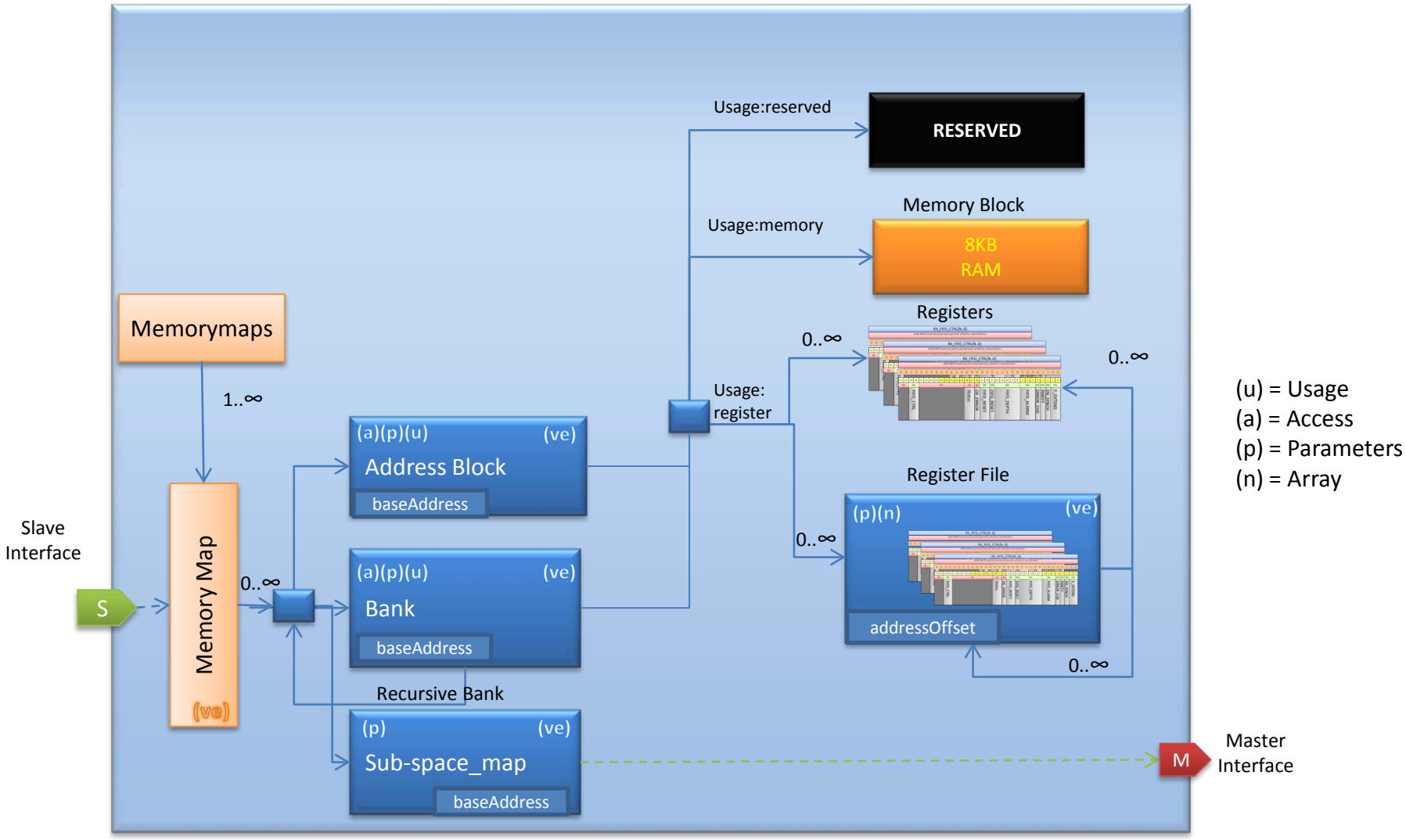


IP-XACT : RegisterFile



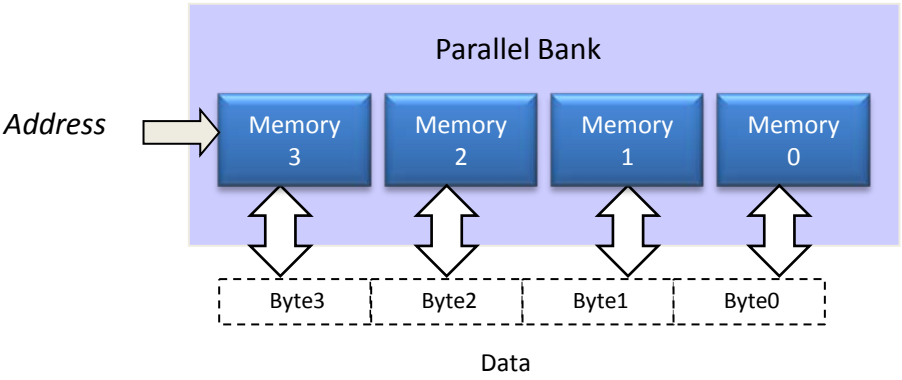
IP-XACT : Memory Mapping

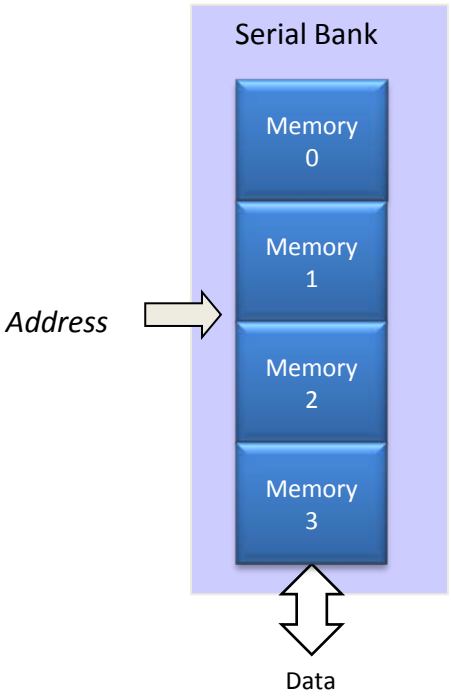
(From Slave)



(u) = Usage
(a) = Access
(p) = Parameters
(n) = Array

IP-XACT : Banks





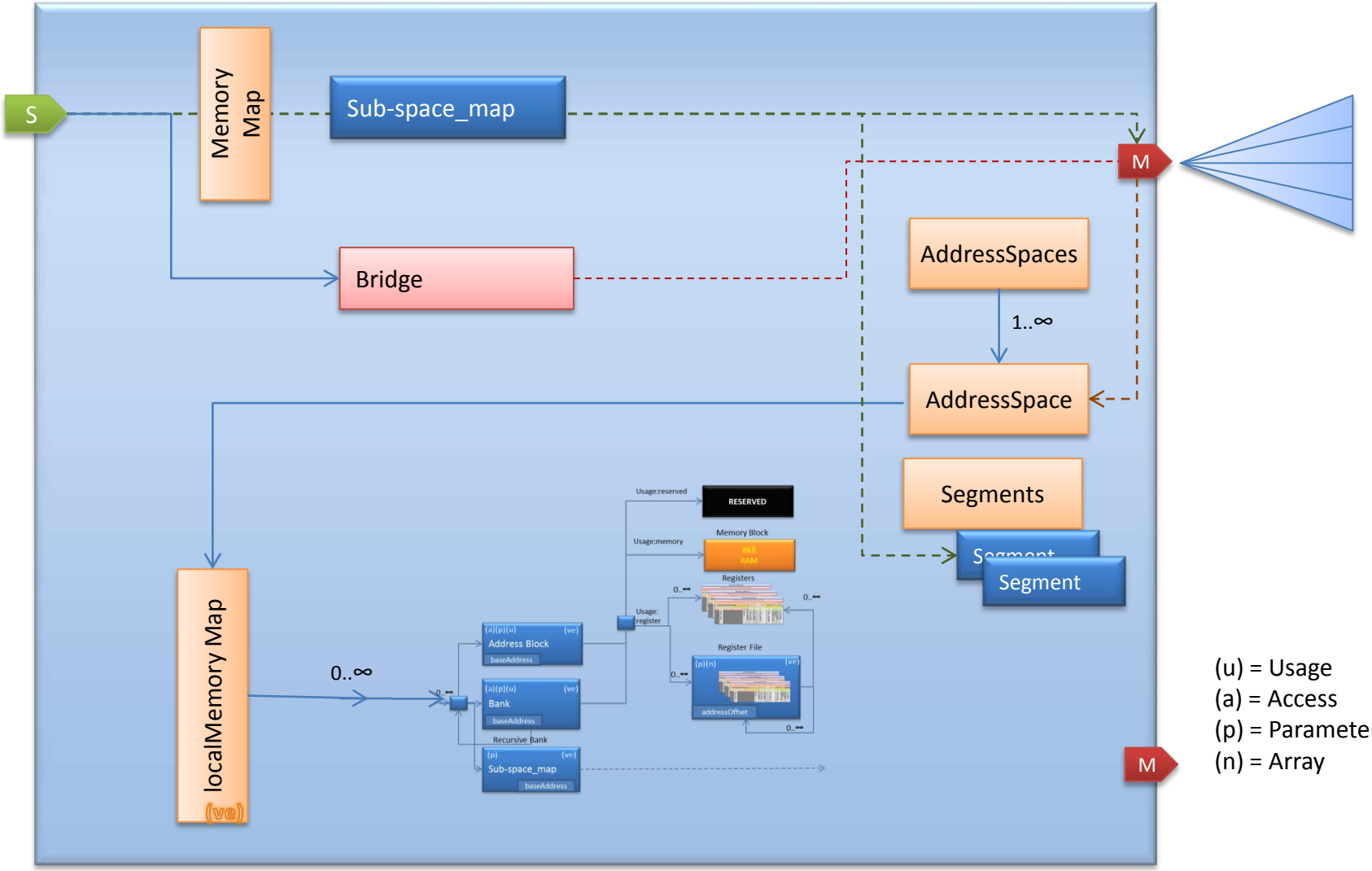
BANKS

A typical use case is where an overall memory structure is an aggregate of smaller memory units.

These can be arranged in **parallel** e.g. 4 x memories x 8 bits wide and all share the same address or they can be arranged **serially** and have contiguous addresses.

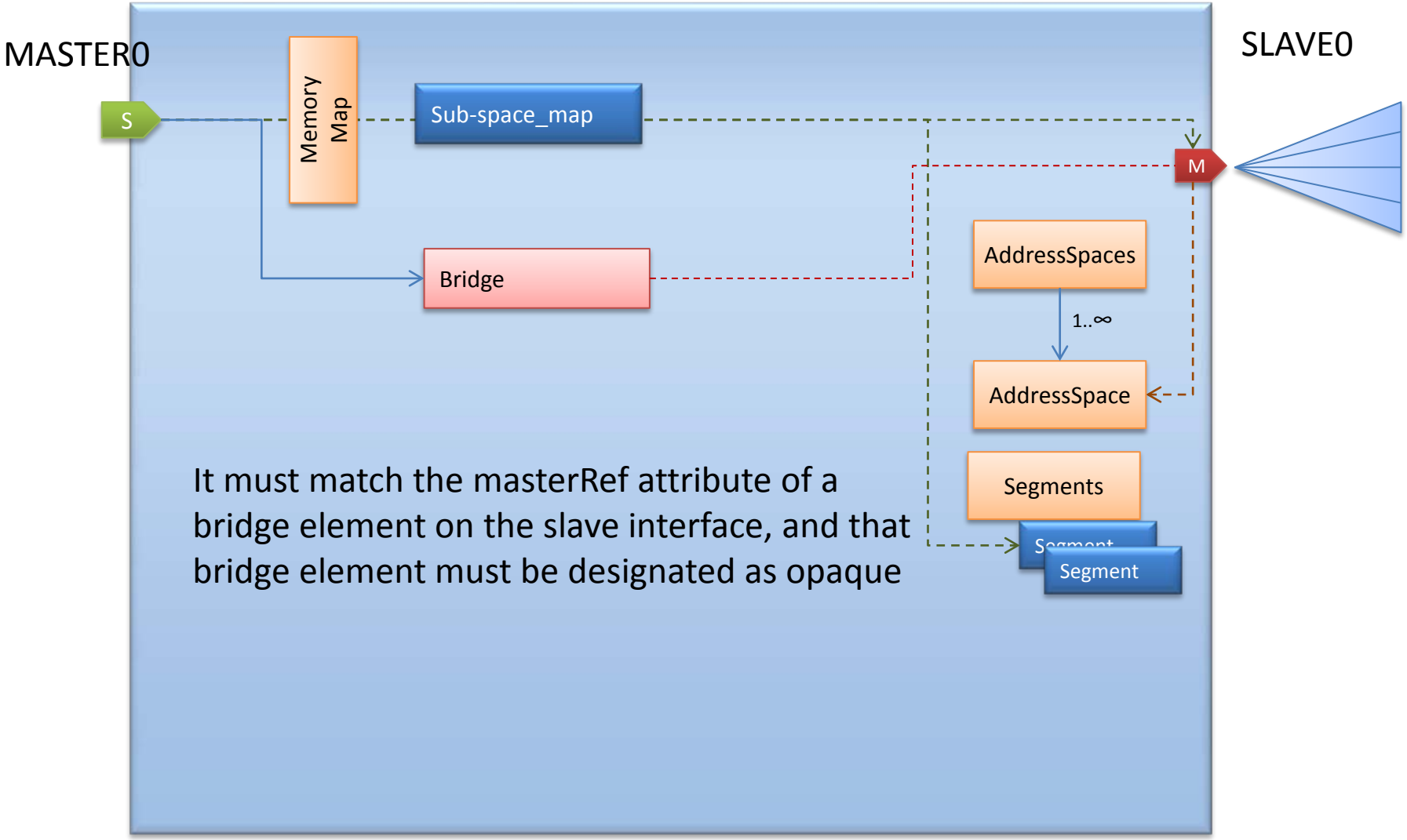
IP-XACT : Memory Mapping

(From Master)



IP-XACT : Memory Mapping

(From Master)



Standardizing HW/SW Interfaces Summary

- Very Extensive Register Model
- Full Hierarchical Memory map specification
- HW/SW Model is being improved [RWG]

Standardized checkers

- Static Schema checks :
  - Because a schema defines explicitly what element must be found in the xml, it is possible detect static errors : element found at wrong position or wrong type
- Semantic checks :
  - To complete the schema validation, a list of semantic rules aka SCRs have been defined by IP-XACT to handle complex situation and help catch other kind of errors.

Today's Agenda

- Welcome & introduction
- Users perspective of IP-XACT
  - NXP Semiconductor
  - ST Microelectronics
  - Texas Instruments
  - Xilinx
- An overview of IEEE P1685
  - What is IP-XACT
  - How IP-XACT is used to package components
  - How IP-XACT is used in design environments
  - What is an IP-XACT generator
- Standardizing HW/SW interfaces, registers and memory maps
- Other Accellera activity at DAC and an introduction to IP-XACT tool providers and where to fine more information about them

Accellera Activity at DAC 2011

- **DAC Workshop on Universal Verification Methodology (UVM) Verifying Blocks to IP to SOCs and Systems – Sorry, this was Sunday…**
- **Accellera Breakfast at DAC: *UVM User Experiences – Sorry this was this morning…***
- **Accellera IP-XACT Seminar – Thanks for attending!**
- **Birds-Of-A-Feather Meeting: Soft IP Tagging Standardization Kickoff – Today from 7:00 PM-8:30 PM San Diego Convention Center Room 31AB**

Birds-Of-A-Feather Meeting: Soft IP Tagging Standardization Kickoff

- ***Topic Area:*** General Interest **Tuesday, June 7, 2011**
  Time: 7:00 PM — 8:30 PM
- **Location:** 31AB
- **Summary:** Soft IP, particularly from third party vendors, must be tracked to satisfy contractual obligations such as royalty reporting and usage. Control of the third party IP source is lost once an IP is licensed, unlocked or otherwise made available in clear code.

  This meeting will discuss the problem space and the opportunities to contribute in the new Accellera technical committee being formed to address the issues.

  Organizer: Kathy Werner - *Freescale Semiconductor, Inc., Austin, TX*

For More on IP-XACT you can go to

- http://www.accellera.org/activities/ip-xact