

Virtual Platform: Modeling and Simulation

Franco Fummi



UNIVERSITÀ
di VERONA
Dipartimento
di **INFORMATICA**

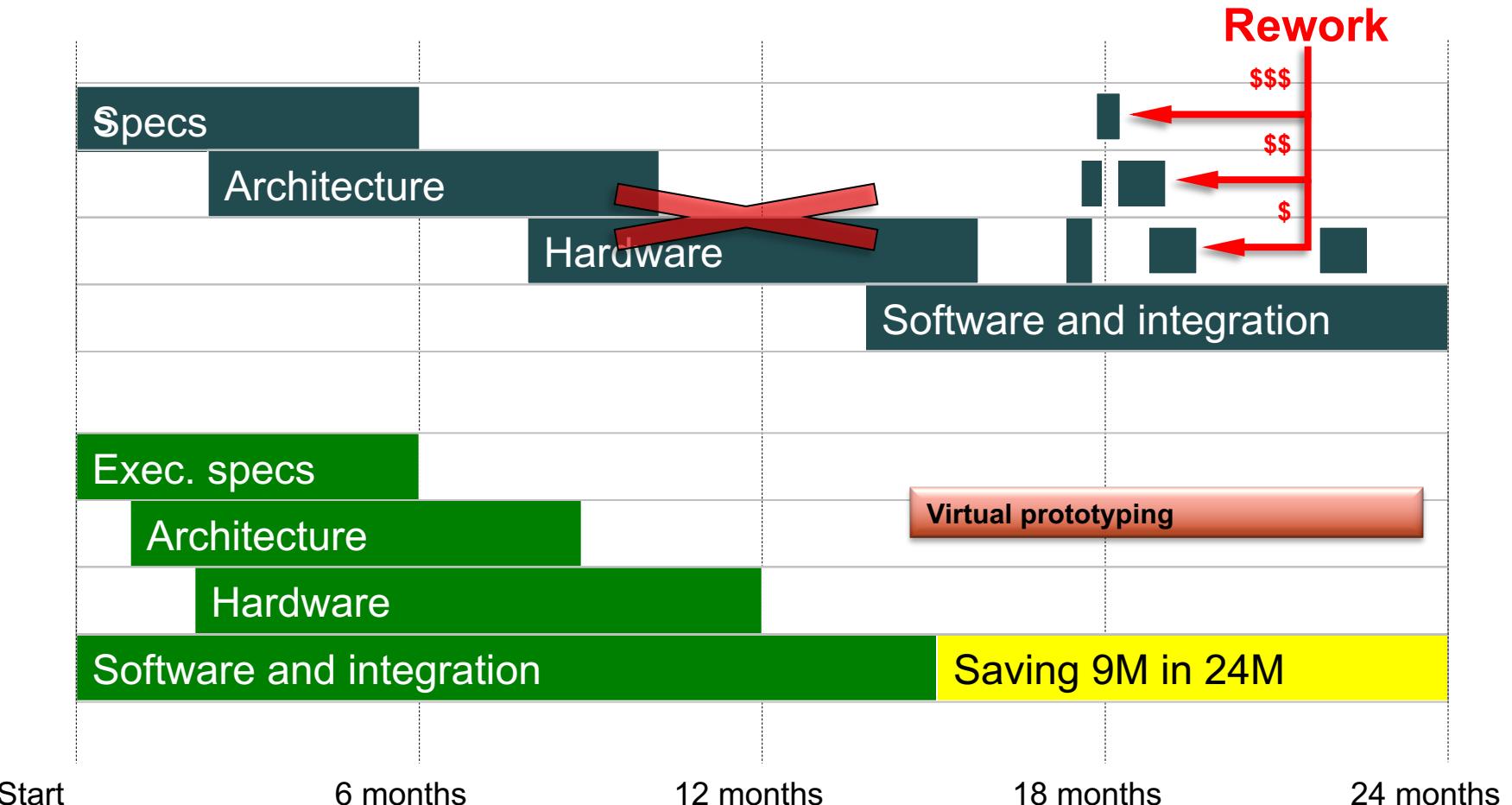
Version 1.0

Contents

- Virtual Platform (VP) motivations
- VP modeling languages:
 - IP-Xact
- VP simulation
 - homogeneous simulation
 - co-simulation
- A complete example

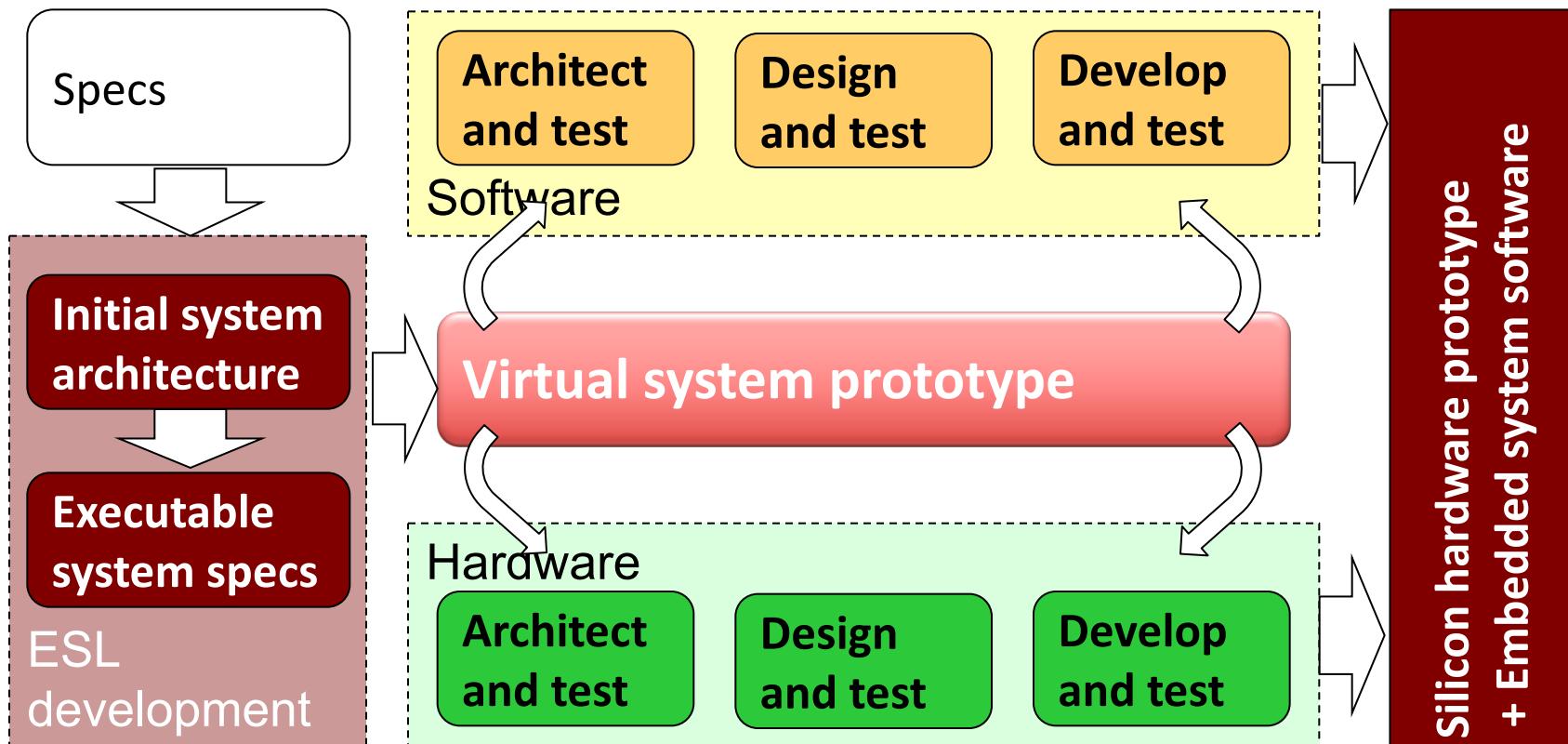
Conventional Design Process?

Source: G.Hellestrand, "Enhance Communications Platform Design with Virtual Systems Prototyping"



Virtual prototyping - Virtual platform (VP)

Source: G.Hellestrand, "Enhance Communications Platform Design with Virtual Systems Prototyping"



VP: not only functionality

- Power/thermal/aging estimation for a subsystem is typically done at a low-level design stage
- ...but crucial decisions based on these estimations need to be made at platform level
 - Virtual platform takes into account performance, power, thermal modeling etc.
 - Virtual platform forms a basis for architecture/IP/subsystem evaluation and continuity down to silicon level
 - Virtual platform becomes key to incorporating extra-functional type requirements into design technologies

VP MODELING LANGUAGES: IP-XACT

IP-XACT Standard

IP-XACT CAN Collect and connect components coming from different sources (and languages)

- IP-XACT has been created by the SPIRIT Consortium
- It has been approved on 2009 as IEEE 1685-2009 standard
 - it allows the integration of HW and SW IP-blocks to be independent from the IP vendor, the tools and the languages



IP-XACT and XML

- The IP-Xact standard is a XML model:
 - it allows the documentation of an IP in a uniform and automatizable way
 - there are .xls files to specify the XML conventions
- It includes interface information for each component in order to define the possible interconnection of components
- There is not information about component behaviors
- It is possible to have some “views” for each component, each one related to a different HDL

IP-XACT and XML

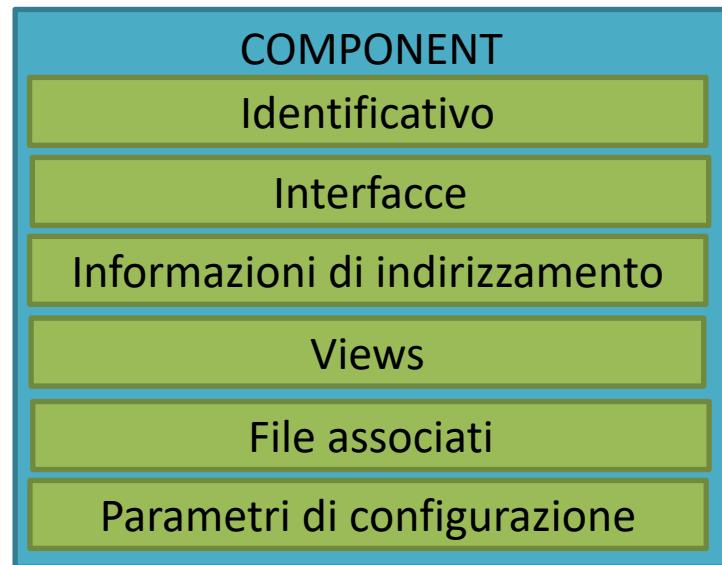
- The IP-XACT schema is the core part of an IP-XACT model
- Seven different objects are specified at top-level:
 - Component (it defines an IP or an interconnection)
 - Bus definition (it allows to define all properties of a bus)
Very important
 - Abstraction definition (other attributes related to bus)
 - Design (it defines the interconnection between components and their configuration)
 - Abstractor (an adapter that allows to interconnect components at different abstraction levels)
Very close to SystemC/TLM abstractor
 - Generator chain (a list of generators)
 - Design configuration (parameters related to design of each component)

IP-XACT XML: component I

- Un IP-XACT component è il contenitore principale per i metadati degli oggetti
- Può essere static o configurable: nel primo caso l'oggetto non è modificabile, nel secondo contiene elementi configurabili (o parametrizzati) che possono configurare non solo la descrizione IP-XACT ma anche i sottostanti modelli RTL o TLM
- Può essere un oggetto gerarchico o terminale: i componenti terminali non contengono altri componenti IP-XACT, mentre quelli gerarchici contengono altri IP-XACT sub-componenti

IP-XACT XML: component II

- Un IP-XACT component può contenere:
 - Identificativo:
 - Una quaterna VLVN: vendor, library, name, version
 - Identifica univocamente l'oggetto IP-XACT e può essere usato per farne riferimento
 - Interfacce
 - Tramite bus (busInterfaces) o channel (channels) ossia interconnessioni tra le interfacce bus
 - Informazioni di indirizzamento
 - Spazi di indirizzamento (addressSpaces) e mappature di memoria (memoryMaps)
 - Views
 - Vengono inserite nel model e specificano le varie viste (views) che implementano l'interfaccia (ports) dell'oggetto descritto
 - File associati
 - Riferimenti a file (*fileSets*) che sono relativi all'oggetto descritto, solitamente quelli che contengono le implementazioni delle view
 - Parametri di configurazione
 - Parametri (parameters) che permettono di modificare la configuazione dell'oggetto



IP-XACT XML: component III

- versionedIdentifier
 - VLNV, mandatory
- busInterfaces
- channels
- remapStates
- addressSpaces
- memoryMaps
- model
- componentGenerators
- choices
- fileSets
- whiteboxElements
- cpus
- otherClockDrivers
- description
- parameters
- vendorExtensions

IP-XACT XML: versionedIdentifier

- Contiene le informazioni che permettono di identificare univocamente l'oggetto descritto e di farne riferimento da altri file IP-XACT
- I campi, tutti obbligatori, di questo gruppo sono:
 - vendor identifica il proprietario della descrizione
 - Viene rappresentato tramite il dominio internet della compagnia in ordine da sinistra a destra
 - library identifica la libreria di cui la descrizione fa parte
 - name identifica il nome della descrizione all'interno della libreria
 - version identifica la versione della descrizione per poter distinguere oggetti con lo stesso nome, ma diversi
 - Ha la forma di una stringa alfanumerica, suddivisa in sottostringhe con delimitatori non alfanumerici (ad esempio :,,,-,_)

IP-XACT XML: model

- Un elemento di tipo model descrive le view, le porte e parametri relativi al modello
- Questo elemento può contenere:
 - views contiene una lista delle view (elementi di tipo [view](#)) per questo oggetto
 - Un oggetto può avere diverse view: una RTL può descrivere il sorgente del modulo o dell'interfaccia hardware; una software può descrivere il sorgente del device driver scritto in linguaggio C; una di documentazione può rappresentare la specifica scritta dell'IP
 - ports contiene una lista di porte (elementi di tipo [port](#)) per questo oggetto
 - Un oggetto può avere un solo insieme di porte che deve essere valido per tutte le view
 - modelParameters contiene una lista di parametri necessari per configurare l'implementazione del modello specificata in una view

IP-XACT XML: view I

- Questo elemento è presente in un numero arbitrario all'interno dell'elemento views
- Ciascun elemento di tipo view specifica un livello di rappresentazione di un component
- Contiene i seguenti elementi:
 - nameGroupNMTOKEN (obbligatorio) identifica la view e il campo name deve essere unico nell'elemento views
 - envIdentifier (obbligatorio) specifica informazioni sull'ambiente del tool con cui la view è stata sviluppata, può essere più di uno
 - hierarchyRef (obbligatorio) se specificato indica che questa view si riferisce a un oggetto di tipo design gerarchico che conterrà i dettagli; in alternativa vengono specificati i campi successivi che si riferiscono a un [fileSet](#) specificato nel documento

IP-XACT XML: view II

- language (opzionale) indica l’HDL usato per la specifica view
- modelName (opzionale) è l’identificatore del modello specifico del linguaggio
- fileSetRef (opzionale) è una lista arbitrariamente lunga di riferimenti al campo name di un fileSet contenuto nel documento
- Lo standard specifica anche altri elementi opzionali che compongono la view, ma non sono utilizzati nell’implementazione del tool

IP-XACT XML: view III

- Seguono le specifiche di quali informazioni vengono effettivamente inserite nell'elemento view
- Il gruppo nameGroupNMToken della view ha come campi name e displayName
 - name (obbligatorio): verrà inserita una stringa diversa a seconda del linguaggio con cui è scritta l'implementazione della view; VHDLView, VerilogView o SystemCView
 - displayName (opzionale): anche in questo caso una stringa dipendente dal linguaggio della view che la descrive (ad esempio “This is the VHDL view of entity”)

IP-XACT XML: view IV

- L'elemento envIdentifier ha la forma Language:Tool:VendorSpecific dove:
 - Language (opzionale) indica il linguaggio con cui il tool deve essere compatibile
 - Tool (opzionale): indica il tool utilizzato per lo sviluppo dell'implementazione della view o un tool compatibile con l'implementazione
 - Il nome del tool è una stringa identificativa approvata dai proprietari
 - VendorSpecific (opzionale): indica ulteriori informazioni per la compatibilità dell'implementazione
- Il valore di questo campo sarà standard in base al linguaggio dell'implementazione, con un elenco dei più diffusi tool di sviluppo per quel linguaggio

IP-XACT XML: view V

- L'elemento language conterrà il nome del linguaggio usato per l'implementazione
- L'elemento modelName ha una forma specifica in base al linguaggio dell'implementazione:
 - VHDL: la coppia entity(architecture)
 - Verilog (e SystemVerilog): il nome del modulo
 - SystemC: il nome della classe sc_module
- L'elemento fileSetRef contiene un solo campo, localName che contiene il campo name di un [fileSet](#) specificato nel documento; questo fileSet sarà quello che contiene il riferimento al file che implementa la view

IP-XACT XML: port

- Questo elemento è presente in un numero arbitrario all'interno dell'elemento ports
- Ciascun elemento descrive una singola porta esterna del component
- Contiene i seguenti campi:
 - nameGroupPort (obbligatorio) il nome della porta
 - wire (obbligatorio) rappresenta una porta fisica, risolta in termini di bit
 - transactional (obbligatorio) rappresenta un servizio, fornito o utilizzato, per descrizioni TLM
 - Alternativo al campo wire
- Lo standard specifica anche altri campi opzionali, ma non sono utilizzati nell'implementazione del tool

IP-XACT XML: port/wire I

- L'elemento port/wire descrive le proprietà di porte fisiche che trasportano tipi scalari o vettori di scalari
- Lo standard stabilisce che i tipi scalari includono solamente valori binari o vettori di binari che corrispondono nell'implementazione hardware ad una singola connessione
- I campi definiti sono:
 - direction (obbligatorio) specifica la direzione di questa porta, può essere in, out, inout o phantom
 - phantom indica una porta presente nel file IP-XACT, ma non nell'implementazione

IP-XACT XML: port/wire II

- vector (opzionale) se presente indica che la porta è un vettore di bit, altrimenti è uno scalare
 - Possiede due campi, left e right che indicano rispettivamente l'indice del bit più a sinistra e più a destra
 - left può essere più grande di right e può essere il MSB o LSB (right viceversa)
- wireTypeDefs (opzionale) composto da elementi wireTypeDef, descrive il tipo della porta come definito dall'implementazione
 - Può avere più di un elemento componente, perché il tipo può essere diverso tra le varie view
 - Ogni elemento può riferirsi a più di una view
 - I campi di wireTypeDef sono:
 - typeName (obbligatorio) indica il nome del tipo della porta; ha come attributo booleano constrained (opzionale) che indica se il tipo ha un numero fissato o variabile di bit

IP-XACT XML: port/wire III

- typeDefinition (opzionale) fornisce un riferimento a dove il tipo è stato effettivamente definito, in un modo specifico del linguaggio (ad esempio per VHDL il contenuto della direttiva use, per SystemC il nome del file della direttiva include)
- viewNameRef (obbligatorio) indica tramite il campo name la o le view in cui questo tipo viene usato
- driver (opzionale) definisce il driver per questa porta
 - Può essere solo uno di questi tre tipi:
 - defaultValue definisce un valore di default che si applica se la porta non è connessa
 - clockDriver definisce una forma d'onda periodica per la porta
 - singleShotDriver definisce una forma d'onda non periodica per la porta

IP-XACT XML: port & port/wire I

- Seguono le specifiche di quali informazioni vengono effettivamente inserite negli elementi port e wire
- Nel gruppo nameGroupPort di port verrà utilizzato solo il campo name e conterrà il nome della porta
- Verranno considerate solo le porte di tipo wire e quindi le sole implementazioni RTL
- Nell'elemento wire si considerano:
 - direction che corrisponderà alla direzione della porta come specificato nell'implementazione (quindi saranno escluse le porte phantom)
 - vector che sarà presente in caso di porte multibit e anche nel caso in cui il tipo di dato sia composto da più di un bit

IP-XACT XML: port & port/wire II

- wireTypeDef sarà presente in ogni caso, anche se vengono utilizzati i tipi di default definiti dallo standard
 - Il campo typeName conterrà il nome del tipo e l'attributo constrained verrà sempre specificato e varrà true per i tipi vettoriali
 - Il campo typeDefinition sarà presente solo per i tipi di cui è noto a priori il file di dichiarazione
 - Il campo viewNameRef conterrà il nome della view in cui il tipo viene utilizzato
- defaultValue è l'unico driver che verrà utilizzato nel caso in cui l'implementazione lo definisca

IP-XACT XML: fileSet I

- Questo elemento è presente in un numero arbitrario all'interno dell'elemento fileSets
- Contiene una lista di file e directory associati al componente, logicamente legati tra loro
- I campi definiti sono:
 - nameGroup (obbligatorio) contiene il campo name che identifica il fileSet ed è unico nel fileSets contenente
 - name (opzionale) descrive con una singola parola la funzione o lo scopo del fileSet

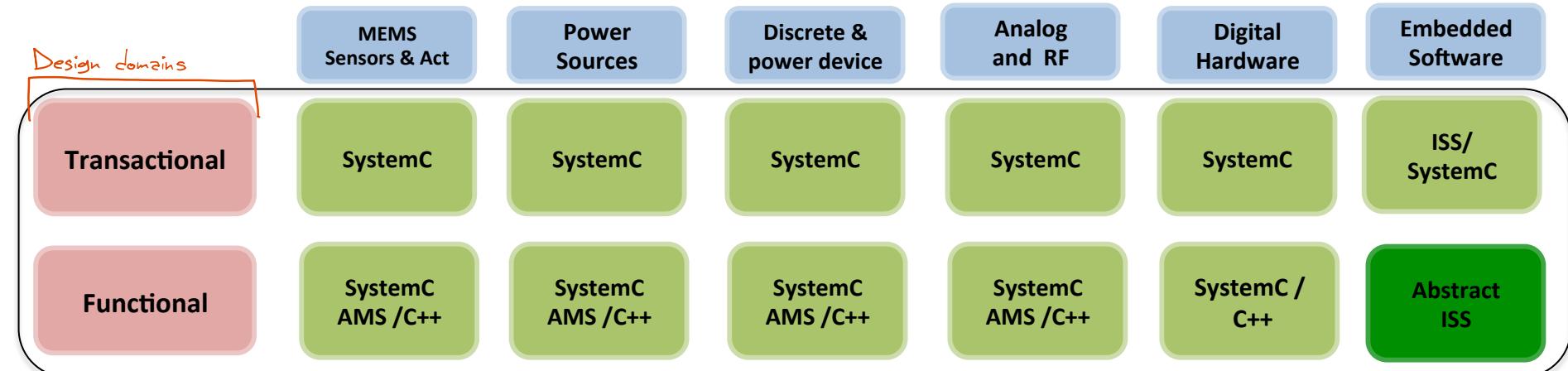
IP-XACT XML: fileSet II

- **file** (opzionale) una lista di lunghezza arbitraria di file o directory associati al **fileSet**
 - Si struttura in sottocampi che ne permettono l'identificazione nel file system
 - **name** (obbligatorio): il path al file o alla directory, assoluto o relativo (rispetto al documento XML contenente) e può contenere variabili d'ambiente del sistema host nella forma \${ENV_VAR}
 - **fileType** (obbligatorio) è un gruppo e può essere di due tipi alternativi:
 - fileType se il tipo è noto allo standard e ha un valore enumerativo definito dallo standard (ad esempio per VHDL vhdlSource)
 - userFileType per tutti i tipi che non ricadono in quelli noti allo standard
 - può esserci più di una volta
 - **logicalName** (opzionale) nome logico per il file o la directory
- Lo standard specifica anche altri campi opzionali, ma non sono utilizzati nell'implementazione del tool

VP SIMULATION

Simulation or Co-simulation?

Simulation



Co-simulation

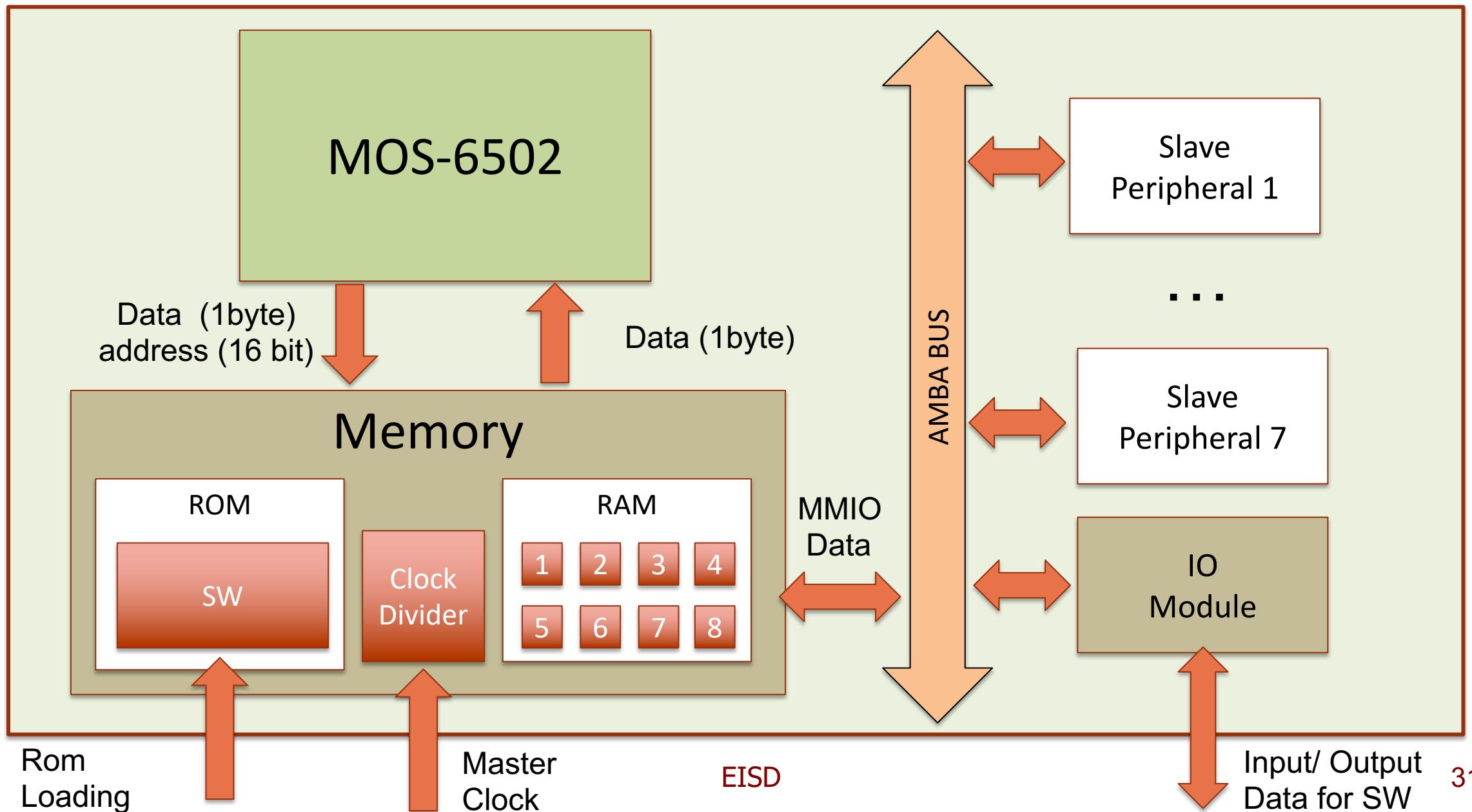
Structural	ADS Cir. Sim. AMS HDL Matlab MEMS+	Matlab/ Simulink AMS HDL VerilogA	ADS circuit simulator	AMS HDL Matlab Suite ADS circuit simulator	RTL HDL	Cycle Accurate ISS
Device	AMS HDL Matlab MEMS+	FEM models Spice	EMPro Momentum Spectre	EMPro Momentum Spectre	AMS HDL	
Physical	FEM models Matlab MEMS+	FEM models Spice	EMPro Momentum Spectre	EMPro Momentum Spectre	AMS HDL	



Default entry level for each domain

A COMPLETE EXAMPLE

COM6502



COM6502

- The platform includes:
 - CPU MOS 6502 (1975)
 - Vic 20, Commodore 64, Nintendo Entertainment System (NES), Apple II, Atari 800, Terminator
 - 16 bit addressing (16KB ROM, 16KB RAM),
 - 8 bit data width
 - Memory
 - ROM in one single bank
 - RAM splitted in 8 different blocks to enable multi read/write operations
 - Clock divider
 - manage MMIO operation between Peripherals, Memory
 - Manage multiple write on same cell between CPU and MMIO Interface
 - BUS ARM APB (Advanced Peripheral Bus)
 - Supports up to 8 peripherals
 - IO Module
 - Used to request or send data out from the platform

COM6502

- SW can be written in C Language
 - CC65 Cross Compiler for MOS 6502 CPU
 - <https://www.cc65.org/>
 - No support for float/double types
 - IEEE 754 didn't exists when MOS 6502 was developed!
 - Int datatype is only 1 byte
- ESD Team Developed a MMIO library in C
 - Allows to easily write simple drivers for Custom Slave Peripherals

