ARM RealView ESL APIs
5th NASCUG Meeting
July, 24th 2006

Nizar ROMDHANE
ESL Technical Marketing Manager, DSMK
July 2006

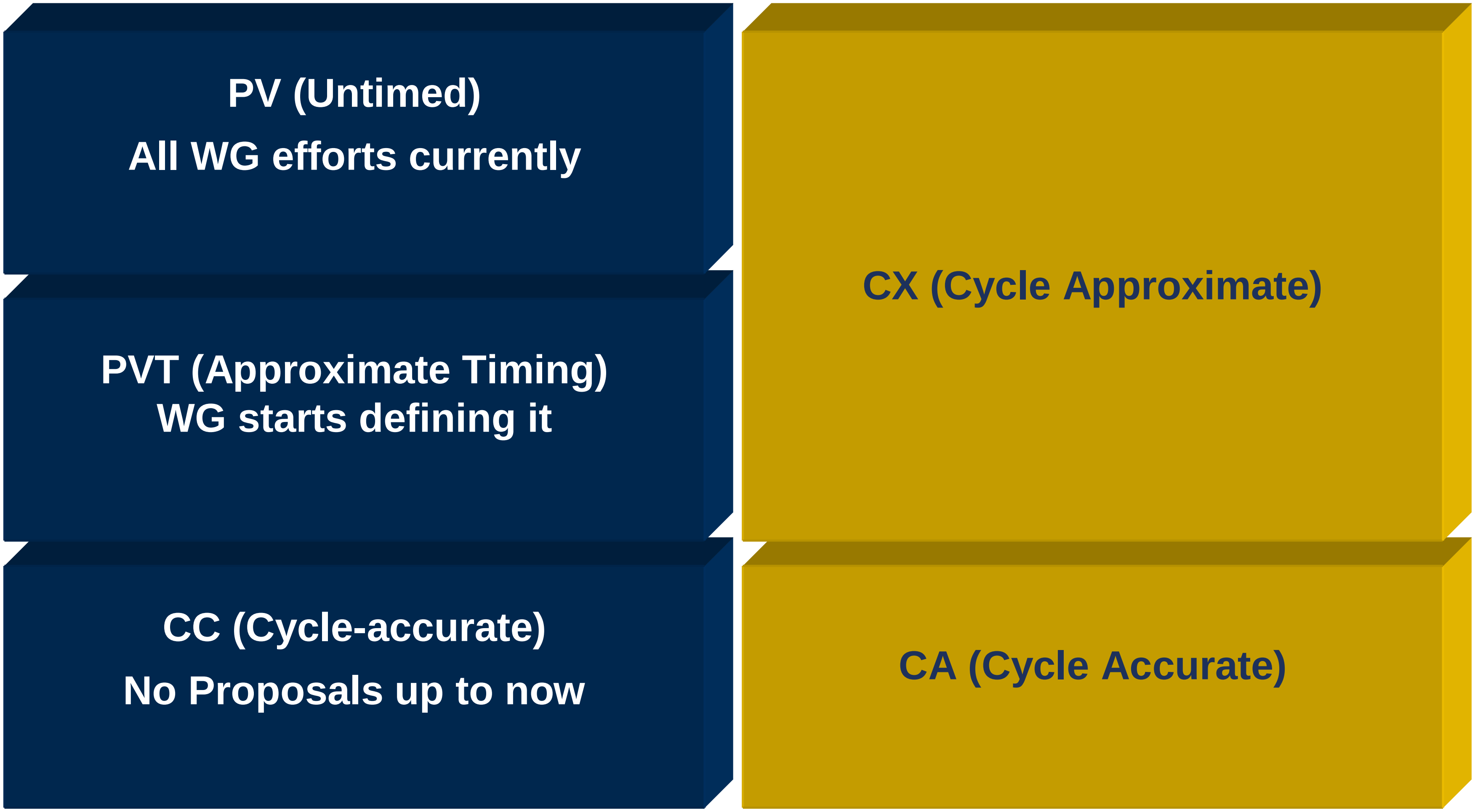
Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

Transaction Level Modelling

- Nowadays we can classify 2 main Modelling Styles
  - **Pin Level Modelling**
    - Pin accurate with RTL model at the Bus interface
    - Cycle accurate with RTL model
    - Internal behaviour and timing is abstracted to the clock cycle boundary
    - Introduced by SystemC v1.0 and uses heavily sc_in, sc_out, sc_inout and sc_signal
  - **Transaction Level Modelling** (TLM)
    - Bus interfaces abstracted to ports carrying transactions
    - Communication uses function calls
    - Timing can be:
      - Cycle accurate (CC)
      - Timing Approximate (PVT)
      - Untimed (PV)
    - Introduced by SystemC v2.0 and is based on interfaces and channels

Abstraction Layers: OSCI TLM



OSCI TLM

PV (Untimed)

All WG efforts currently

PVT (Approximate Timing)
WG starts defining it

CC (Cycle-accurate)

No Proposals up to now

ARM

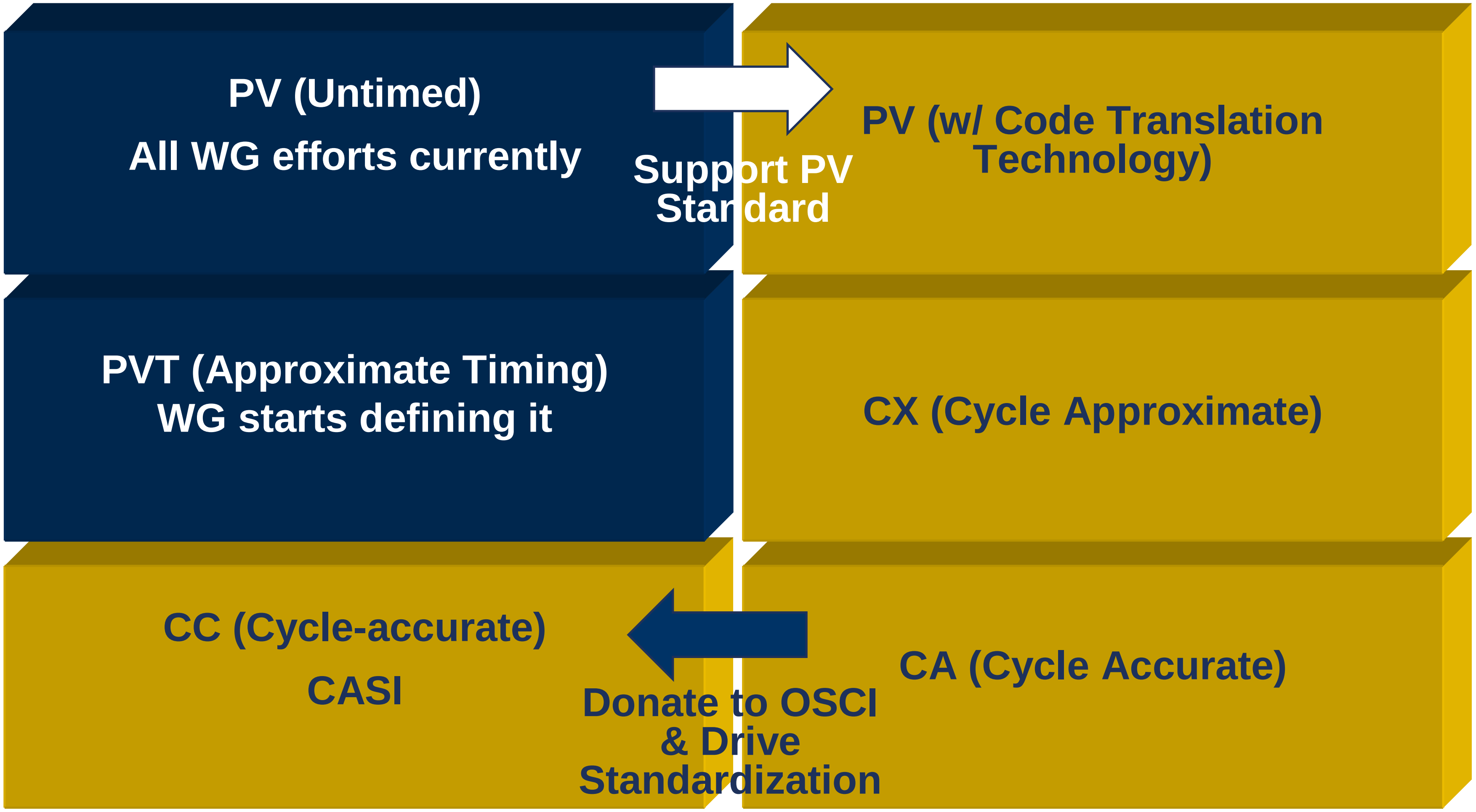
CX (Cycle Approximate)

CA (Cycle Accurate)

ARM is Adopting and Donating

OSCI TLM

ARM



PV (Untimed)

All WG efforts currently

Support PV Standard

PV (w/ Code Translation Technology)

PVT (Approximate Timing)
WG starts defining it

CX (Cycle Approximate)

CC (Cycle-accurate)

CASI

Donate to OSCI & Drive Standardization

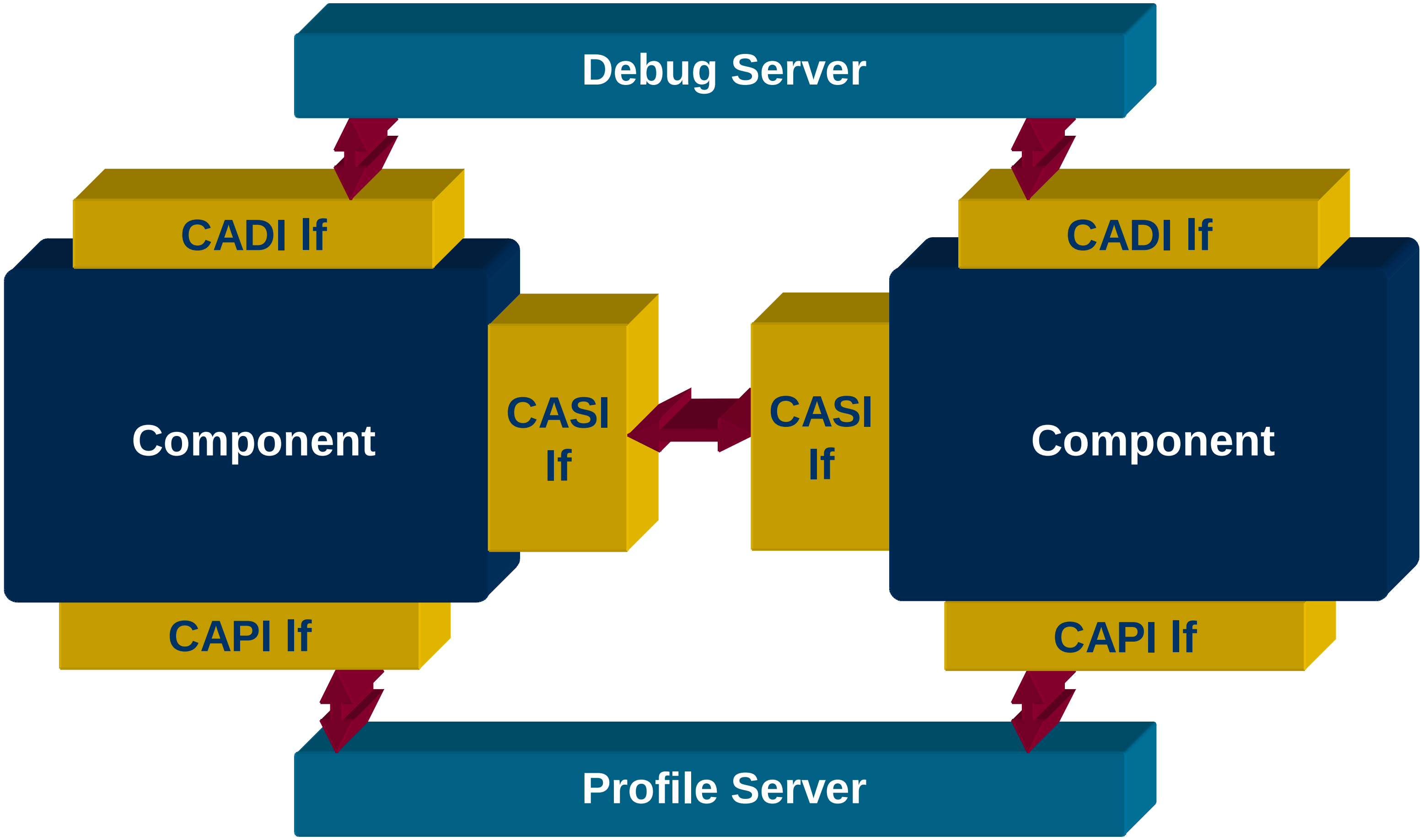
CA (Cycle Accurate)

ARM RealView ESL APIs

- ARM RealView ESL APIs are a Bundle of:
  - Cycle-Accurate Interfaces
    - CASI: Cycle Accurate Simulation Interface
    - CADI: Cycle Accurate Debug Interface
    - CAPI: Cycle Accurate Profiling Interface
  - AMBA (AXI, AHB and APB) Implementation on top of CASI
- APIs are provided openly
  - Web-hosted under an AMBA-like click-through agreement
  - http://www.arm.com/products/DevTools/RealViewESLAPIs.html
- Version 1.0 released in Oct 2005 with endorsements from major EDA vendors
- **Version 1.1 has just been released at DAC06**
- Deliverables:
  - Specifications: Including APB/AHB/AXI Transaction Level Modelling
  - Header files
  - Examples:
    - Source code examples provided for OSCI Open Source Proof of Concept Simulator of:
      - AXI simple master, slave and connection
      - AHB simple master, slave and connection

RealView ESL APIs Overview

- CASI: Cycle-based SystemC simulation interface
- CADI: C++ debug interface
- CAPI: C++ profiling interface



Presentation Structure

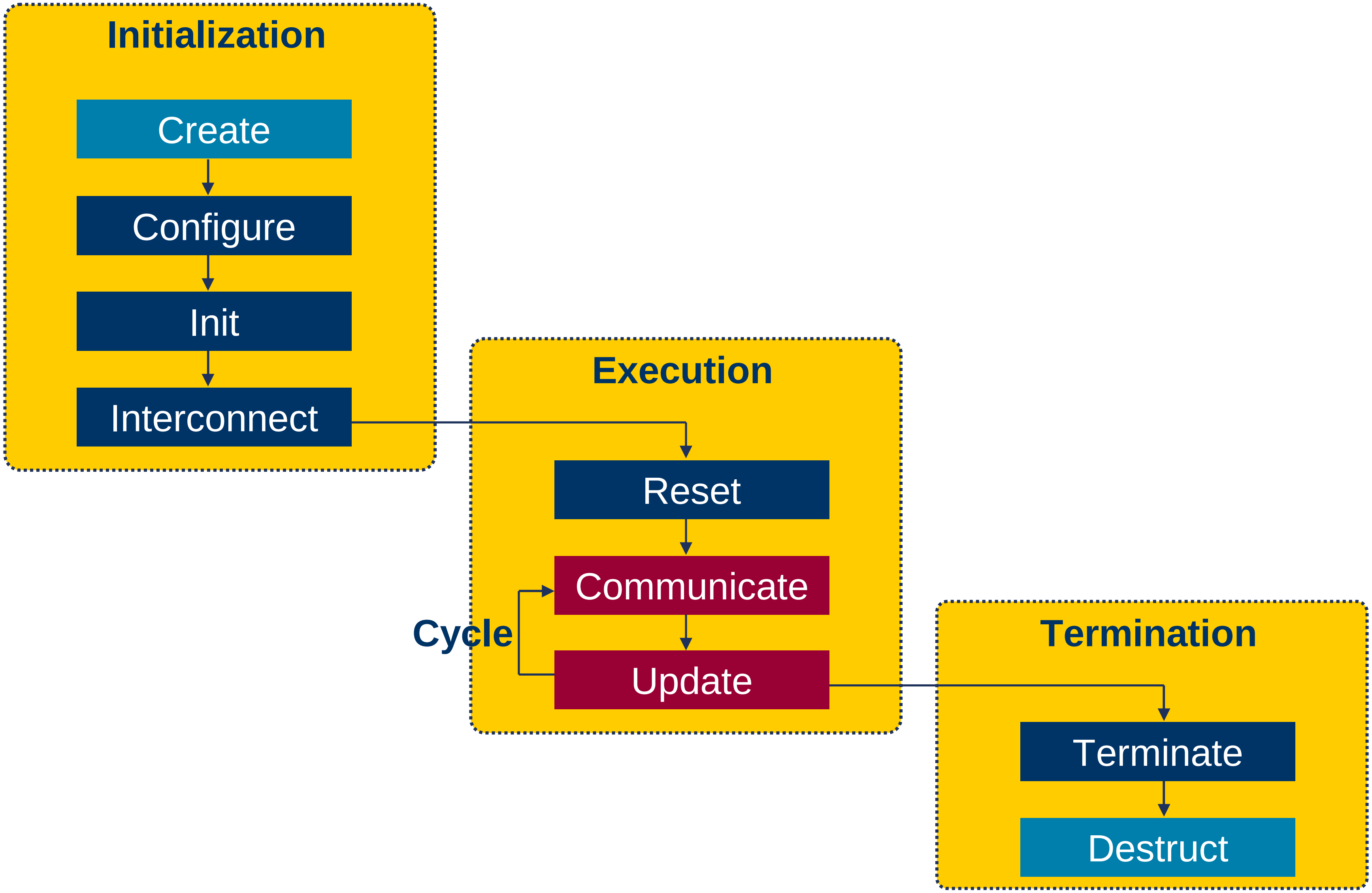
- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

Definition of ‘Cycle’

- Timing is represented in the form of cycles.

- The RealView ESL APIs do not define the meaning of a cycle to be a “clock cycle”.
  - Rather this decision is left to the individual model.
  - The meaning of a cycle is decided by the model abstraction level. E.g.:
    - Instruction-accurate core: 1 cycle = 1 instruction.
    - Cycle-accurate core: 1 cycle = 1 clock cycle.
    - Functional memory component: 1 cycle = 1 read/write operation.

Cycle-based Scheduling: Sequencing



Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

Cycle-Accurate Simulation Interface (CASI)

- Supports both event-driven and cycle-based scheduling
- Non-blocking
- Bi-directional
- Interfaces:
  - Transaction-level:
    - read/write, readReq/writeReq
    - driveTransaction
  - Signal-level:
    - driveSignal

CASI Terms:

OSCI	CASI
Port (sc_port)	Master port
Channel/sc_export	Slave port

CASI TLM Access Methods

- CASI Supports 3 access methods:

1. Synchronous accesses:
   - read/write
   - Every cycle read/write is called to check if data ready
2. Asynchronous accesses:
   - readReq/writeReq, readAck/writeAck
   - Callback function sent to slave. Slave calls it when ready.
3. Asynchronous shared memory accesses:
   - driveTransaction/notifyEvent
   - Shared data structure sent on first call. Slave updates this data structure directly.
   - repeated calls as well as notification callbacks allow interim update triggering

CASI TLM Main Constructs

- CASI Interfaces
  - CASI Transaction Interface

    ```
    class casi_transaction_if : public sc_interface {…};
    ```

  - CASI Transaction Callback Interface

    ```
    class casi_transaction_callback_if {…}
    ```

  - CASI Notify Handler Interface

    ```
    Class CASINotifyHandlerIF {…};
    ```

  - CASI Signal Interface

    ```
    class casi_signal_if : public sc_interface {…};
    ```

- CASI Transaction Structures
  - CASI Transaction Properties

    ```
    struct CASITransactionProperties {…};
    ```

  - CASI Transaction Info

    ```
    struct CASITransactionInfo {…};
    ```

- CASI Module

  ```
  class casi_module: public sc_module {…};
  ```

Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

CASI support for abstraction / protocols

- CASI supports any bus protocol
  - CASI interface has been proven to support AMBA protocols, as well as other industry standard bus protocols
  - Bus-generic transactional interface can be easily personalized for support of a specific bus protocol
  - ARM RealView ESL APIs are provided with support for all AMBA protocols: AXI, AHB, APB
- CASI supports any model abstraction
  - Cycle-based execution can be clock-or-instruction cycle based
  - One interface for all ESL model abstractions
  - Enables a simple refinement process for ESL models

Bus protocol mapping on CASI

- The following will illustrate the official AXI protocol mapping to CASI as an example.
- To map a bus protocol onto CASI
  - Create a mapping of the protocol fields to the CASI access method parameters
  - E.g.: AXI using CASI asynchronous shared memory access mode
    - **AXI Master port:**
      - sc_port specialized w/ casi_transaction_if

        ```
        class AXI_TM : public sc_port<casi_transaction_if, 0> {… };
        ```

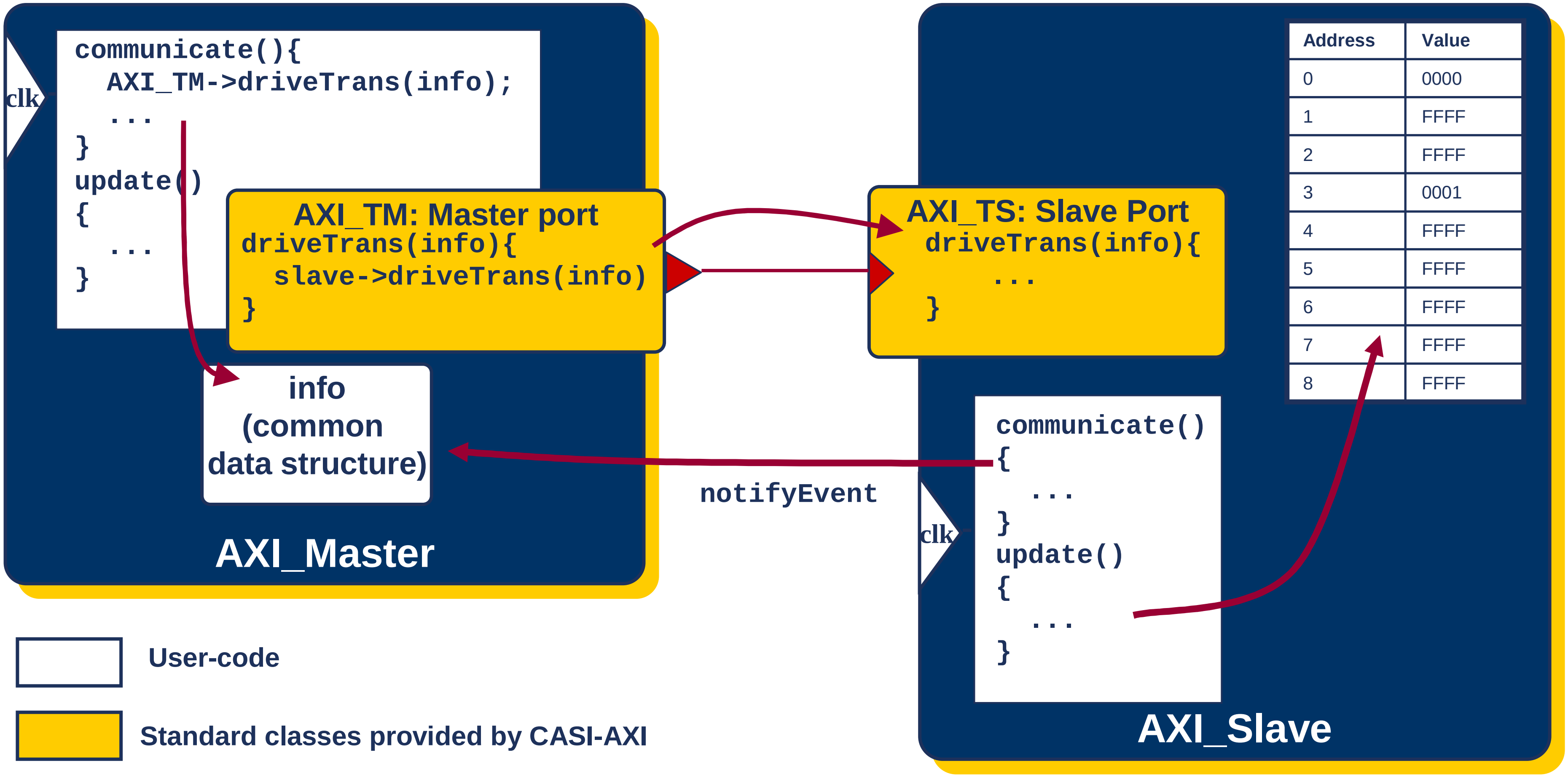
      - Implements driveTransaction method
    - **AXI Slave port:**
      - casi_transaction_if or sc_export specialized w/ casi_transaction_if

        ```
        class AXI_TS : public casi_transaction_if {…};
        ```

      - Implements driveTransaction, notifyEvent
    - **AXITransaction:**
      - inherited from CASITransactionInfo

        ```
        class AXITransaction : public CASITransactionInfo {…};
        ```

Example AXI Protocol in CASI



```
communicate(){
  AXI_TM->driveTrans(info);
  ...
}
update()
{
  ...
}
```

AXI_TM: Master port

```
driveTrans(info){
  slave->driveTrans(info)
}
```

AXI_TS: Slave Port

```
 driveTrans(info){
     ...
 }
```

```
communicate()
{
  ...
}
update()
{
  ...
}
```

Address	Value
0	0000
1	FFFF
2	FFFF
3	0001
4	FFFF
5	FFFF
6	FFFF
7	FFFF
8	FFFF

□ User-code

🟨 Standard classes provided by CASI-AXI

Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

CADI Overview

- The Cycle Accurate Debug Interface (CADI) is intended for use in conjunction with the Cycle-Accurate Simulation Interface (CASI) to allow inspection and modification of the internal state of SystemC models through an externally attached debugger.
- However, these hooks would be of little value if the simulation cannot be controlled from the debugger
- The models run under the supervision of a simulation host which is in charge of managing the SystemC simulation. For cycle-based scheduling, calling the communicate/update methods in each cycle.
- CADI is intended to serve as a bridge between high level simulation commands that a debugger may issue and low level simulation commands that a simulation engine may understand.

Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

CAPI interface Overview

- The CAPI interface supports a generic implementation of profiling, allowing collecting of different types of data, organized around streams and channels of information.
- **Profiling streams and channels**
  - A CAPI interface implemented by a model will have one or more profiling streams.
  - Each stream will contain one “thread” of information.
  - For each profiling stream CAPI maintains two distinct data structures: the stream metadata and the stream content.
  - The stream metadata describes what profiling data is collected, plus hints to present the data for final human consumption.
  - The stream content is the actual data collected during simulation. Each profiling stream collects data for one or more profiling channels. Each channel represents one piece of information to be collected in the stream.

Presentation Structure

- Introduction
- Cycle-based Scheduling
- CASI: simulation interface basics - technical overview
- Protocols: encapsulating a bus interface in CASI
- CADI: debugging interface basics - technical overview
- CAPI: profiling interface basics - technical overview
- ARM RealView ESL APIs timelines

Releases

- V1.0: already released in October 05
  - CA interfaces: CASI / CADI / CAPI
  - Media Alert with Major EDA vendors endorsements
- V1.1: Scheduled for July 06
  - CASI:
    - Memory Map Interface, Dynamic Config
  - AMBA AXI-CASI:
    - Support of 128-bit data
- V1.1.1: Scheduled for Oct 06
  - Component Wizard
- V2.0: Scheduled for March 07
  - PVSI: PV Simulation Interface (based on OSCI TLM v2.0)
    - First OSCI TLM PV to CASI adaptation technology
  - AMBA AXI-CASI:
    - New Ports classes to support sequential and combinatorial behaviours targeting AMBA4

ARM®

Thank You