# SPI Engine

## A framework for creating SPI controllers

Lars-Peter Clausen
<lars-peter clausen@analog.com>

## Introduction

- ADI makes SPI compatible peripherals
  - ADC, DAC, Gyroscope, IMU, …
- SPI interface is used for
  - Configuring the peripheral
  - Reading/writing sample data from/to the peripheral
- A data ready signal indicates completion of a conversion

## Software based flow

- Configure peripheral
  - Sampling rate, active channels, …
- Enable conversion
- Capture data (repeat)
  - Wait for data ready IRQ
  - Read result
- Disable conversion

## Software based flow - Challenges

- Configure peripheral
  - Sampling rate, active channels, …
- Enable conversion
- Capture data (repeat)
  - Wait for data ready IRQ
  - Read result
    - Start SPI transfer
    - Wait for SPI transfer to finish
- Disable conversion

← Critical section

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

## Software based flow - Challenges

- Peripherals often have no FIFO
- Samples rates can go up to several MSPS
- Low latency access is required
  - Polling has huge CPU overhead (no other processes can run)
  - Interrupt driven approach has non deterministic latency on a general purpose OS
  - Samples are lost
- At high samplerates one interrupt per sample becomes unsustainable
  - CPU is saturated

## Software based flow - Challenges

- Peripheral is SPI compatible, but ...
- For datasheet performance very precise timings are necessary

## Software based flow - Challenges

- Moving sample data into a processing pipeline ...
  - Requires additional memory copies
  - Introduces additional latency

Use a FPGA!

## Solution – Use FPGA!

- Use FPGA to SPI bus interfacing
- Initial solution: Custom IP core for each project
  - Works fine if number of projects is small
  - Useful if no pattern has emerged yet

## One Core per Project - Issues

- Very application specific and monolithic
  - Tightly coupled application specific logic and interface logic
  - Only specific feature subset of the peripheral are supported
  - Makes it hard for the customer to adapt and re-use it
- Assumes that there is exactly one SPI peripheral connected
- Custom and incompatible interfaces
  - Needs custom software driver support

## One Core per Project - Issues

- Copy & Paste
  - Large maintenance overhead
  - Leads to bit-rot (lots of outdated projects)
- Very little documentation
  - Makes it hard for the customer to use

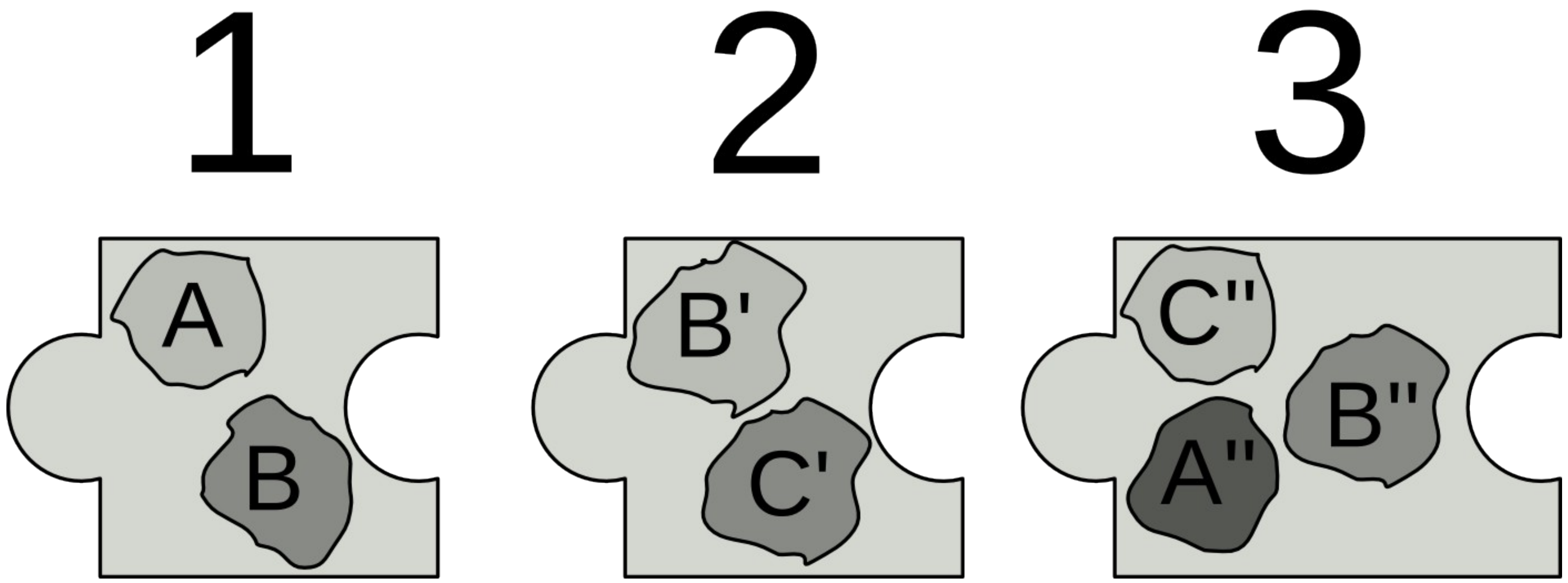
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

## One Core per Project - Issues

- Summery
  - Low flexibility and re-usability
  - Works fine for one or two projects but does not scale
- Focus should be on providing solution building blocks to the customer
  - Demos only example of how to use the building blocks

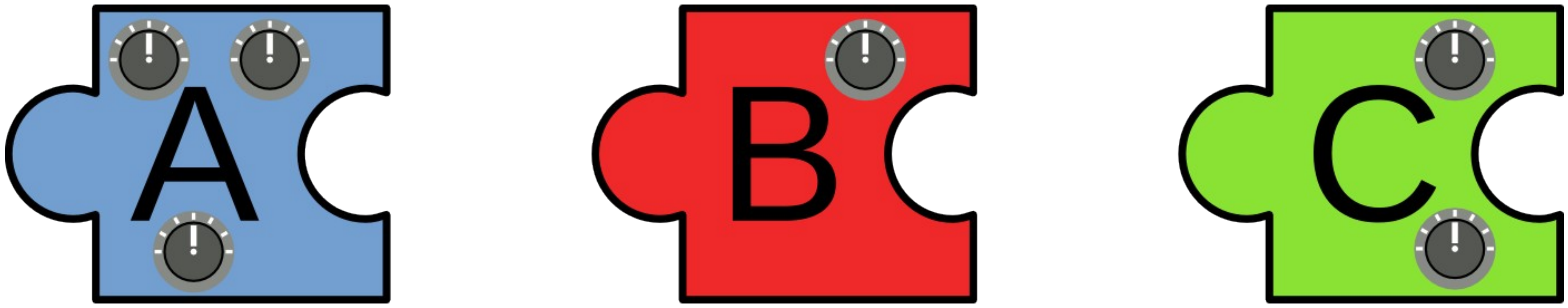
## Modularization

## Design Pattern - Modularization



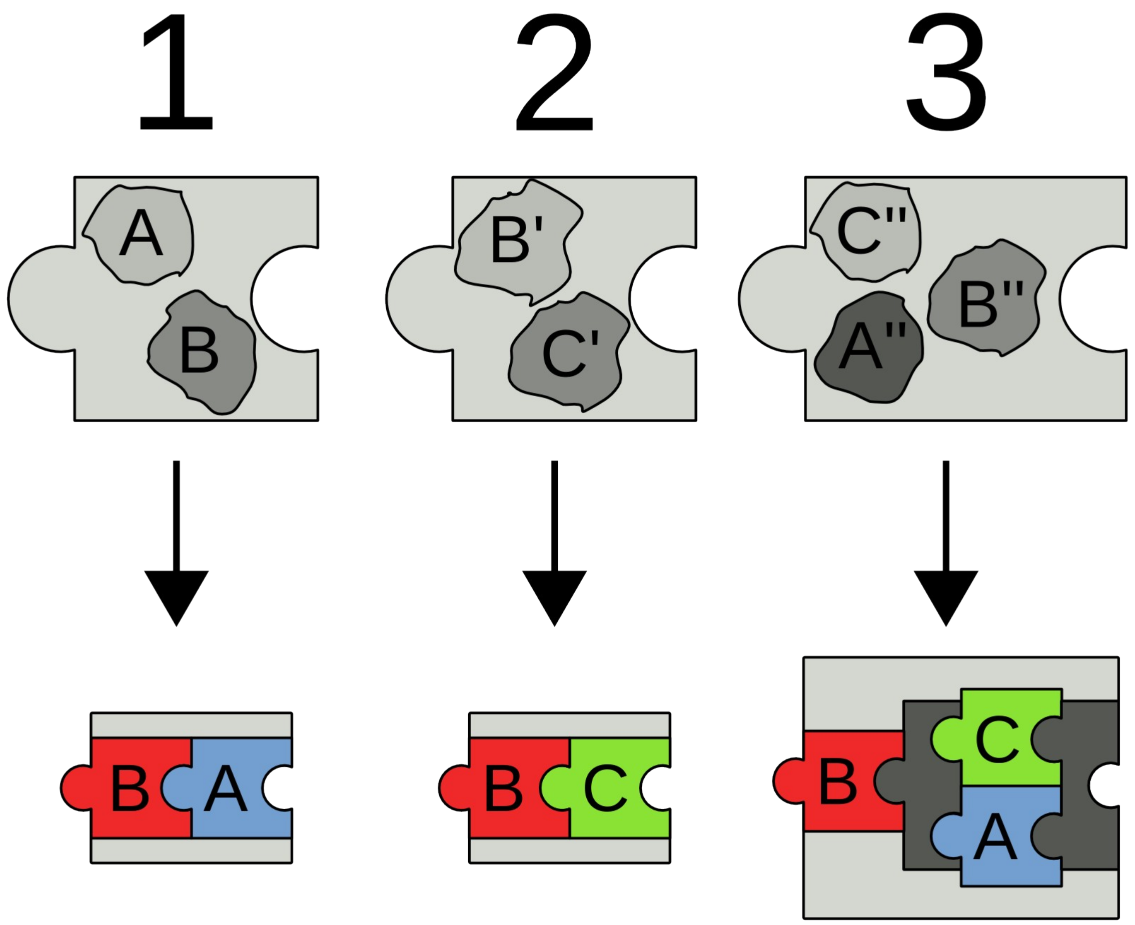
- Starting point: Multiple modules implementing variations of similar functionality

## Design Pattern - Modularization

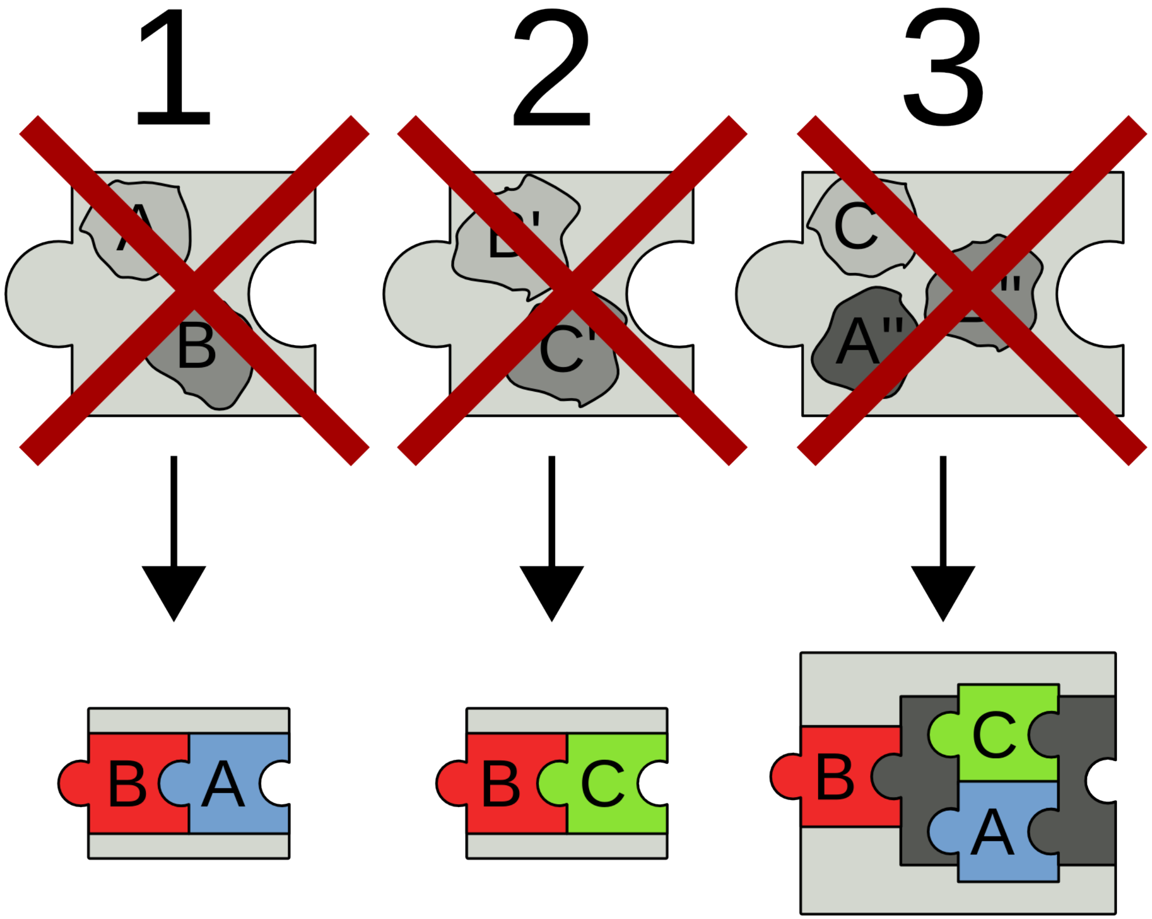


- Extract common functionality and put them in their own module
- Instead of having multiple similar versions have one version with configuration parameters
- Define standard interfaces to communicate between blocks

## Design Pattern - Modularization



## Design Pattern - Modularization

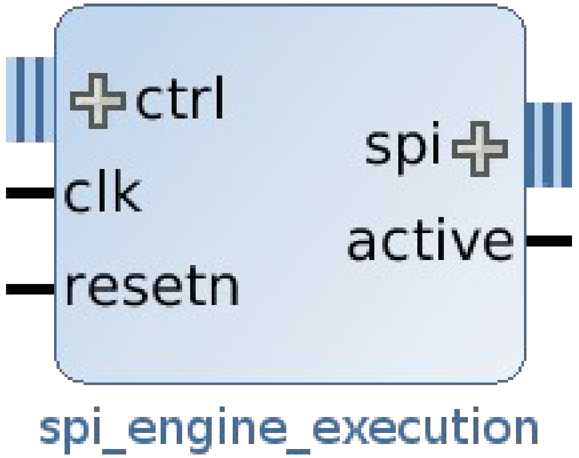


## Design Pattern - Modularization

- Combine multiple blocks using standard interfaces
  - Vivado makes this easy using IP integrator and/or hierarchies
- Sometimes glue logic is necessary
- Advantages
  - Easier to create new applications from existing pieces
  - If a block is improved all applications benefit from it

# SPI Engine - Command Stream Execution Engine

- Heart of the SPI engine framework
- Implements the low level SPI bus interface logic
- Has two interfaces
  - SPI Engine command stream slave
  - SPI bus master

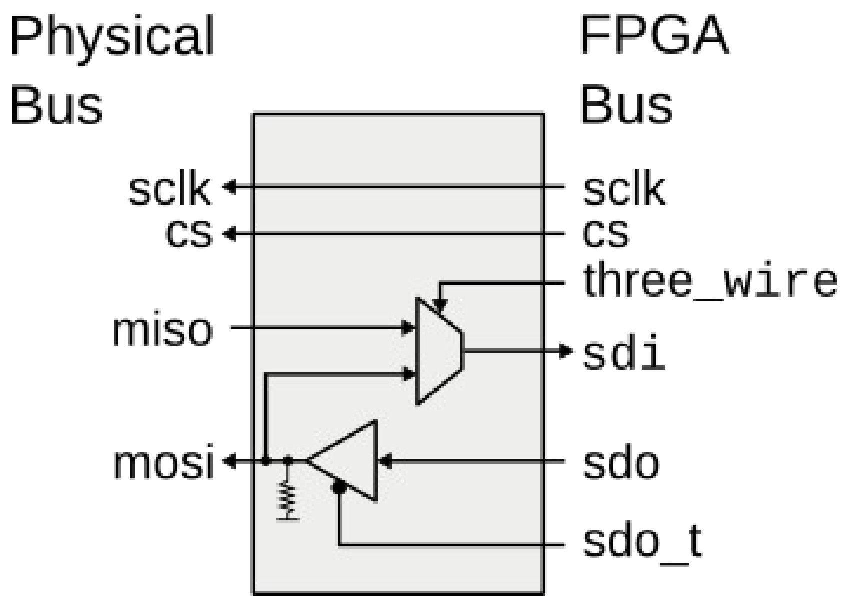


## SPI Engine - Command Stream Interface

- Command stream (master to slave)
  - Defines the execution engine behavior
- SDO data stream (master to slave)
  - Data written to the SPI bus
- SDI data stream (slave to master)
  - Data read from the SPI bus
- Synchronization event stream (slave to master)
  - Used to notify that a certain point in the command stream has been reached

## SPI Engine - SPI Bus Interface

- Logical SPI bus signals
- Low-level SPI signal
  - sclk, sdo, sdi, cs
- Control signals
  - three_wire, sdo_t



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

## Interface Logic
and
## Application Logic

## Design Pattern - Separate Interface and Application Logic

- Application logic defines what is done
  - E.g. write value 0x4 to configuration register 0x10
- Interface logic defines how it is done
  - Assert chip-select, output first bit, toggle clock signal, …

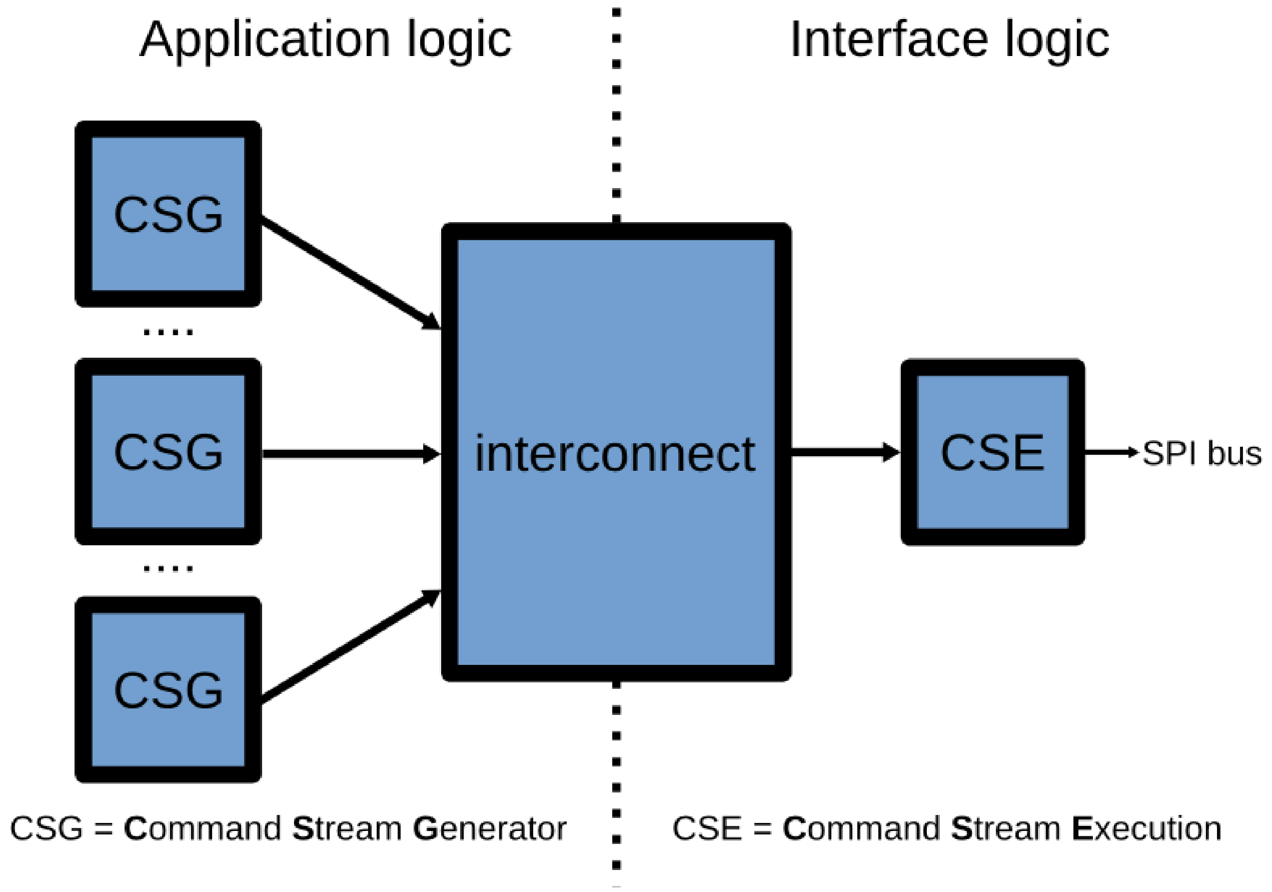
## Design Pattern - Separate Interface and Application Logic

- Multiple applications can use the same interface logic
- The same application can use different interface logic block
  - E.g. SPI bus or I2C bus for register map configuration

## Design Pattern - Separate Interface and Application Logic

- Interface logic defines a set primitives
  - Basic operations offered by the interface logic
  - Typically atomic operations where it does not make sense to further break them down
- Application logic combines primitives to accomplish complex tasks
  - Different applications will use different combinations

# SPI Engine - Separate Application and Interface Logic


Application logic
Interface logic
CSG
....
CSG
....
CSG
interconnect
CSE
SPI bus
CSG = **C**ommand **S**tream **G**enerator
CSE = **C**ommand **S**tream **E**xecution

AHEAD OF WHAT'S POSSIBLE™

## SPI Engine - Command Stream Generator

- Command stream generator implements application logic
  - Generates SPI Engine commands to control execution engine
- Interconnect makes it possible to connect multiple applications to the same execution engine
  - Allows more then one device on the same SPI bus

## SPI Engine - Command Stream Interface Primitives

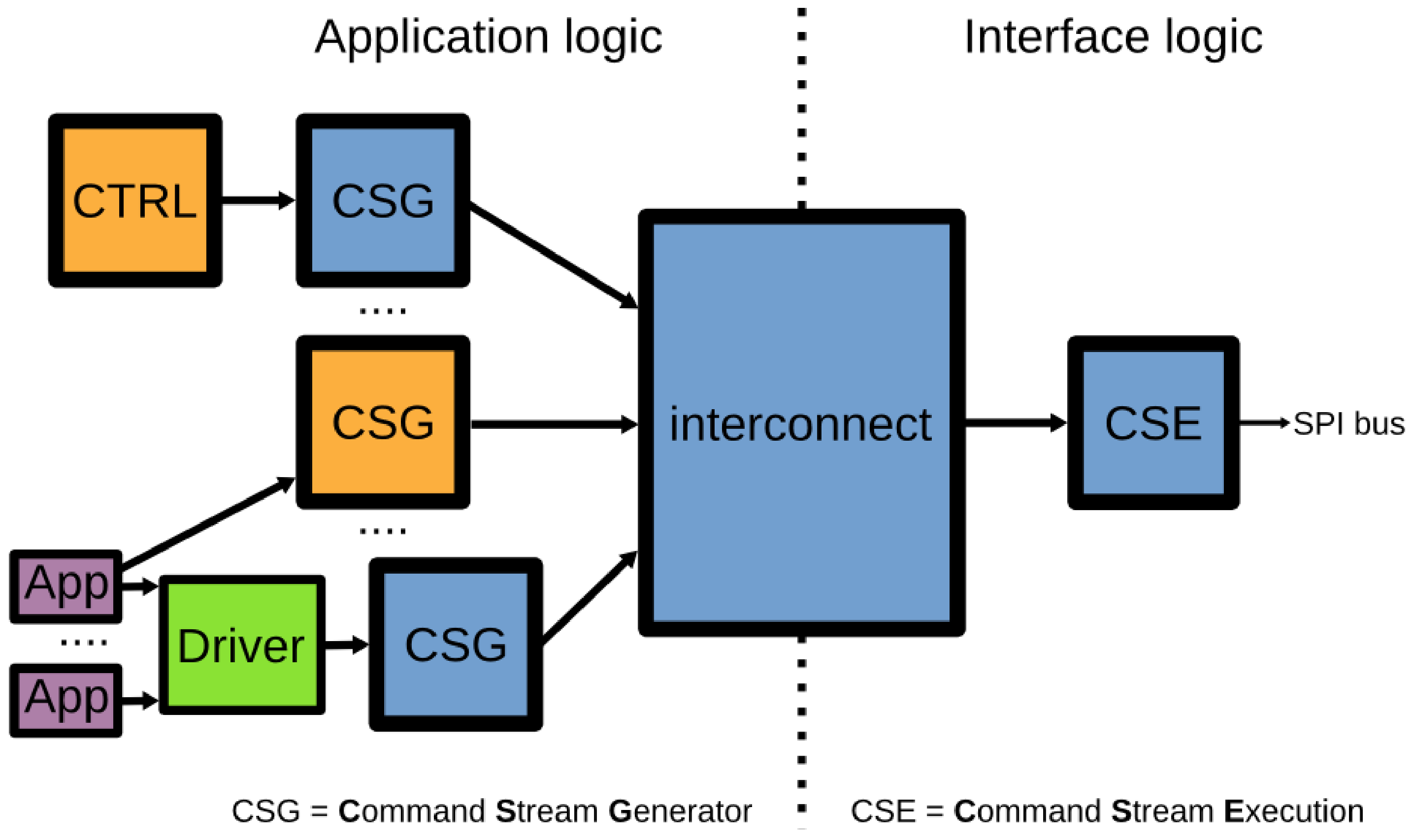
- Command stream interface primitives
  - TRANSFER: Read and/or write data to the SPI bus
  - CHIPSELECT: Changes the chip-select setting
  - CONFIG: Writes run-time configuration registers
  - SLEEP: Wait for a period of time
  - SYNC: Generate synchronization event
- Support for primitives that are not needed can be disabled
  - Design uses less resources

## Layering

## Design Pattern - Layering

- Application logic and interface logic can be stacked multiple layers
- Layers in the middle act as ...
  - interface logic to the layer on top
  - application logic to the layer underneath
- Allows very flexible architecture

## SPI Engine - Layering



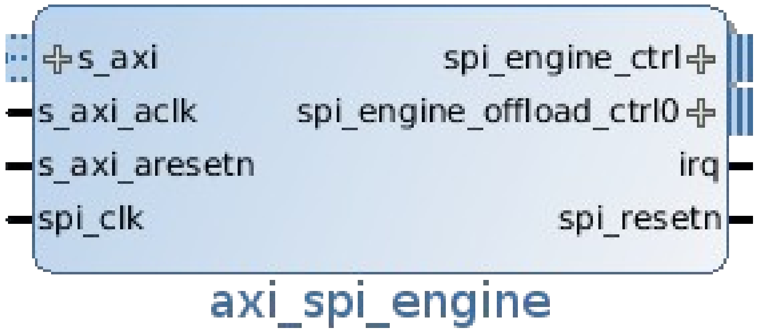
CSG = **C**ommand **S**tream **G**enerator CSE = **C**ommand **S**tream **E**xecution

🟦 Standard HDL block 🟧 Custom HDL block 🟩 Standard software 🟪 Custom software

AHEAD OF WHAT'S POSSIBLE™

## SPI Engine - AXI-SPI Engine

- Memory mapped access to command stream interface
  - Allows fully software controlled CSG
- AXI Lite control interface
- Interrupt driven control flow
- (Optional) FIFO for the control streams



## Partitioning

## Design Pattern - Partitioning

- Decide which functionality is implemented at which level
  - Software, HDL, …
  - Standard logic, Custom logic, ...
- Trying to find the sweet-spot from a cost perspective
  - Can be difficult
  - Depends on many different factors, result depends on the weighting of each factor
- Consider the existing ecosystem you want to integrate into

## SPI Engine - What's the problem again?

- Configure peripheral
  - Sampling rate, active channels, …
- Enable conversion
- Capture data (repeat)
  - Wait for data ready IRQ
  - Read result ← Critical section
    - Start SPI transfer
    - Wait for SPI transfer to finish
- Disable conversion

## SPI Engine - Partitioning

- Re-use existing software drivers
  - Reduces maintenances overhead
- Accelerate only critical paths in HDL
  - Keeps HDL lean and simple
  - Software only requires small changes
- Use standard blocks for acceleration when possible
  - Same logic blocks can be used in many applications

## Offloading

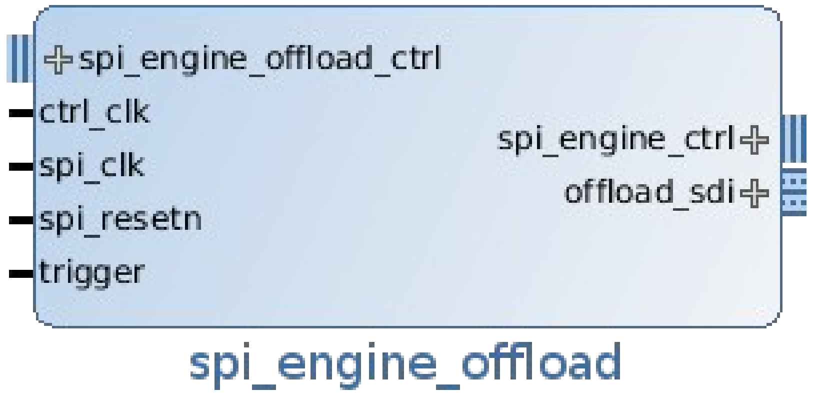
## SPI Engine - The Concept of Offloading

- Introduce the concept of SPI offloading
  - SPI controller takes care of tasks traditionally done by the CPU
- Interrupt offloading
  - SPI controller automatically executes pre-programmed transfer
  - Very low latency
  - A lot less overhead for repetitive transfers
- Data offloading
  - SPI controller is capable to send data to a hardware port
  - Avoids memcpy
  - Reduces latency

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

## SPI Engine - Offload module

- Internal RAM/ROM for CMD and SDO stream
- When trigger is asserted the CMD stream is send out
- Received data is send onto the offload_sdi interface
- Offload control interface allows dynamic reconfiguration



AHEAD OF WHAT'S POSSIBLE™

## SPI Engine - The Concept of Offloading

- SPI offloading is not specific to SPI Engine
  - Other SPI controllers can offer SPI offloading functionality
- Linux kernel will get SPI offloading support to its standard SPI API
- ADI converter drivers will be SPI offloading aware
  - Will use SPI offloading when available
  - Otherwise fall-back to CPU processing

## Conclusion

## SPI Engine - Conclusion

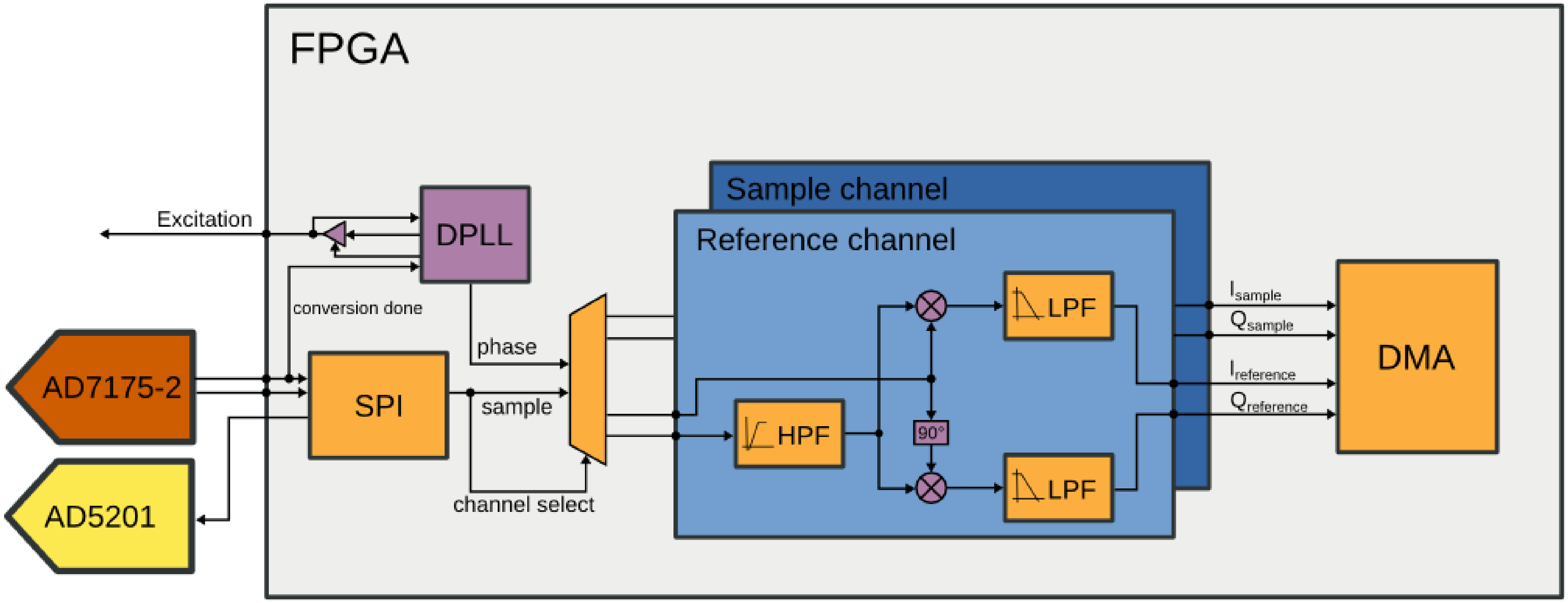
- A framework for creating SPI controllers
  - Standard blocks and custom blocks can coexist
  - Standard interfaces
  - Clear separation between interface and application logic
  - High flexibility and re-usability
  - Extensive documentation
- SPI-Engine pipeline is assembled from SPI-Engine blocks

## Case Study

## Case Study - Colorimeter

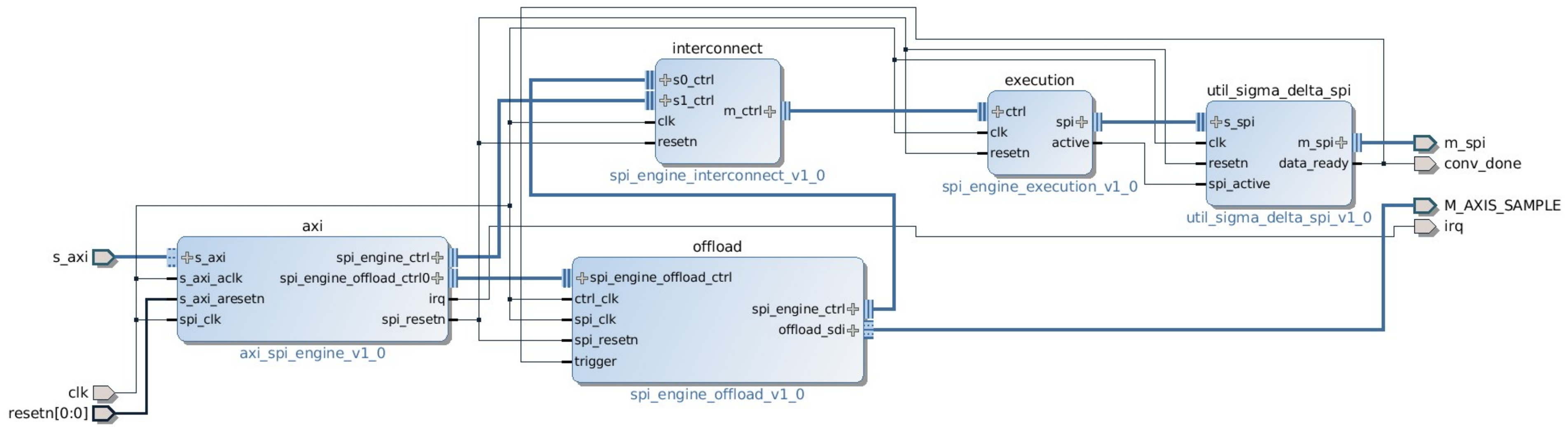
- Colorimeter application
  - Uses AD7175-2 ADC (50kHz)
  - Uses AD5201 digital potentiometer
  - Both on the same SPI bus
- AXI SPI-Engine used for general register access to both parts
- SPI-Engine offload used for low-latency, high-throughput sample conversion
  - Conversion results passed to processing pipeline
- http://wiki.analog.com/resources/eval/user-guides/eval-cn0363-pmdz

## Case Study - Colorimeter



AHEAD OF WHAT'S POSSIBLE™

## Case Study - Colorimeter



AHEAD OF WHAT'S POSSIBLE™

Thanks

## More Information

- http://wiki.analog.com/resources/fpga/peripherals/spi_engine

AHEAD OF WHAT'S POSSIBLE™

## Questions?