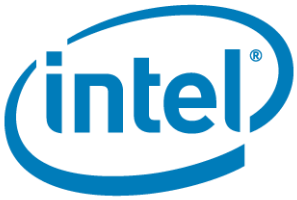

CFG AMBA IP Feature Addendum – Configurable Slave

Version: ORION – 19.01

Revision: A.0

Intel Confidential

CFG AMBA IP Feature: Configurable Slave

About this document

This document describes the CFG AMBA Configurable Slave and usage model where user can use as reference when integrating the model into their target system.

Audience

This document is intended for users of NocStudio:

- NoC architects, designers and verification engineers
- SoC architects, designers and verification engineers

Prerequisites

Before proceeding, you should generally understand:

- Basics of Network on Chip technology
- AMBA4 specification

Related documents

The following documents can be used as a reference to this document.

- CFG NocStudio Orion AMBA User Manual
- CFG Orion IP Integration Specification
- CFG Orion Technical Reference Manual

Customer support

For technical support about this product and general information, contact CFG Support.

Revision History

Revision	Date	Updates
0.0	Jan 07, 2019	Initial Version
A.0	Feb 24, 2019	No change, revision update

Contents

List of Figures

No table of figures entries found.

List of Tables

1 ARCHITECTURE

The Configurable Slave is an AXI agent that can be added to an AMBA NoC as a functionality supplement. This module can be configured at design time with different kinds of on-chip storage, such as SRAM or register files. This on-chip storage is wrapped with control logic allowing straightforward instantiation within an AMBA system.

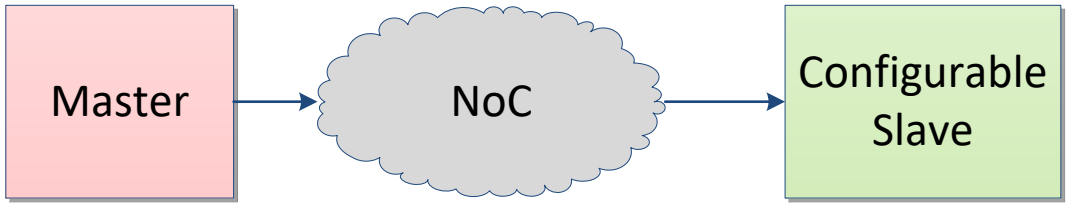

Figure 1: Configurable slave System View

The Configurable slave looks like any other slave agent, connecting to the NoC through a slave bridge. Multiple configurable slave components can be added to the network to increase bandwidth or to have lower latency by moving storage closer to a particular source.

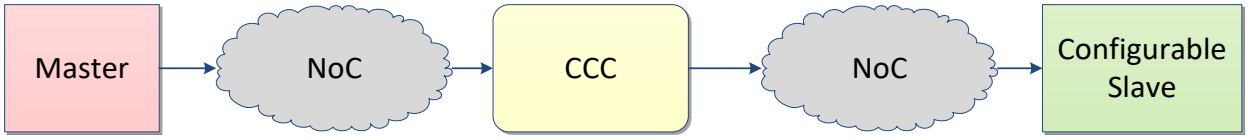

Figure 2: Configurable slave in Gemini System

The Configurable slave can also fit into a Gemini system either as a non-coherent slave, or as a coherent slave. A coherent slave means CCC will track addresses for this slave and handle cache coherency functions. The slave would then act as a backing store for the coherent range.

The configurable slave can be designed with multiple storage components for. It can instantiate SRAM arrays, Register Files, or Flip Flops, or a combination of them. Each storage component can be sized independently of the others. Multiple instances of an array can be used for banking, either to increase bandwidth or to limit the size of each array instance.

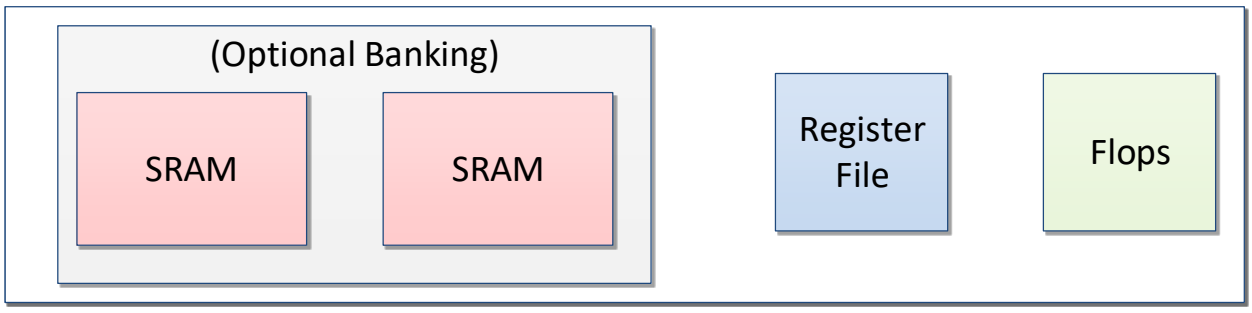

Figure 3: Configurable Storage

One of the storage types is SRAM. This is a single-ported SRAM. The size of each SRAM array must be power-of-2 number of array elements. Each array can have variable latency and bandwidth, depending on the implementation of the SRAM (these values are given as part of the specification). Since bandwidth may be a limit, an option is provided to allow a power-of-2 number of banks of SRAM, allowing parallel accesses for increased bandwidth. Banking will be based on the bus width of the slave. So a 128bit data path will bank on 128bit granularity, so consecutive addresses will access different banks.

For smaller structures, or some non-power-of-2 size storage, cache line (64B) storage blocks can be added using either flops or Register Files.

Banking of arrays happens on the granularity of the data bus width, and will use the next lowest order bits depending on the size of the banking.

1.1 Configurable Data Bus Width

The bus data width of the configurable slave is configurable. Larger widths can be used to increase bandwidth. The bus data width also defines the width of the logical arrays. So an SRAM will have a data width matched to the bus width. Configurable slave can support data width of 4, 8, 16, 32, or 64 bytes.

1.2 Configurable Address Bus Width

The address bus width of the configurable slave is also configurable. This can allow address bit truncation to reduce area and to avoid needing to manipulate the address before sending to the slave.

1.3 Exclusive Monitors Option

The configurable slave can optionally have AXI Exclusive monitors. This option will default to off, so less hardware will be instantiated. The user must configure this option in NocStudio to enable it.

The exclusive monitors will be set up based on NocStudio definitions of the agents, which means it can fully populate as many Exclusive monitors as needed by the system, including those needed for logical agents (i.e., sub-agents).

The Exclusive monitors are set up during an Exclusive ReadNoSnoop command (ARLOCK==1). A store by another agent to the same 64B address will reset the validity of the monitor. An Exclusive WriteNoSnoop (ARLOCK==1) will perform a conditional store. If the monitor has been cleared before the conditional store is processed, the store will be canceled.

The Exclusive Monitors are designed to track 64B of data. If different semaphores are located within the same 64B block of data, any successful modification of the line will clear the monitors tracking that address.

1.4 Bandwidth of Configurable Slave

The configurable slave is designed to break up bursts into single beat operations and issue them in order to the storage modules. The AR and AW/W interfaces provide a very small amount of storage to allow request to arrive, arbitrate, and send requests to the storage modules with the beat-specific address.

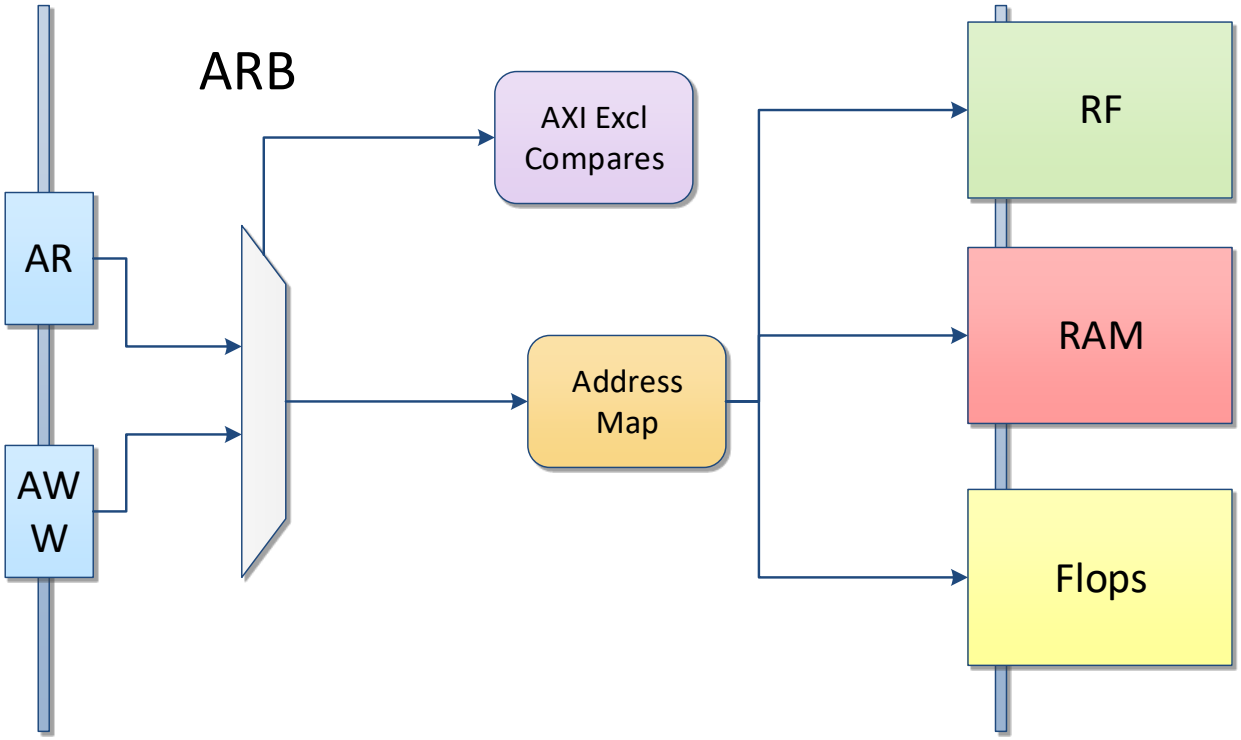

Figure 4: Logical view of arbitration

As shown in the diagram above, reads and writes arbitrate together to send a single beat-sized request to the storage components. This means that the configurable slave shared bandwidth between reads and writes. A 32B interface will only support 32B per cycle of bandwidth, shared between reads and writes.

When a read or write wins the arbitration, it will continue until it has completed. The configurable slave does not interleave the requests, to avoid unexpected behavior where a read to the same address as a write returns partially new and partially old data.

1.5 Configurable Slave Clock Domain

The configurable slave runs at a single frequency for all internal components. There is no clock crossing per individual storage entry. The slave is expected to run at NoC frequencies, but like any slave on the NoC, it can be configured to run at a different clock with clock crossing within the NoC.

1.6 Adding ECC Protection

The Configurable Slave can support ECC protection on each storage bank. The ECC will protect data based on the data width of the slave. So a 32B bus data width will have ECC protection of 32B, which can use 10 bits. Smaller bus width will end up with more ECC storage bits for the same total capacity.

Table 1: ECC Overhead based on bus width

Data Width	ECC Bits per Data Width	ECC percentage per 64B
512 Bits	11	2.1%
256 Bits	10	3.9%
128 Bits	9	7.0%
64 Bits	8	12.5%
32 Bits	7	21.9%

The ECC protection forces a functional change as well. If partial data writes are sent to the configurable slave storage with ECC protection, the Configurable Slave will perform a read-modify-write in order to set the ECC bits correctly. This can reduce the bandwidth of a storage array, so ECC enabled storage should have additional banking. If partial writes are rare, additional banking may be unnecessary.

The ECC algorithm is SEC/DED (single-bit error correct, double-bit error detect).

1.7 Configuring a Storage Module

The following values must be specified for each storage module:

Table 2: Storage component parameters

Parameter	SRAM	Register File	Flops	Description

# of banks	Y	Y	Y	Default 1. Can be used to have multiple banks
Total Capacity	Y	Y	Y	SRAM must be power of 2 capacity. RF and flops can have any multiple of data width.
ECC Protection	Y	Y	Y	Enables ECC protection. Will increase storage to hold ECC bits.
Read Latency	Y	Y	Y	For SRAM, this latency includes RAM latency, and any other cycles needed for transportation to and from array. For flops and RF, this only includes the transportation latency.
Bandwidth	Y	-	-	For SRAM, this indicates the throughput
mem_info_in	Y	Y	-	This parameter allows a set of input signals to be routed to the array through the top level module. These are typically used for BIST or other DFT signals.
mem_info_out	Y	Y	-	This parameter allows a set of output signals to be routed to the array through the top level module. These are typically used for BIST or other DFT signals.
Address range	Y	Y	Y	Specify address ranges for these slaves

1.8 CONTROL AND STATUS REGISTERS

This section will describe the various control and status registers within the configurable slave. Details are TBD.

1.8.1 Interrupt Event Register

This register will track Interrupt events within the configurable slave. This includes:

1. ECC single bit errors
2. ECC double bit errors
3. Address decode errors (access to unmapped area)

4. Security violations (access with insufficient security permission)
5. Even counter overflow

1.8.2 Interrupt Mask Register

This register controls which of the events in the interrupt event register actually trigger an interrupt.

1.8.3 Event counter

This register contains the current event counter count. It can be written to initialize the counter to a specific value, or read to see the current status.

1.8.4 Event counter vector

This register will contain a vector of internal events that can be counted. If multiple events are selected, the AND of the specified values will be selected. Event list is TBD.

1.8.5 Error Event Info

This register will track ECC errors (single or double bit), tracking information about number of each event, one address that caused an ECC error (first double bit or first single bit).

1.8.6 Indirect Storage Access Registers

A set of register that can be used as a backdoor access to the storage arrays. Can be used to read or write array, or to atomically modify them (such as to induce a single or double bit ECC error).

1.8.7 Security Filters

A set of registers (configurable number) that can be used to mark certain ranges as either secure or non-secure. Accesses with insufficient permissions will be rejected.

1.9 Coarse Grain Clock Gating

The configurable slave has built in clock-gating support. It uses a hysteresis value to decide when the agent has been idle for long enough to induce clock-gating. New requests will automatically trigger a wake-up of the clock, but there is one-cycle penalty to perform the wakeup.

2 NOCSTUDIO COMMAND AND PROPERTIES

2.1 ADDING CSB AND SETTING PROPERTIES

2.1.1 add_config_slave_block

The Configurable Slave Block (CSB) can be added using the **add_config_slave_block** command.

```
add_config_slave_block <name>
                       [ color <value> ]
                       [ pos     <pos_id | xy position> |
                         phy_pos <xy position in um> ]
                       [ size <size_x> <size_y> |
                         phy_size <size_x in um> <size_y in um> ]
                       [ bridge <name> csbs
                         [<pos_id or xy position>[direction]]]
```

The block name, color, position, size, bridge names and bridge positions can all be specified similar to the **add_host** command. The only difference being that a CSB can only have exactly one slave bridge port of type *csbs*.

2.1.2 add_csb_storage

Multiple storage components can be added to a CSB. Each of the storage component can be defined as flop based, regFile or SRAM based implementation.

```
add_csb_storage -name <name> -csb <previously defined CSB name>
```

Using csb_storage_prop, user can define all csb related properties listed below. Please refer to NocStudio help manual for details and examples.

```
csb_storage_prop add_csb_storage -name <name>
                                 -csb <previously defined CSB name>
```

Name	Range	Comment
csb_storage_addr_range	Custom value	Specify a sub-address range for a storage inside the CSB
csb_storage_bandwidth_delay	1 ~ 3	Set the bandwidth of the storage component

csb_storage_capacity	Number of bytes / 1024	Set the capacity of the storage component in KB. Decimal value is allowed when the storage size is less than 1KB. Please see Equation 1.
csb_storage_ecc_enable	yes or no	Enable or disable the ECC on the storage component
csb_storage_latency	1 ~ 16	Set the latency in cycles for a SRAM storage component
csb_storage_memory_in_width	1 ~ 1024	Set the number of info input bits to the storage component for BIST or DFT purposes
csb_storage_memory_out_width	1 ~ 1024	Set the number of info output bits to the storage component for BIST or DFT purposes
csb_storage_num_banks	1,2,4,8,16,32,64	Set the number of banks for the storage component
csb_storage_secure_access_enable	yes or no	Enable or disable the need for secure access to the storage component
csb_storage_type	flop reg_file sram	Set the storage type for the storage component

Equation 1 calculation of csb_storage_capacity

The minimal number of enties in a storage component is 2 (fixed). The width is determined by the host_prop csb_data_width (e.g. W). The number of banks is defined as "B". To calculate the storage capacity, please refer to the equation below:

$$csb\_storage\_capacity = \frac{2 \text{ x } W \text{ x } B}{8 \text{ x } 1024}$$

2.1.3 Host property - csb_data_width

Using host_prop, user can define the data width of the Configurable Slave Block. Please note that the interface (AXI) data width property is derived from this value.

```
host_prop <csb name> csb_data_width <width>
```

2.1.4 Host property - csb_exclusive_support

Using host_prop, user can enable/disable the exclusive monitor support in CSB.

```
host_prop <csb name> csb_exclusive_support <yes|no>
```

2.2 ADDITIONAL NOCSTUDIO COMMANDS

2.2.1 list_csb_storage

This command lists all storage components in all CSBs of a design.

```
> list_csb_storage
CSB     Storage    Storage    Num of    Address
Name    Name       Type       Banks     Range
-------------------------------------------------------------------------------------------
csb0    st0        sram        4        range0 0x000_0000_0000_0000:0xfff_ffff_ffe0_0000
        st1        reg_file    2        range0 0x000_0000_0020_0000:0xfff_ffff_ffff_ffe0
        st2        flop        1        range0 0x000_0000_0030_0000:0xfff_ffff_ffff_fc00
csb1    st0        sram        4        range1 0x000_0000_0100_0000:0xfff_ffff_ffe0_0000
csb2    st0        sram        4        range2 0x000_0000_0200_0000:0xfff_ffff_ffe0_0000
csb3    st0        sram        4        range3 0x000_0000_0300_0000:0xfff_ffff_ffe0_0000
csb4    st0        sram        4        range4 0x000_0000_0400_0000:0xfff_ffff_ffe0_0000
csb5    st0        sram        4        range5 0x000_0000_0500_0000:0xfff_ffff_ffe0_0000
csb6    st0        sram        4        range6 0x000_0000_0600_0000:0xfff_ffff_ffe0_0000
csb7    st0        sram        4        range7 0x000_0000_0700_0000:0xfff_ffff_ffe0_0000
```

2.2.2 del_csb_storage

This command allows user to delete a previously added CSB in a design. It is most useful in NocStudio GUI interactive mode during design exploration.

3 INTEGRATION

3.1 NOC SANITY TESTBENCH

Similar to CCC, LLC or any other CFG developed agents, NocStudio supports NOC sanity testbench and CSB is just one of the NOC agent components. Please refer to IP Integration Specification for running sanity test.

3.2 SRAM / REFFILE INSTANTIATION

For each user modifiable RTL module, SRAM or RegFile, NocStudio supports SRAM sanity testbench. After instantiating the technology dependent SRAM or RegFile modules in the wrapper file, user can invoke sram sanity testbench individually to validate the replacement has been done correctly.

Please note that this feature is not part of the 1901 release.

3.3 VERIFICATION COMPONENTS

A newly added checker, ns_cslv_checker.sv, can be found under the noc_verif_ip sub-folder. Please refer to IP Integration Specficiation for checker usage model.

3.4 FAST INITIALIZATION FOR CSB

By default, hardware explicitly initializes CSB after reset. This can slow down simulation time. Similar to CCC or LLC initialization, a backdoor fast initialization mechanism can be used to minimize the simulation initialization cycles: `define NS_FORCE_RTL_CSLV_SHORT_INIT.

Please refer to IP Integration specification for details.

2200 Mission College Blvd
Santa Clara, CA 95054
www.intel.com