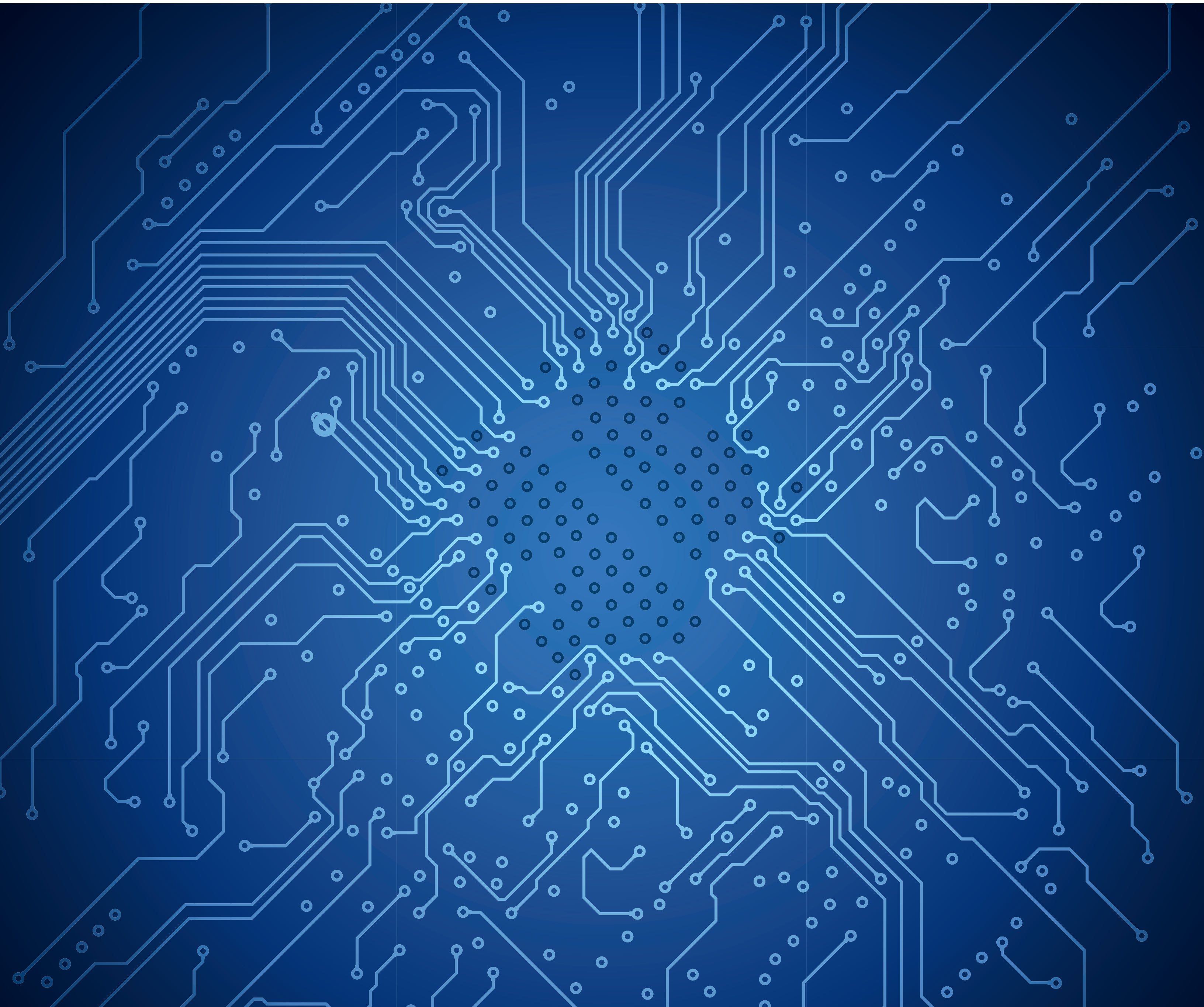
Designer's Guide to AMBA Generic Flash Bus specification (GFB)

arm

September, 2018

White Paper



Non-Confidential Proprietary Notice

Contents

1 Overview

Generic Flash Bus details

1.1. Executive Summary

This whitepaper discusses the new Arm AMBA Generic Flash Bus (GFB) interface specification and the main considerations for designers. It walks through the key components of the GFB specification, how it works and the important things to think about when adopting the specification. The full GFB specification is in document IHI-0083 "AMBA Generic Flash Bus Protocol Specification" available on Arm's Developer website here: **https://developer.arm.com/docs/ihi0083/latest**

1.2. Origin of the problem

There is a wide variety of applications using embedded Flash technology. Each vendor and process technology can have differences in the implementation, but the key functionality of using a non-volatile storage is very similar for every use-case. When system designs using this functionality are retargeted to newer processes, or different vendors are selected for production, then there is a design cost to port the flash controller to the other technology.

1.3. Purpose of GFB

The Generic Flash Bus (GFB) simplifies the integration of embedded Flash controllers in large subsystems by providing a simple interface between the system and the Flash, or other non-volatile storage technology. The GFB defines a boundary between two sides of the Flash controller, a master side and a slave side. The master side is closely related to the system that contains general functionality that is supported by most eFlash macros. The slave side is closely related to the eFlash macro used for a specific process. This ensures reusability of the general functions with different processes. The GFB represents the data path and provides direct access to the Flash memory resources. The control path, such as access to registers in the Process-Specific Part, or different power modes of the Flash, is managed over other interfaces.

1.4. Flash controller structure

The following figure shows the structure of the Flash controller defined by the GFB.

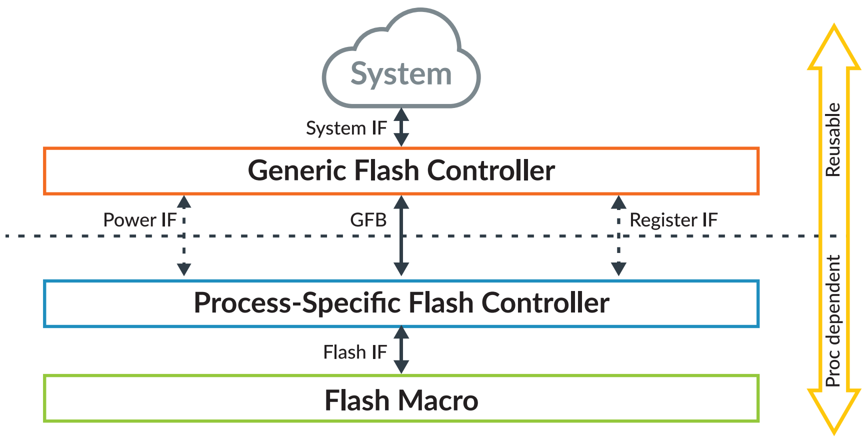


Figure 1 System architecture using GFB

The Generic Flash controller is the master side, which is technology independent, this means it can easily be retargeted. The Process-Specific Flash controller is the slave side, which deals with the different requirements of the selected technology. It also serves as a light-weight converter for the data traffic between the GFB and the embedded Flash macro. The two parts together form a complete Flash controller, but each part can be designed independently of the other, as the GFB defines the protocol that needs to be supported by both sides.

2 Designing with GFB

2.1. Functional partitioning

The GFB specification does not strictly define the functions in either the master or the slave side. However, to create an efficient design with GFB, there are recommended rules for partitioning. Embedded Flashes generally execute a single command at a time, so the GFB interface is aimed to be the channel where all data-related transfers are visible. The Generic Part initiates the transfers but the protocol allows the Process-Specific Part to do housekeeping tasks in where necessary. The Generic Part should be aware of every action happening within the Process-Specific Part.

2.2. Generic Flash Controller features

The Generic Part serves as the master of the interface, and therefore controls and drives every transaction towards the Flash. The Generic Part can be viewed as a converter from any kind of standard system interface that is capable of reading, writing, and erasing from a non-volatile memory to GFB.

The following list of features is an example of what may be part of a Generic Flash controller:

- Read, write, and erase support to the memory.
- Buffering for data width conversion.
- Arbitration between multiple system interfaces.
- Caching of transfers for energy efficiency.
- Security protection of memory regions.
- Low-power management.
- Data encryption.
- Emulation of different memory behaviors

As shown in Figure 1, the Generic Part may also be responsible for supporting a power management and a register interface towards the Process-Specific Part. These interfaces are required in the Generic Part to enable a complete master role, where every aspect of handling the embedded Flash within the system is going through the master.

2.3. Process-Specific Flash Controller features

The Process-Specific Part serves as the slave on the GFB interface. It receives and executes every command generated by the master, but in exceptional cases it can also execute internal tasks and indicate this towards the master. The GFB protocol is aimed to match the most common features of the embedded Flash technology so the conversion from GFB to the appropriate can be done with minimal logic.

The following list of features is an example of what may be part of a Process-Specific Flash controller:

- Command conversion from GFB to the Flash interface.
- Control registers for setting the timing of the signals of the Flash interface.

- Initialization of the Flash at startup.
- Redundancy handling (remapping of faulty banks or pages).
- Flash protection against damaging due to brown-out or reset in the system.
- Error Correction Code (ECC) calculation.
- Built-in Self Test (BIST) execution.

The structure of the Process-Specific Part can further be segmented into smaller internal blocks as shown in the following figure.

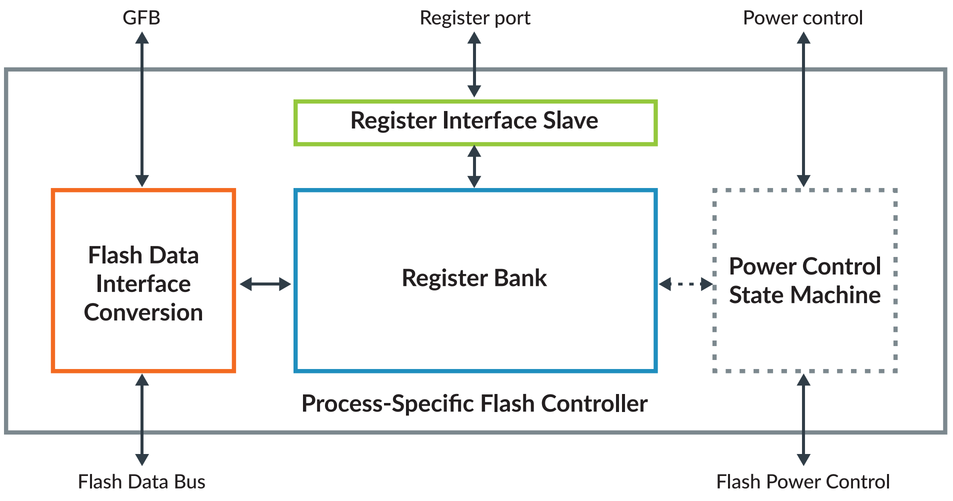


Figure 2 Process-Specific Flash Controller hierarchy

The figure shows that the GFB interface is connected to an interface conversion block. This turns the GFB commands into Flash control signals based on the timing settings or ECC control signals coming from the register bank. The registers are configured via the dedicated register port, where the Generic Part is the master. All the power related signals are handled by a dedicated Power Control State Machine, which is also managed by the Generic Part via the power control interface.

Arm recommends separating the power control state machine from the rest of the process-specific logic, as it needs to be placed into a different (possibly always-on) power domain to control the power switches of the flash macro. This is highlighted in the figure by the dashed lines around the Power Control State Machine.

2.4. Initialization considerations

At system startup, the Flash may execute initialization steps before it is fully operational. Power sequencing is triggered by the Generic Part over the Power Control interface, so the Generic Part is aware of the operational state of the Process-Specific Part. Additional tasks, such as automatic BIST execution and redundancy page mapping may also be part of the initialization sequence. Until the initialization is complete, the Process-Specific Part indicates this by setting the FREADY signal on the GFB interface to low.

It is not mandatory for the Generic Part to block GFB transfers until the power control handshake is done, and the Process-Specific Part is in a fully powered mode. It is the responsibility of the Process-Specific Part to only execute the incoming commands when the attached embedded Flash is operational.

2.5. Power domain considerations

Arm recommends placing the Generic Part and the Process-Specific Part within the same power and clock domain. However, the embedded Flash macro may require its own power domain. To manage it separately from the rest of the system a Power Control State Machine is required, instantiated within or outside the Process-Specific Part. The design of the Power Control State Machine needs to allow placing it in a relatively-on power domain and a different reset domain compared to the rest of the Flash controller. This enables the power to be managed independent of the logic within the Flash controller.

After coming out of reset, it is also recommended that the Power Control State Machine provides information to the Generic Part about the actual power state of the Flash. This enables the Generic Part to continue the power management of the Flash from the point before the reset happened. This is also a reason to place the Power Control State Machine to a separate reset domain.

2.6. Housekeeping considerations

The Process-Specific Flash Controller can initiate housekeeping to execute internal processes. The Generic Part needs to acknowledge these activities so that it can deny power and clock quiescence requests when housekeeping is ongoing. The Process-Specific Part can extend the duration of the register access or pull the FREADY low (or combine this two) until the housekeeping operation is finished. In case the Generic Part is not aware of the housekeeping in time, then it can result in accepting a clock quiescence during a critical operation in the Flash macro, that could result in a device damage without clock. The Generic Part needs to be aware of the ongoing activity in every clock cycle, so either the FREADY is pulled low on GFB or the register interface is busy when preparing the housekeeping.

However, the GFB interface needs to be in IDLE to start the housekeeping, so the driver software is expected to synchronize these activities between the control and data path.

2.7. Row Write considerations

The ROW WRITE commands require special attention to utilize the efficiency gain of staying in the programming mode for more than a single command. A GFB command has two phases, an address phase and a data phase. When the first ROW WRITE command enters data phase, the driver needs to prepare the next ROW WRITE before the data phase ends to benefit from the ROW WRITE sequence. This allows the Process-Specific Part to snoop the FCMD bus when the actual write is finished internally within the eFlash macro and it can then decide to continue or close the programming sequence. The Generic Part may provide interrupts to the driver software to help keeping the flow of ROW WRITE commands. The phase changes of a command may serve as the source of the interrupts. Buffering and acknowledging commands within the Process-Specific Part is not ideal as this would put a requirement to the Generic Part to constantly provide new ROW WRITE commands on the GFB bus. Any caching or buffering is better suited in the Generic Part, so that the Process-Specific Part can remain as simple as possible.

2.8. Software considerations

The driver for the Generic Part and the Process-Specific Part are separated to maintain portability between different platforms. The Generic driver can be reused to support all generic commands defined by GFB, and all process related timing settings, ECC considerations need to be placed within the Process-Specific driver.

3 Future technologies

3.1. Support for upcoming NVM technologies

The GFB aims to be a simple and generic interface to cover all basic functions supported by any NVM storage device. MRAM and ReRAM technologies are predicted to serve as a replacement for eFlash in the long term. All these technologies store information in different ways, but the interface and access capabilities are similar. The Process-Specific Part that belongs to these new technologies shall hide all these differences, and still provide an optimal access to the memory content. The Generic Part and the related driver software shall allow simple read and write access to the memory, even if there are commands that are not supported by the Process-Specific Part.

4 Summary

4.1. Conclusion

This whitepaper discussed the different aspects of using the new Arm AMBA Generic Flash Bus (GFB) interface specification. The recommendations followed in this whitepaper help designers to easily and successfully adopt the new standard in their designs. For full specification details see the Arm Developer website here: **https://developer.arm.com/docs/ihi0083/latest.**