CROSSOVER TO MEMORY EXPANSION WITH ADESTO ECOXiP AND NXP'S i.MX RT CROSSOVER PROCESSORS

Donnie Garcia, NXP Semiconductor: Solutions Architect
Eyal Barzilay, Adesto Technologies: System and Software

INTRODUCTION

With 8.4 billion connected "things" having shipped in 2017, the internet of tomorrow is clearly upon us. We have entered a new age of human to machine interactions where technology is guiding many aspects of our lives. For a variety of end devices such as wearables, home monitoring nodes and industrial controllers, the capabilities of the embedded processor play a vital role in addressing the insatiable demand for a higher order of functionality. This has led to industry focus on machine learning enabled by vision and audio processing to bring the computation needed to make decisions at the edge node. These capabilities require elevated levels of processing performance and memory space for MCUs. The push for processing has led to a new breed of semiconductor device which does not fit into a traditional definition of a microcontroller. The 'Crossover Processor' integrates attributes of a microprocessor such as higher CPU speeds, multimedia interfaces and expandable memory into a microcontroller form factor built for cost effectiveness and fastest development time. This new crossover processor class of device provides embedded developers the ability to solve many problems in today's fast-moving technology markets.

Collaboration between semiconductor manufacturers and memory vendors plays a vital role in ensuring that the embedded systems that are brought to market achieve performance and usability goals. This is accomplished by closing the gap between the typical embedded flash device and the crossover MCU with external memory. Using external memory, crossover processors have the ability to support massive amounts of software and data memory space. This is done with keeping the same look and feel of a traditional embedded flash microcontroller. Together, the right serial flash memory coupled with a capable processor address the challenges of performance, security, power consumption and development experience.

For the processor, considering eXecute-in-place (XiP) from the start of the semiconductor chip design brings together a microarchitecture that is built for memory expansion. For serial flash, there are advancements in the interface protocol, low energy read of memory, and read while write programming capabilities to address these challenges. This paper will provide an overview of how performance and usability are addressed for systems depending on external memory. The following sections will explore how the Adesto EcoXiP serial flash and the i.MX RT1050 crossover processor pair together to provide the embedded platform needed to conquer the challenges of future embedded designs.

TABLE OF CONTENTS

OVERVIEW OF SERIAL NOR FLASH AND EXECUTE IN PLACE

Serial NOR flash comes in the form of integrated circuits (ICs) with a range of memory size and physical interface options. These memory devices typically operate at 1.8V or 3.3V, support 100 thousand write erase cycles, and can easily be placed on printed circuit boards. The serial flash IC allows embedded systems to easily introduce a non-volatile memory (NVM) with various packages ranging from the basic 8-pin to very small chip scale. There are many use cases for applying serial NOR flash to a system. Persistent data logging is one example of a common application use case which benefits from this technology. Another important use is storing and executing software for the ever growing embedded applications.

The eXecute in Place, or XiP, is a capability that allows a processor to execute code directly from external flash memory. Many embedded applications require connectivity stacks, audio processing, and vision. The amount of executable code for these functions has grown to substantial sizes. When considering these application requirements together for one embedded system, the capability of XiP with external flash is an essential enabler as it allows nearly limitless data space for the embedded system. In the semiconductor industry, thousands of capable microcontrollers are already integrating the type of memory controller needed to support XiP cability from Serial NOR flash.

Microcontroller Memory Architectures

For embedded processing, there are several common memory architectures as shown in Figure 1. Starting from the left, for most microcontrollers, internal non-volatile memory provides the execution space for the software. Here the NVM is all provided internal to the chip. There are advantages due to the system integration, but a limitation with regards to scalability. If the system needs more memory than what is provided internal to the processor, then external memory must be added. Often, external memory (such as EEPROM) is needed to store persistent data for other uses in the system as shown in the diagram.

The second architecture in the middle, is a copy-to-execute architecture. This means that the code is stored in external flash but copied to internal RAM at startup and then executed. In this case external NVM is used in conjunction with execute memory in RAM. This architecture, will be limited by the size of the internal SRAM memory. If the size of code is larger than internal SRAM, software must bring in portions of code as needed by the application. This copy to execute has penalties with regards to copy time and software complexity. Large internal SRAM size could have a significant impact on cost. Alternatively, if external DRAM is used, system cost can be reduced because of the low cost per bit for DRAM versus internal SRAM.

When using DRAM there are challenges with regards to power consumption. This is due to the volatile nature of the DRAM memory and the need for self-refresh for low power states of DRAM. Even if the code fits into SRAM, a low-power system would probably require shutting down the SRAM during sleep mode. This means that a copy to SRAM would be necessary on each transition from sleep to active mode. In other words, the system will be slow to wake up.

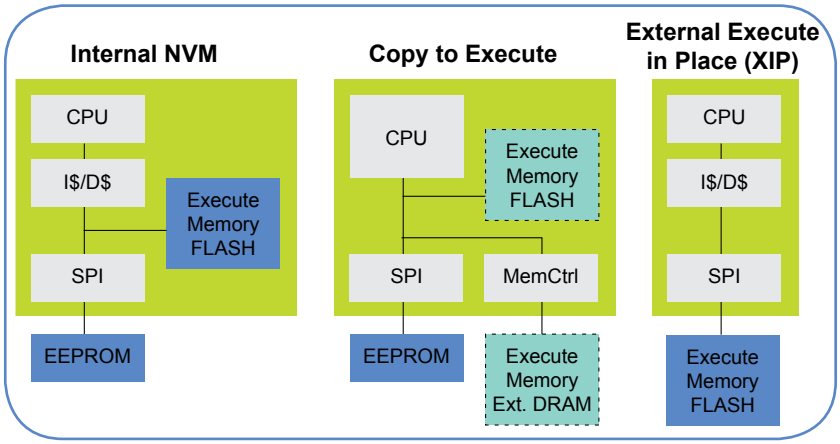

Figure 1: Memory architecture diagrams

Furthest to the right, the XiP architecture depends on the external memory for the execution of code. This memory architecture has advantages with regards to scalability. Designers do not have to face issues with over buying for a larger memory size to protect against software growth. The choice of external memory can be made for what is needed for the embedded design. This ensures that every penny spent on the processor components in the system goes towards relevant features for the end product. This architecture reduces both risk and design cycle times as the XiP system architecture can be scaled with only a change to the serial NOR flash in the bill of materials for the circuit boards. In addition, XiP brings an advantage in terms of power and fast wakeup from sleep mode.

Still, there are challenges when using this architecture. In the coming sections, we will discuss how these challenges are being mitigated by intelligent designs incorporated for both the processor and the serial flash.

How XIP is achieved

Central to the support of XiP is the integration with a smart SPI (Serial Peripheral Interface) host controller on the processor. Akin to a standard SPI, these host controller peripherals support a synchronous serial protocol that depends on data and clock signals. For example, Figure 2 shows the most basic SPI read where an opcode and address are sent to a slave device via Serial In (SI), and data is returned to the master device via Serial Out (SO).

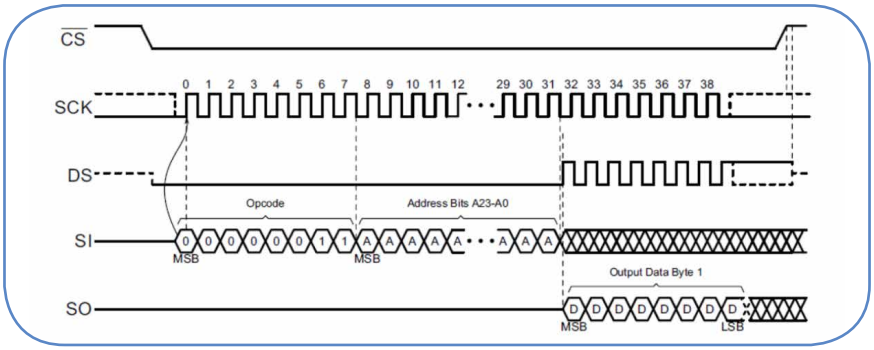

Figure 2: Example SPI data transfer

In addition to operating as traditional SPI, in order to better support the XiP use case, these enhanced peripherals also operate as system memory controllers. They can take internal bus transfers generated in the chip and translate them into the right serial commands needed to interact with the external memory. In this way, data transfers from the external memory are accelerated by hardware. The instructions and data residing in external serial NOR flash are directly fed into the CPU pipeline or other chip peripherals based on memory transfers occurring inside the microarchitecture of the chip.

FlexSPI Memory Controller

One such memory controller is the FlexSPI. FlexSPI is NXP's latest generation of the serial flash memory controllers. The block diagram in Figure 3 represents the FlexSPI which is integrated on the i.MX RT crossover processors. The 64bit AHB bus is the interface to the system bus which will come from a CPU or other on-chip masters such as an LCD controller. The IPS BUS is a separate interface which allows software to directly send commands to the NOR flash device by way of the FlexSPI register model. This interface is also used for the initialization and configuration of the external serial flash as it can be used to initiate the process of sending commands.

The capabilities of the i.MX FlexSPI memory controller enhance XiP. In the diagram, just to the right of the AHB_CTL block, both transmit (TX) and receive (RX) buffering are shown. This buffering is used for prefetching data when reading the external memory to improve latency and overall compute performance for the XiP operation.

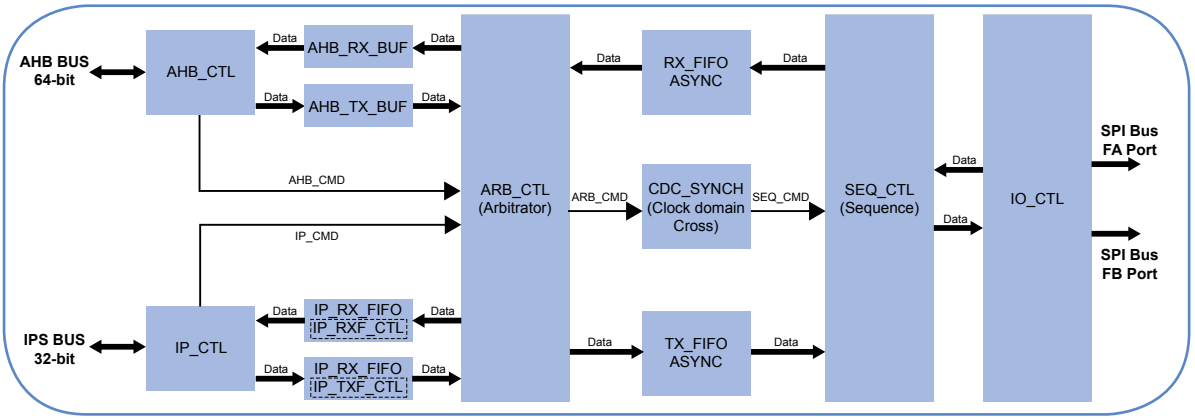

Figure 3: FlexSPI Block Diagram

Shown in the diagram on the right side is the sequence control block. The sequence control block is a large look-up table which holds preset instructions for different serial flash operations such as read, erase and program. This block is what links accesses from the 64-bit AHB bus to the read command sequence which is sent to the external serial flash. Not every flash will have the same command set or I/O interface. The sequence control engine is programmable for adjusting the SPI transfers based on the command set defined by the serial flash. This allows processors like the i.MX RT to interface to a broad range of external flash types and capabilities. This flexibility allows the crossover processor to utilize flash attributes that play an important role in supporting the most capable XiP embedded systems.

ADESTO ECOXIP: ADVANCED SERIAL FLASH

Serial flash is not only for storing code and data but also for executing code directly from flash (Execute-in-Place or XiP). Advancements in serial flash technology have made it possible for newer serial flash to be used in systems with high performance requirements. These advancements allow serial flash devices such as Adesto EcoXiP to respond quickly to read requests from the host MCU and deliver instructions and data with low latency and high throughput.

One advancement is the multi-line SPI interface. Traditionally, communication with a serial device was (as the name suggests) serial. Data would be transferred over a single line at a time. For more capable devices, communication is parallel, and data is transferred over up to eight data lines as shown in the Octal-SPI transfer diagram in Figure 4. Adesto's EcoXiP devices are equipped with JEDEC's latest Octal SPI protocol (xSPI), making the communication close to 8x faster than a single wire serial flash.

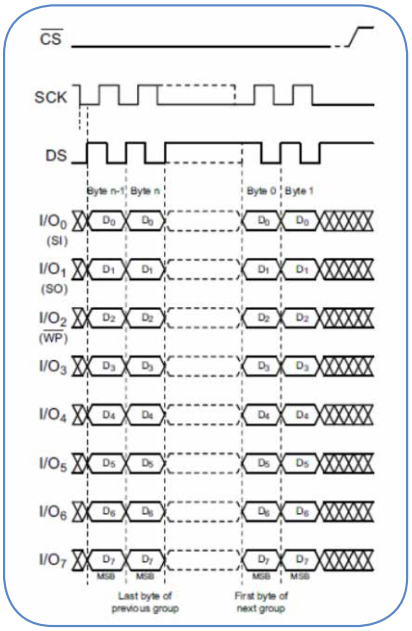

Figure 4: Example Octal-SPI Data Transfer

Supplementing the Octal interface, serial flash can feature double data rate (DDR). This capability is more common in high-speed DRAMs. With DDR, data bits are sampled on both the rising and falling edges of the serial clock. Since it takes only half a clock cycle to send out a data bit, this feature has the potential to double the throughput from the external memory. In addition, modern serial flash devices deliver high clock speeds north of 100MHz. This is achievable due to a data strobe signal driven by the flash during the data phase of a read.

To address latency, Adesto EcoXiP supports features to reduce the overhead of the command interface. Latency is the time from when there is a request for data until the time that the data is available to the requestor. EcoXiP supports special read commands such as Read Array to allow faster access to data by reducing the number of clocks needed for subsequent reads of data. As shown in Figure 5, the Read Array command with Octal SPI and DDR reduces the number of clock cycles needed for passing the command and address data. An 8-bit command and 24-bit address are passed with only 3 clocks. Then subsequent accesses to sequential data are available. All of these serial flash features (read array command, DDR, fast clock speeds and Octal SPI) work to support the XiP use case.

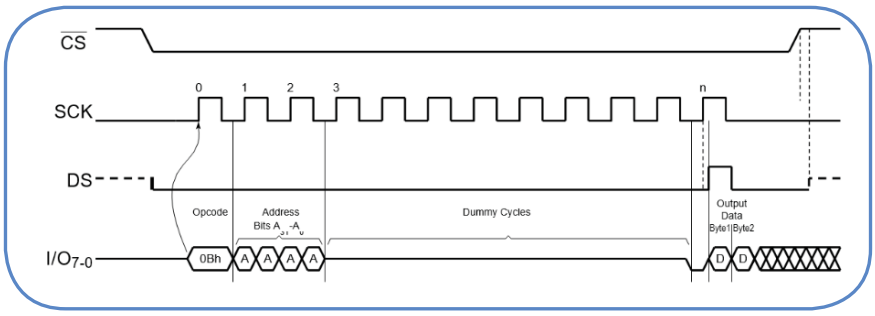

Figure 5: Read Array Command

Application use cases

Beyond addressing the performance of eXecute in Place operation, there are other unique features in EcoXiP to support application use cases. EcoXiP's concurrent read-write, also known as read-while-write or RWW, allows the host processor to continue reading from a partition of the flash memory array while modifying data on another part. As an example, periodic logging of data which involves erase and program operations to the serial flash does not put the XiP program on hold. With the RWW feature, instruction and data fetching during programming continues as usual in a different partition of the flash. This scheme allows read operations from one bank while the device is busy programming or erasing another bank. The serial flash device can be configured into two banks: Bank A and Bank B. The border between the banks can be set with a granularity of 1/8th of the full flash array size. Read commands to one bank can be done while a write is in progress in the other bank.

The XiP architecture also provides advantages to systems which leverage power-down modes to save energy. Unlike execute-from-RAM scenarios, wake up from very low-power modes is much faster. There is no need to copy from a non-volatile memory device into the SRAM execution memory. The system can be set to start executing immediately from external flash. The flash standby power consumption is significantly lower than DRAM systems due to NOR flash memory technology.

In general, the serial flash leakage of Adesto's memory devices is so low that there is no need to turn the flash completely off. Devices like EcoXiP offer deep power-down and ultra-deep power-down modes which result in an extremely low power consumption with only a small impact to wake up time. As shown in Table 1, there are power modes as low as 200 nanoAmps. The end energy consumption (current over time) is significantly lower than what would be required to copy the code into RAM for DRAM based architectures which may require self-refresh.

Parameter	EcoXiP Specifications
Densities	32 Mbit (4 MByte), 64 Mbit (8 MByte), and 128Mb (16 MByte)
Interface	Quad/Octal, SDR/DDR
Read Bandwidth (max)	133 MBs
Power Supply	1.7V – 1.95V
Max. Operating Frequency	133 MHz
Temperature Range (Ta)	-40 °C - 85 °C
Temperature Range (Tj)	-40 °C - 105 °C
Supply Current (Ultra Deep Power Down)	200 nA
Supply Current (Deep Power Down)	4 µA
Supply Current (Standby)	35 µA
1.8V Supply Current – Octal DDR	35 mA
1.8V Supply Current (Program/Erase)	15 mA

Table 1: Adesto EcoXiP Specifications

When not in power-down mode, EcoXiP offers competitive power consumption for active mode while reading from memory and sending data to the host processor. The savings can be as much as half compared to similar Octal SPI devices in the market. For 133MHz Octal SPI reads, the Adesto EcoXiP read current is typically 35mA.

Flash devices offer security features as well. For example, EcoXiP contains a specialized OTP (One-Time Programmable) security register that can be used for purposes such a unique device serialization, system-level Electronic Serial Number (ESN) storage, locked key storage, etc. This register can be programmed but not erased, so only a one-direction transition is possible for each bit. In addition, this register can be permanently locked.

Flash devices are supported by the embedded development ecosystem in different ways. EcoXiP provides flash-loader plug-ins for various embedded tool chains. The flash loader is engaged by the integrated development environment once it detects that a program's binary image, or part of it, falls into the flash memory address range. It will initialize the flash and erase and program memory regions on-demand as requested by the host tool. In this context, it's worth mentioning a new feature called Serial Flash Discoverable Parameter (SFDP) which provides useful information about the flash in a standardized way. This allows the host to automatically figure out flash attributes and set it up the interface accordingly. In theory, one could develop a universal flash loader which would work on all serial flash devices. An update of SFDP to support the new Octal-SPI (xSPI) standard has been recently ratified by a JEDEC committee JC42.

I.MX RT: ADVANCED PROCESSOR ARCHITECTURE

Contributing to the support of the external serial flash in embedded systems are the advanced processor architectures which are now available. For example, the i.MX RT crossover processor is built with the highest-performance Arm® Cortex-M® processor, the Arm Cortex-M7. This CPU can execute up to two instructions every clock cycle and supports 6-stage pipelining, improving computational ability versus other CPUs in its class. The high-performance CPU ensures that even though slower memory accesses may stall the CPU, the high compute power is delivered when data is made available. In addition to the CPU, the internal bus system associated with this class of processor is the same as what has previously been used for higher-end controllers built with Arm Cortex-A family of devices.

The diagram in Figure 6 represents the architectural details of the i.MX RT 1050 crossover processor. With regards to cache, the i.MX RT integrates 32KB for the instruction and 32KB for the data caches. This is the largest size in the market and reduces the CPUs sensitivity to any delays imposed by slower memories. For the Tightly Coupled Memory (TCM), the i.MX RT has a FlexRAM block of memory. This intelligent RAM memory controller allows customization of the TCM up to the largest sizes available on the chip. The user can select the maximum size, or repurpose the FlexRAM to work as on-chip SRAM to be shared with other chip peripherals. Having a large TCM allows software architects to choose this memory option for the portions of their code which need the absolute maximum performance. Software placed in the TCM will achieve the lowest latency access times, producing the highest performance.

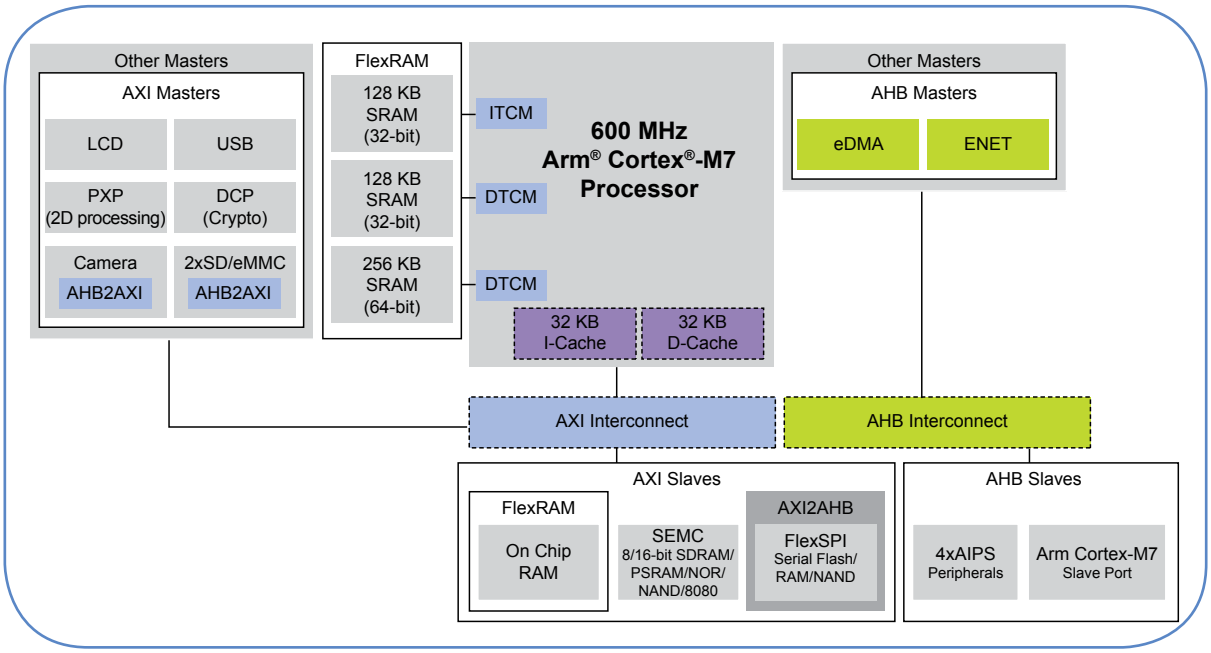

Figure 6: i.MX RT Architecture Diagram

With regards to the use of the 64-bit AXI on the i.MX RT, there are a broad range of AXI masters which are integrated onto the chip. The AXI bus is a split-transaction protocol and supports multiple outstanding transfers. Some specific peripherals to highlight which are relevant to emerging application trends are the camera interface and cryptographic accelerator (Data Co-Processor-DCP). These components differentiate the i.MX RT in the market and align with the need for image processing capabilities and security. The FlexSPI controller allows for these other masters to make use of the receive buffer. This allows the data stored in the external flash to be quickly accessed as with the case of displaying graphics on a screen.

Finally, most relevant to the computational capabilities of the i.MX RT with external flash is the processor speed. Reaching 600MHz allows the i.MX RT to be throttled up for the most intensive calculations. Once data is available to the processor, it is processed at the CPU speed. With all of these capabilities working together, the end result is a processor using XiP that can achieve high performance and is expandable to a nearly limitless memory footprint. Figure 7 details how the CPU and the FlexSPI work together to reduce stalling the flow of application code. Starting at stage 0, the figure represents the case of a full miss of the target data and subsequent prefetching done by the FlexSPI. The stages show how the levels of cache and buffers have to be missed to stall the CPU.

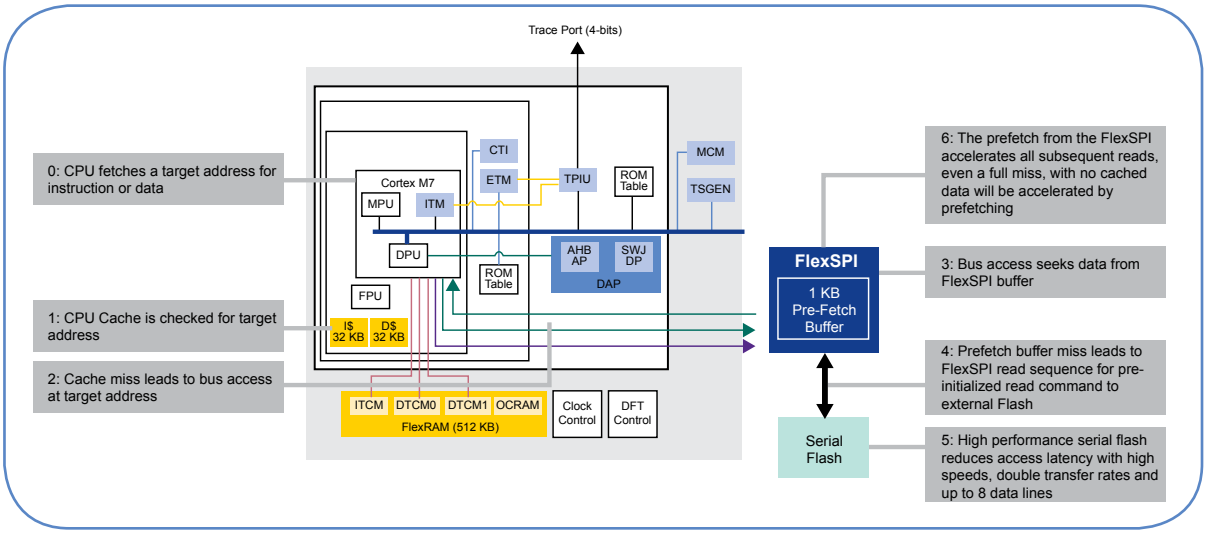

Figure 7: XIP Memory Access Stages

UNDERSTANDING XIP PERFORMANCE

As detailed in the previous sections, the technology associated with the processor and the external NOR Flash memory is built to obscure the latencies involved with using XiP. This presents a challenge with regards to fully understanding the performance impact for this architecture. For example, with the FlexSPI receive buffer, each read access made to the serial flash can range from one cache line (32 bytes for the Arm Cortex-M7) up to the full size of the receive buffer. The receive buffer is 1KB for the i.MX RT1050 processor. The maximum size of the read transaction to completely fill the receive buffer is preset as part of the configuration of the FlexSPI.

Due to the receive buffer, smaller code loops, such as an iterative mathematical calculation, or a case statement, after a few cache lines are pulled from external memory, the processor no longer depends on additional data. At this point, the processor will be executing from buffered data. The receive data continues to be drawn from the serial flash to fill the buffer size that has been preset. Because of this, traditional methods of monitoring memory accesses as an indication of performance do not apply. High performance is achieved even with high access rates to the external memory. Performance cannot be directly correlated to the amount of external memory accesses made by the system.

In addition, many standard industry benchmarks are relatively small programs. These programs often fit in the caches integrated on the processor. As such, they don't represent full scale applications which push memory size boundaries. Thus, in order to understand the expected performance levels for XiP, various methods have to be applied. These are divided into the following three cases: throttling, instrumenting and evaluating example application code.

Throttling Test Case

The throttling test case simulates a scenario where a change in program execution would result in processor accesses which are all outside the CPU cached data. For throttling test cases, the industry standard benchmark EEMBC CoreMark® is used. This benchmark is first placed in zero latency TCM to produce the ideal case CoreMark score. This is the control measurement. Then, the benchmark is run in external serial flash while periodically invalidating the instruction cache at set intervals. This method has the advantage of relating to a standard benchmark (CoreMark). The generated results can be compared to many number of publicly posted results that are hosted by EEMBC.

The drawbacks to this method are that for typical application code, such drastic changes to program flow would rarely lead to a scenario where all of the CPU instruction cache would be invalidated. Estimating the rate at which the cache should be invalidated is challenging. Regardless of these limitations, this test case provides insight into how the technology enables high performance with XIP. The results show that with feature rich serial NOR flash devices such as the Adesto EcoXiP set for Octal SPI and double data rate, performance is only slightly affected by the CPU cache invalidation events.

Figure 8 shows measurements taken with various cache invalidation rates (1ms, 500us, 250us and 125us). There are two different serial flash conditions: the orange line represents a a single data rate, 4 I/O serial flash, and the blue line represents the Adesto EcoXiP set for Octal SPI and DDR. The chart shows the performance advantage of high performance serial flash like Adesto EcoXiP versus slower, lower pin count flash. Considering the 1ms invalidation rate, there is just over a 3% impact to the CoreMark benchmark. The 1ms condition is a relevant test case as the typical RTOS tick rate is set to 1ms. Even lower-performing serial flash devices represented by the orange line have a minimal impact at this rate, delivering 88% of the CoreMark score versus the ideal case. When considering more extreme cases where CPU cache invalidation occurs 8 thousand times per second for example, the higher-performance technology delivers nearly 83% of the performance compared to the ideal case.

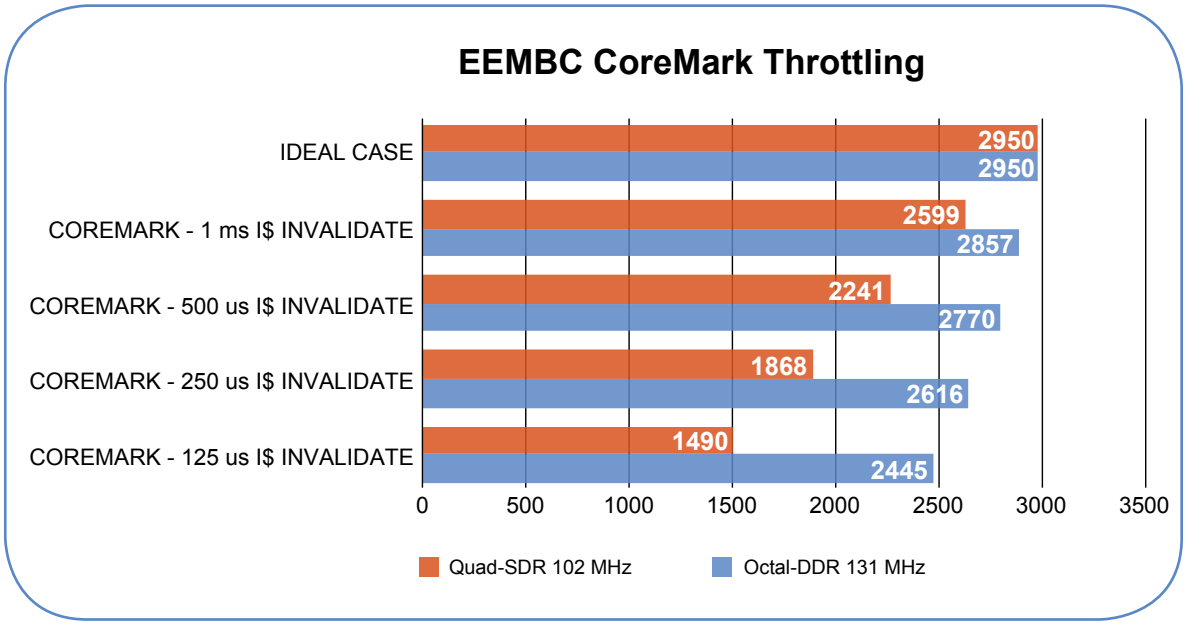

Figure 8: Throttling CPU Cache Results

For the case of invalidating the CPU cache every 125 microseconds, the end result still achieves a 2,445 CoreMark score. This is significantly higher than many other processors in the market.

Instrumenting Test Case

In order to evaluate performance without using a drastic cache invalidation, code can be instrumented in a way to allow cache misses to occur more naturally. For the instrumenting test case, a large block of code is placed in sequential address space which is larger than the size of the CPU cache. So when there is a cache miss, it is due to a more natural software execution scenario. This method involves creating a number of smaller loops which can be set to execute a variable number of times (n). These smaller loops are concatenated together to create a sequential code block that is larger than the CPU cache. When the smaller loops are executed more frequently, by setting larger values of n, there are more cache hits. When the smaller loops are executed less frequently, then there are more cache misses.

Figure 9 is a graphical representation of this method. For the purpose of creating a measurement to evaluate, Fibonacci calculations were used. As shown in the diagram, the processing of each block always requires one pass of the Fibonacci calculation loop leading to cache misses for that pass. When the CPU first reaches a Fibonacci block, the first iteration will be cache misses, but all subsequent passes will be executed from cached data. For the case of n = 10, the first Fibonacci calculation is a miss and the subsequent 9 Fibonacci calculations are cache hits. For the case of n = 30, the first Fibonacci calculation is a miss and the subsequent 29 Fibonacci calculations are cache hits.

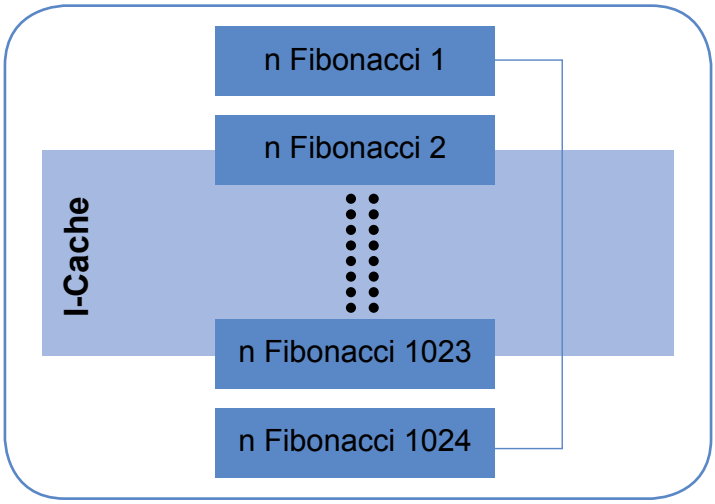

Figure 9: Instrumented Code

Measurements were taken for 10, 20 and 30 iterations of the Fibonacci calculations. Measurements of the total number of Fibonacci calculations are taken with different memory space location and different types of serial flash. Higher performance is represented by a higher number of Fibonacci calculations. As shown in Figure 10, at 30 iterations, the impact to the number of Fibonacci calculations is just over 15% reduction.

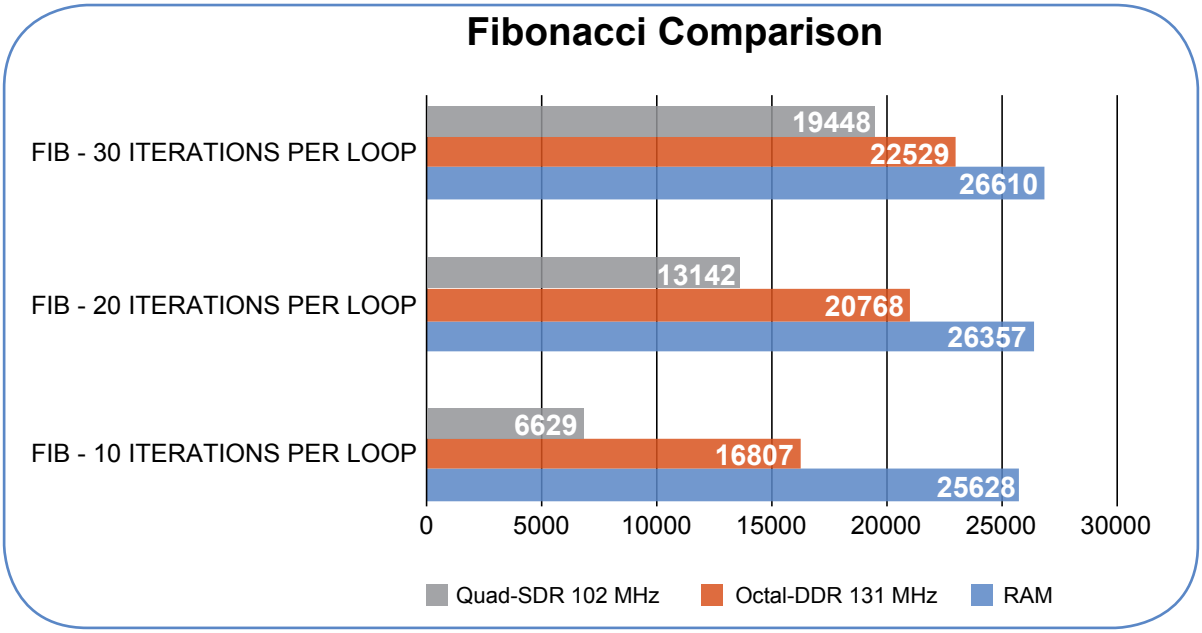

Figure 10: Results of Instrumented Code

As the cache miss rate is increased, the data shows that having high-performance serial flash leads to less impact than using standard serial flash. This is comparing the orange bar to the gray bar results. Though this method allows precise control over the cache miss rate, it does not fully represent standard application code. The cache miss rate on standard application code can vary broadly depending on use case.

Examining example applications

To overcome the limitations of instrumented code and throttling cache, running an example application in different target memory scenarios offers additional proof points to the performance when using XiP. This test scenario is easily accomplished because XiP is enabled through the MCUXpresso Integrated Development Environment. (IDE). The MCUXpresso IDE projects can be created to place software into the TCM zero-latency memory. After performing measurements, the same software can be applied to the external serial NOR flash space and measured again. There are many example projects to choose from in the software

development kits (SDKs) offered by NXP. The entire process with the measured results is detailed in a step-by-step lab guide (see link provided in the resources section). This guide allows developers the opportunity to explore these methods themselves. The examinations can be done with the provided SDK application examples or with the final application software created by the developer.

For the case demonstrated by the lab guide, Arm Mbed TLS benchmarking of Elliptical Curve Digital Signature Algorithm (ECDSA) was performed. The results show that with CPU cache enabled, for this specific benchmark the measured difference between ITCM and using external flash does not change. Whether executing from the best case memory, the TCM, or executing from external serial flash with XiP, the ECDSA benchmark application shows the same results.

For a different case, when using MCUXpresso compiler optimizations set for performance, the measured difference for ECDSA computations is shown to be less than 6% lower for the XIP case. Changing compiler settings changes the generated machine code so that it is much more compact. The end result is approximately a 4x improvement for the ECDSA calculations. As the code becomes more optimized, the throughput provided by the external serial flash begins to affect the measured performance, leading to the slight impact when using XiP.

DEVELOPMENT AND DEBUG WITH XIP

As demonstrated by the lab guide, other experiments for XiP can be performed with the enablement provided by the MCUXpresso. For example, the speed of the external memory can be varied by changing definitions inside the project. The MCUXpresso platform provides the tools needed to quickly examine this and other scenarios, allowing the developer to fully leverage the benefits of the expandable XiP architecture. For downloading and debugging application software, the MCUXpresso IDE is preset to allow a seamless connection to the serial flash components placed on the i.MX RT Evaluation Kit. (EVK). When a debug session is initiated by the user, the flash loader scripts are automatically used by the debug tool. In addition to the development tools, the off-the-shelf configuration of the i.MX RT EVK hardware has both a high-performance 8-wire SPI as well as a 4-wire SPI. With both of these serial flash options placed on the board, the user can choose the right attributes for their end design.

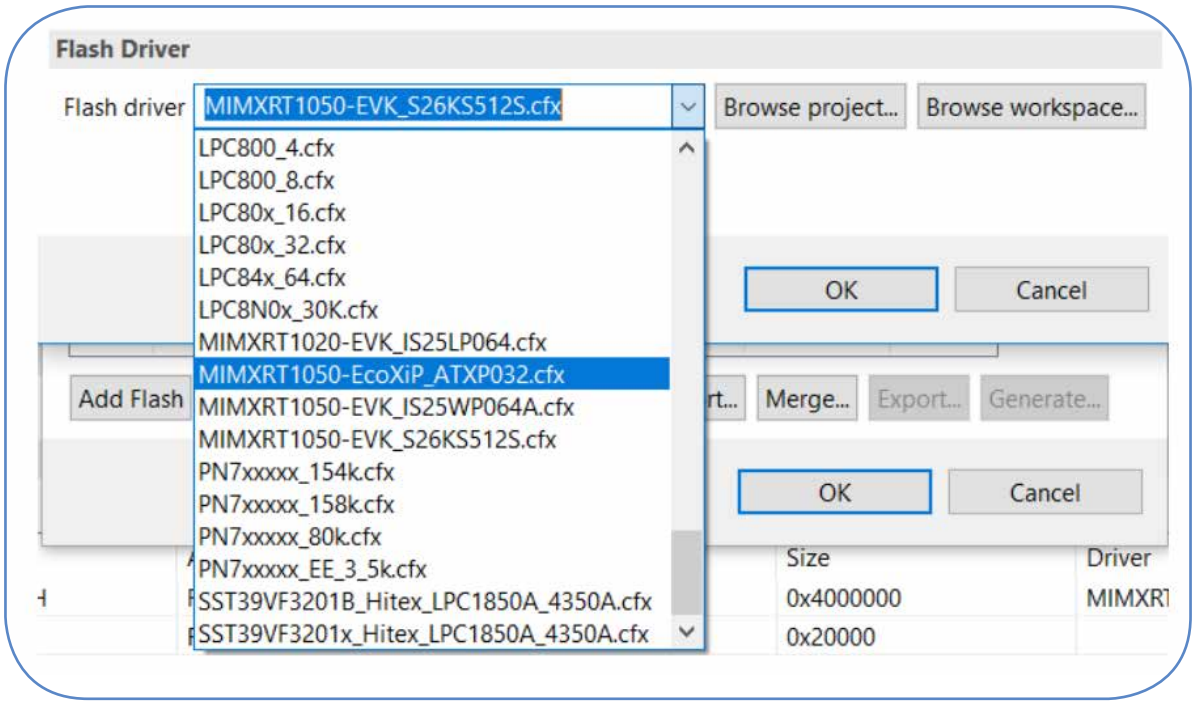

Figure 11: Selecting Adesto Serial Flash

When importing SDK projects into MCUXpresso, the choice of the serial flash hardware is made based on the memory settings in the memory configuration editor. The lab guide provides the detailed steps to choose the Adesto flash during the import as highlighted in Figure 11. With a special edition of the i.MX RT EVK that has the Adesto EcoXiP placed on the board, nearly all of the SDK examples can be run and debug with the Adesto external flash. The operation of the enablement tools with the crossover processor is just as it would be for a traditional microcontroller which contains embedded flash.

CONCLUSIONS

External memory for an embedded processor offers a scalable platform aligning to the challenges of today's embedded systems. When using external serial flash memory, success can be achieved with the right processor and memory technology. Modern Arm CPUs integrate cache that greatly enhances the use of external memory. In addition, processor designs are architected to use execute in place with memory controllers, such as the FlexSPI memory controller which provides buffering and prefetch. Coupling this with the enhanced capabilities offered by serial NOR flash addresses cost, power, performance and security challenges. Furthermore, the infrastructure provided by tools such as MCUXpresso allows developers the ability to get from concept to deployment quickly and efficiently.

RESOURCES

The following table includes links to resources which support developer investigation into using XIP.

Resource	Description
Processor summary page	The i.MX RT1050 family summary page provides links to chip documents (Data Sheet and Reference Manual)
Hardware evaluation kit	The i.MX RT EVK provides a platform for embedded development. Multiple boot interfaces are supported
Software SDK	The MCUXpresso SDK is the software enablement which provides drivers and middleware for the i.MX RT
Arm Cortex-M7 Whitepaper	Detailed description of the Arm Cortex-M7 CPU
MCUXPresso IDE training	Training material to understand the MCUXPresso Integrated Development Environment features
Using XIP Lab Guide	This is the lab guide mentioned in this paper which provides the detailed steps for experimenting with XIP

CONTRIBUTOR

Wim Rouwet
Systems and Architecture Engineer

HOW TO REACH US:

Home Page: **www.nxp.com**
Web Support: **www.nxp.com/support**

USA/Europe or Locations Not Listed:
NXP Semiconductors USA, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.nxp.com/support

Europe, Middle East, and Africa:
NXP Semiconductors Germany GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.nxp.com/support

Japan:
NXP Japan Ltd.
Yebisu Garden Place Tower 24F,
4-20-3, Ebisu, Shibuya-ku,
Tokyo 150-6024, Japan
0120 950 032 (Domestic Toll Free)
www.nxp.com/jp/support/

Asia/Pacific:
NXP Semiconductors Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@nxp.com

Document Number: NXPADESTOWP REV 0
Release Date: September 2018