iMXRTxxxx Memory Performance: ITCM / DTCM / L1 CACHE / LMEM CACHE / OCRAM / SDRAM / FlexSPI (QSPI / HyperFLASH)

NOTE: Preliminary version

General rules for iMXRTxxxx (4 digits) memory performance:

The RTxxxx (4 digits devices like RT1010/RT1020/RT1050/RT1060/RT1170) are ARM Cortex M7 based flash-less devices. They utilizing on-chip memories like:

- ITCM accessed by its own I-TCM bus interface (single cycle memory dedicated for instruction fetches - **code execution, interrupt vector table**)
- DTCM accessed by its own D0-TCM / D1-TCM interface (single cycle memory dedicated for data access - **stack, important static variables**)
- I-CACHE
  - cache hit: represents a single cycle memory for instruction fetches
  - cache miss: generates 32B burst transfer on AXI bus
- D-CACHE
  - cache hit: represent a single cycle memory for data accesses

    - cache miss: generates 32B burst transfer on AXI bus
- OCRAM accessed by AXI bus via interconnect bus fabric NIC (expected to be cached in order to achieve sufficient performance, multiple wait-states are generated when accessed via AXI - **data accessed by multiple masters like CM7 and DMA, avoid placing a stack here if possible, etc.**)

- both cache memories above are coupled with system AXI bus interface utilized when other on-chip or off-chip memories accessed
    - AXI bus is optimized for burst transfers that is why the CACHE should be utilized when accessing memories via AXI bus, especially for read request
    - read requests:
        - it should be highlighted here that a single read requests generated on AXI bus are not effective - low performance can be expected
            - reason (simply said): AXI read address channel requires handshake (ready / valid) + wait for response in data channel which also includes handshake (ready / valid) + NIC introduce additional latency due to clock re-synchronization etc.
        - higher number of burst length on AXI bus read transaction results in much better data throughput
            - for example: 128B read transaction generated as 128x8-bit requests (128x1B) which results in generation of 128x latency mentioned above
            - for example: 128B read transaction generated as one burst transfer, generates 1x latency. **NOTE**: It must be noted that burst request will generate a bit more latency in compare to single request
    - write requests:
        - write transaction on AXI bus results in higher data throughput
            - reason (simply said): AXI write address channel and write data channels can be handled at the same time, no wait for "data reply" from slave required + NIC allows for buffered write

NOTE: RTxxxx devices utilizes FlexRAM which allows configuration of memory banks within FlexRAM to be access by I-TCM or D-TCM or AXI bus. I suggest to configure most of the FlexRAM to xTCM if highest performance required. The OCRAM is non-deterministic (due to cache) and much lower performance (due to AXI + NIC).

NOTE: Considering AXI protocol + NIC latency it should be expected that data throughput on external memories (QSPI/HyperFLASH/SDRAM) will never reach xTCM performance. The performance of RTxxxx devices is intended to be utilized by using xTCM memories and L1 CACHEs.

Selected memory controllers features

FlexRAM and OCRAM memory controllers:

FlexRAM includes memory controllers which are responsible to convert AXI (OCRAM) or TCM (ITCM, DTCM) interface signals into the RAM array interface signals. There are also multiplexers between each memory controller and each RAM array bank which are responsible for proper connection
of memory controller and its RAM bank based on FlexRAM configuration. These memory controllers also allow to control the access to the memory.

xTCM controller:

- TCM controller converts the TCM (64-bit I-TCM bus or 2x32-bit D-TCM buses) interface signals into RAM array signal. TCM interfaces are synchronous to the Cortex M7 and run at the same frequency. Hence it is expected that the access to the xTCM memories is single cycle. TCM controller can also control the access to the RAM bank and affects memory data access time (fetch the instructions on I-TCM or access data on D-TCM).

OCRAM controller:

- The OCRAM memory acts as a slave module connected to 64-bit system AXI bus. It contains of OCRAM controller which is handling FlexRAM banks according to its configuration. The OCRAM controller converts AXI interface signal to the RAM array signal. The read and write transactions are handled by two independent read and write control modules. The controller also contains arbiter which takes control when two simultaneous requests comes from both read and write modules. Arbiter is working in round-robin scheme. The simultaneous read and write transactions are possible when targeted to different memory banks. When targeting the same bank the read access gets higher priority.
- **NOTE 1-1**: On RT10xx OCRAM bus clock is limited by interconnect bus fabric (NIC) slave port to 1/4 core clock, i.e. RT1050 running 600MHz the OCRAM bus clock is 150MHz. Hence at least 4 wait states expected to get 64-bit data.
- **NOTE 1-2**: On RT117x the OCRAM bus clock is asynchronous to the core and it is limited to 240MHz. Hence at least 4-5 wait state expected to get 64-bit data.

L1 CACHE (CM7)

All RTxxxx (4 digits) devices includes core platform (ARM design) L1 cache memories for caching the code (I-CACHE) and data (D-CACHE). All code/data requests on AXI system bus within cache-able region can be (depends on region configuration within MPU) cached by I-CACHE and D-CACHE controllers. All requests for data located in cache can be considered as single cycle similarly as it is in a case of TCM memory. If request for data generate cache miss on cache side then the cache controller can generate burst transfer on AXI system bus to fill cache line. The cache line size is 32B. See more in [2].

LMEM CACHE (CM4) controllers

Multicore RTxxxx devices with asymmetric core CM4 include local memory controller (LMEM) as known from some Kinetis devices. LMEM controller includes 4 memory controllers:

- ITCM controller: used to handle code/instructions within ITCM memory (also known as SRAM lower (SRAM_L)), mapped into the CODE region: 0x1FFE 0000 - 0x1FFF FFFF and accessed by CODE bus.

- DTCM controller: used to handle data within DTCM memory (also known as SRAM upper (SRAM_U)), mapped into the SRAM region: 0x2000 0000 - 0x2002 0000 and access by SYSTEM bus.
- CODE-CACHE memory controller: used to handle code/instructions within cache-able CODE region e.g. FlexSPI alias on 0x0800 0000 - 0x0FFF FFFF
- DATA-CACHE memory controller: used to handle data within cache-able non-CODE region (SRAM / External memory / External devices) e.g. SDRAM alias on 0x8000 0000 - 0xDFFF FFFF.

Access to any memory within LMEM controller is considered as single cycle (CM4 core). It should be highlighted here that core (CM4) has direct access to LMEM through front-door port. Other bus masters as for example DMA access to LMEM controller through back-door port which basically represents one of the slave port on NIC. It is necessary to consider two arbitration logic for other bus masters: NIC + LMEM. For the core only LMEM arbitration can be considered.

SDRAM controller:

- memory mapped into external memory region (starting from 0x8000 0000 address) defined by ARM, accessed basically by system AXI bus on Cortex M7
    - SDRAM controller is access by AXI 32-bit bus width
- 4 regions dedicated for SDRAM controller based on chip select (CS0, CS1, CS2, CS3) with up to 512Mb in size
- 8/16-bit modes supported
    - **NOTE 2**: on RT11xx also 32-bit mode supported
- AXI command 8/16/32-bits access supported
- all burst lengths for INCR type supported
- except 8/16-bit size all burst size supported for WRAP type
- no pre-fetch / cache support
- **NOTE 3-1**: If any other memory controller within SEMC is accessed during SDRAM access reduces access speed to SDRAM due to arbitration process etc.
- **NOTE 3-2**: SDRAM controller is part of SEMC module which is primarily optimized for burst transfers as dedicated by AXI bus protocol. Hence single access request can introduce additional latency and reduce performance. Especially single read access by core (which is most usual way core requests data/instruction) on AXI system bus generates needs for wait for reply from slave port (SDRAM controller).
    - **Example**: SDRAM used for XIP:
        - I-CACHE disabled: Using SDRAM for XIP where code is optimized for size (utilizing thumb 16-bit instructions) can lead in significant reduce of performance. The core pipeline pre-fetch stage generates a single requests directly to AXI system bus. Each particular single request requires wait for response and generates more latency.

  - I-CACHE enabled: The core pipeline pre-fetch stage generates a single requests to I-CACHE. If cache hit occur it can be considered as single access. If cache miss occur the cache controller generates burst transfer request on AXI system bus. The burst size depends on cache line size which is RTxxxx is 32B.
  - Considering sequential 32B code with 32-bit instructions:
    - case 1: I-CACHE disabled - core pipeline generates 32x single data requests which results in **32x wait responses** on AXI bus
    - case 2: I-CACHE enabled with cache miss at first request - core pipeline generates 32x single data requests. The first request generates cache miss on cache which generates 32B burst transfer on AXI bus. Filling one 32B cache line requires **1x wait for response** from SDRAM slave port on AXI bus. When cache line filled the rest 31x single requests are generated only on I-CACHE considered as single cycle.
    - case 3: I-CACHE enabled with no cache miss generation - core pipeline generates 32x single data requests to I-CACHE considered all as single cycle. **No wait for response** on AXI bus.
  - There are lot of dependencies:
    - code instruction content: optimized for speed represents most of 32-bit thumb instructions, optimized for size represents most of 16-bit thumb instructions which can results in even worse scenario of case 1
    - code flow: branching in the code can result in often cache line re-fill which can degraded performance of case 3

FlexSPI controller:

- memory mapped into external memory (starting from 0x6000 0000 address (RT10xx)) defined by ARM, accessed basically by system AXI bus on Cortex M7
  - FlexSPI controller is access by AHB 64-bit bus width
  - **NOTE 4**: on RT11xx the default FlexSPI1 is mapped into the 0x3000 0000 address space, FlexSPI2 is compatible with RT10xx devices and mapped into the 0x6000 0000 address.
- flash access modes:
  - Single/Dual/Quad/Octal mode
  - SDR/DDR mode
  - Individual/Parallel mode
- 128 x 64-bits AHB RX buffer - up to **1kB** prefetch / cache support which can significantly help when continuous address spaces is accessed (linear code, data buffer)
  - **NOTE 5**: on RT11xx **4kB** pre-fetch is implemented
- 8 x 64-bits AHB TX buffer

Simplified system platform

iMXRT10xx based platform

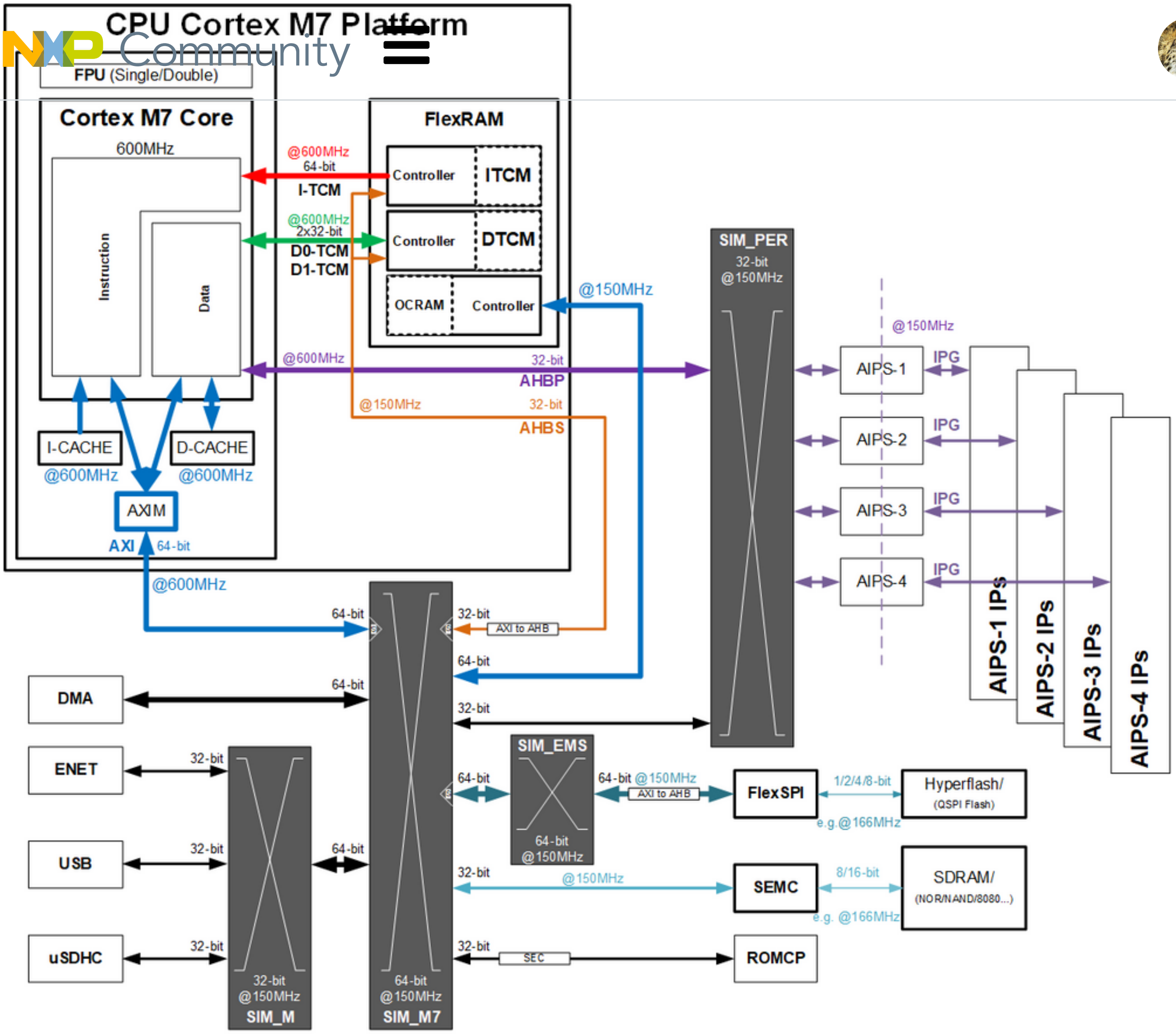


Fig. 1 RT10xx - Simplified Bus Diagram

iMXRT11xx platform

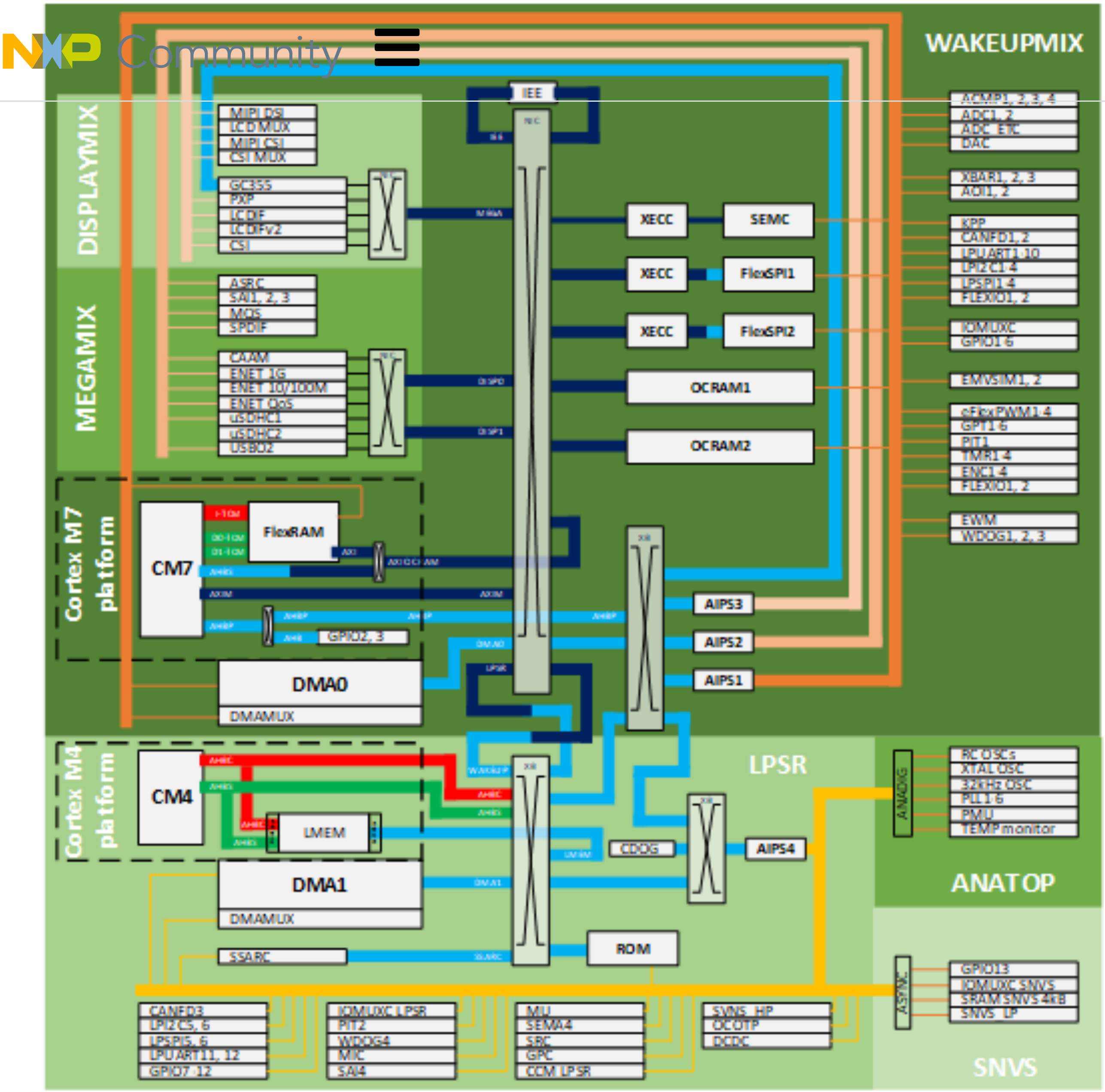
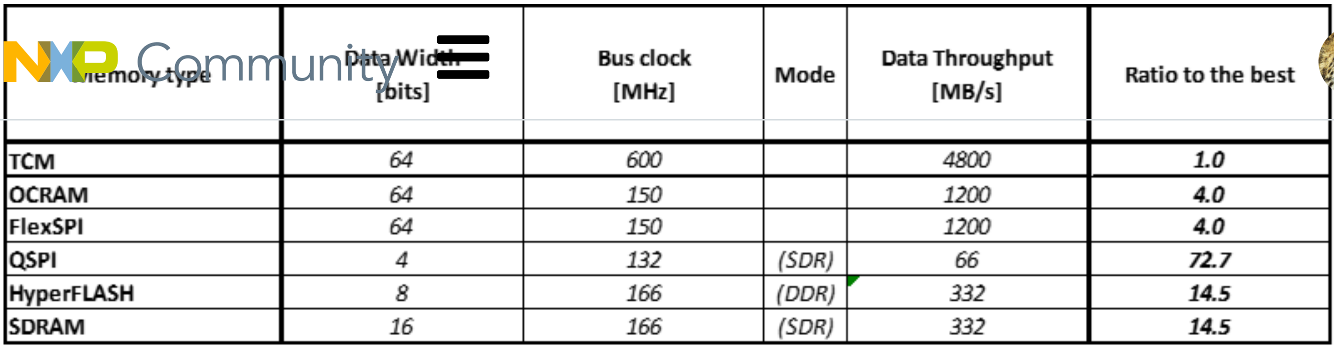


Fig. 2 RT117x0 - Simplified Bus Diagram

Memory benchmark

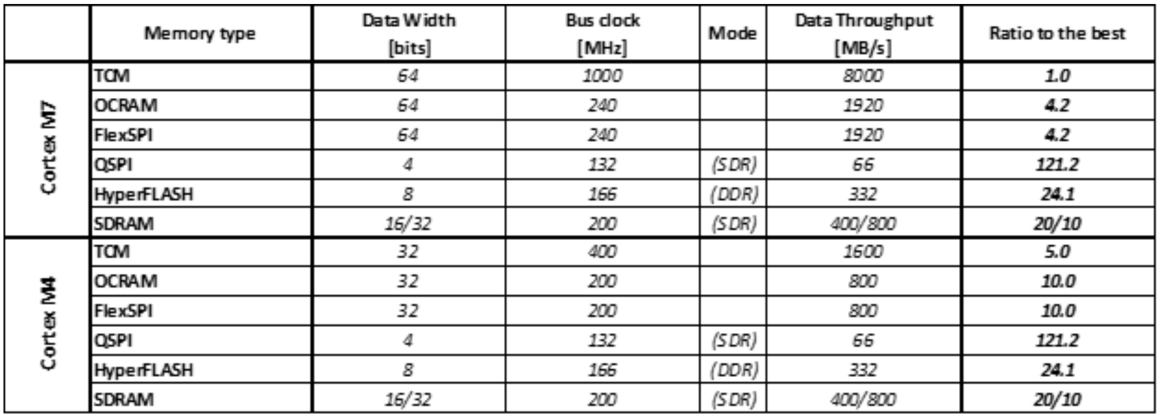
The following section deal with corresponding memory performance. It would practically refer to most correct data throughput for corresponding memory. It does not consider theoretical data throughput calculated as data bus width multiplied by bus clock frequency as marketing usually says but it consider real data bus master (in our case ARM Cortex M7 core) can utilizes - excluding non useful data such as command/address/modes/dummy especially in a case of external serial / parallel memories.

Here is example of theoretical data throughput on selected memories and access time to them:



Memory type	Data Width [bits]	Bus clock [MHz]	Mode	Data Throughput [MB/s]	Ratio to the best
TCM	64	600		4800	**1.0**
OCRAM	64	150		1200	**4.0**
FlexSPI	64	150		1200	**4.0**
QSPI	4	132	(SDR)	66	**72.7**
HyperFLASH	8	166	(DDR)	332	**14.5**
SDRAM	16	166	(SDR)	332	**14.5**

Tab. 1-1 Example of theoretical data throughput RT105x/RT106x



	Memory type	Data Width [bits]	Bus clock [MHz]	Mode	Data Throughput [MB/s]	Ratio to the best
Cortex M7	TCM	64	1000		8000	**1.0**
	OCRAM	64	240		1920	**4.2**
	FlexSPI	64	240		1920	**4.2**
	QSPI	4	132	(SDR)	66	**121.2**
	HyperFLASH	8	166	(DDR)	332	**24.1**
	SDRAM	16/32	200	(SDR)	400/800	**20/10**
Cortex M4	TCM	32	400		1600	**5.0**
	OCRAM	32	200		800	**10.0**
	FlexSPI	32	200		800	**10.0**
	QSPI	4	132	(SDR)	66	**121.2**
	HyperFLASH	8	166	(DDR)	332	**24.1**
	SDRAM	16/32	200	(SDR)	400/800	**20/10**

Tab. 1-2 Example of theoretical data throughput RT117x

Memory access approaches

The code which has been use for testing memory data throughput was pretty simple and divided into three different approaches:

- a simple data read on continuous address space - the next accessed data were placed on next address
- a simple data read of non-continuous address space (ping-pong fashion) - the next addressed data were placed on address 16kB far from previously accessed data (to highlight degradation of pre-fetching data)
- a read data from one address space and then write this data to different address space (16kB address distance between both address spaces)

There was two 16kB buffers withing each memory, it means:

- ITCM (CM7):
  - itcmDataBuffer0 @0x00018000
  - itcmDataBuffer1 @0x0001C000
- ITCM (CM4 as SRAM_L):
  - sramlDataBuffer0 @0x1FFF8000
  - sramlDataBuffer1 @0x1FFFC000
- DTCM (CM7, CM4 as SRAM_U):
  - dtcm_data_buffer0 (sramuDataBuffer0) 0x20018000

  - dtcm_data_buffer1 (sramuDataBuffer0) 0x2001C000
- OCRAM:
  - ocramDataBuffer0 @0x202B0000
  - ocramDataBuffer1 @0x202B4000
- SDRAM:
  - sdramDataBuffer0 @0x80040000
  - sdramDataBuffer1 @0x80044000
- FlexSPI (used only for reading from):
  - flexSpiDataBuffer0 @0x30400000
  - flexSpiDataBuffer1 @0x30404000

The buffers were in following meaning:

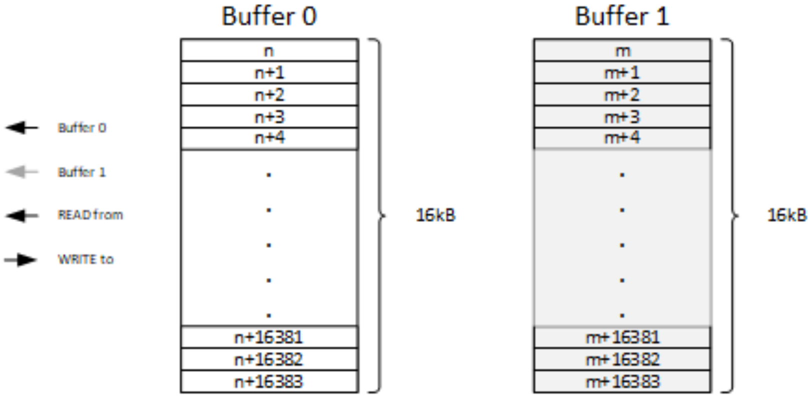


Fig. 3 Memory benchmark buffers

NOTE: It is very important to mention that during benchmark code execution the data CACHEs (CM7 L1 D-CACHE as well as CM4 SYSTEM LMEM CACHE) has been utilized the way that the access to the same data (assumed to be cached) has not appeared. It means the data cache has been invalidated after each data buffers transfer. So, the advantage here is only in generation of burst transfers by CACHE controller. No single cycle access to data considered here.

Simple data READ or WRITE approach

In a case of simple data read/write the address for reading/writing data was incremented by 1 in dependence of access size:

- incremented by 1B when 8-bit access: in code represented by LDRB / STRB instruction, e.g. LDRB.W R3,[R1],#1 / / STRB.W R1,[R0],#1
- incremented by 2B when 16-bit: in code represented by LDRB / STRH instruction, e.g. LDRH.W R3,[R1],#2 / / STRH.W R1,[R0],#2
- incremented by 4B when 32-bit: in code represented by LDR / STR instruction, e.g. LDR.W R3,[R1],#4 / STR.W R1,[R0],#4

- incremented by 32B when BURST 8: in code represented by LDM/STM instruction, e.g. LDM R1,{R4-R11} / STM R0,{R4-R11}

The data was read in continuous address space (Buffer 0) until 16kB data size has been reached.

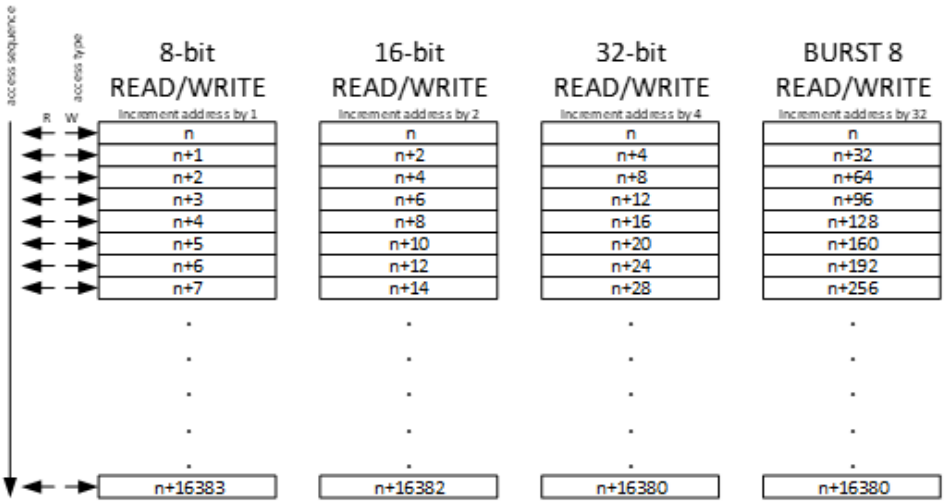


Fig. 4 Buffer READ or WRITE approach with different access sizes

NOTE 6: The above approach consider data access by core (Cortex M4/M7). The data access size is dedicated by the load/store instruction as can be seen in bullets above. The physical memory (cache-able memories) access size can be affected by L2 CACHE memory. L2 cache can translate core data size request into the BURST size data request on system bus. The data size request is then dependent on CACHE line size.

NOTE 7: The different scenario happened when memory is used for code execution (or XIP). Then the data size request is defined by core pipeline fetch stage size (32-bit on Cortex M4 or 64-bit on Cortex M7). The special case is Cortex M7 branch predictor which includes branch target buffer working similar way as cache memory.

Ping-pong READ approach

In ping-pong read approach each data access has been done in interleaving fashion. The even accesses ((8-bit / 16-bit / 32-bit) were done on buffer 0 and the odd accesses were done on buffer 1.

- incremented by 1B when 8-bit access: in code represented by LDRB instruction, e.g. :
    - LDRB.W r3,[r0],#1 (R0 holds the address of buffer 0)
    - LDRB.W r3,[r1],#1 (R1 holds the address of buffer 1)
- incremented by 2B when 16-bit: in code represented by LDRB instruction, e.g. LDRH.W R3,[R1],#2
    - LDRH.W r3,[r0],#1 (R0 holds the address of buffer 0)
    - LDRH.W r3,[r1],#1 (R1 holds the address of buffer 1)
- incremented by 4B when 32-bit: in code represented by LDR instruction, e.g. LDR.W R3,[R1],#4

- LDR.W r3,[r0],#1 (R0 holds the address of buffer 0)
- LDR.W r3,[r1],#1 (R1 holds the address of buffer 1)

It continued until 16kB size has been reached. It means 8kB from buffer 0 and 8kB from buffer 1.

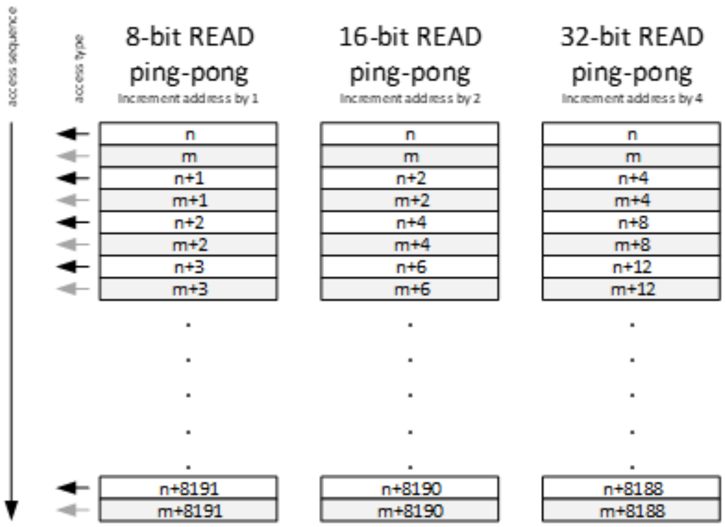
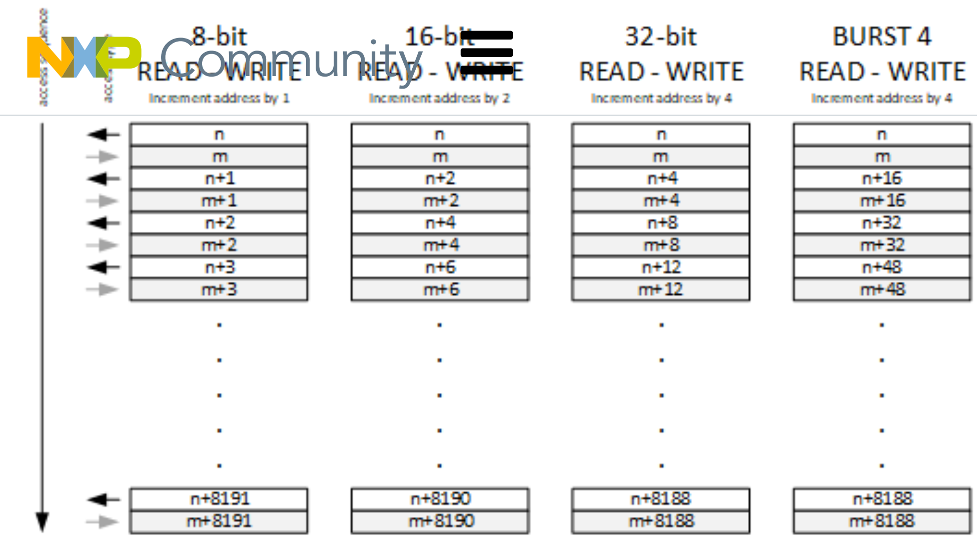


Fig. 5 Ping-pong READ approach with different access sizes

READ then WRITE (copy) approach

In read then write approach the data from buffer 0 has been read and then written into buffer 1, all done within the same memory type until 16kB data size reached. It means read 8kB data from buffer 0 and write 8kB data into buffer 1. This is a typical calling of memcpy function in C and also emulates what DMA machine is doing. In DMA you need to define source (buffer 0) and the destination (buffer 1) to allow DMA controller to copy data from source location into the destination location. When considering Cortex M4/M7 core then the size of data requested on bus is defined by the load/store instruction (see approaches above). When considering DMA bus master then the size can be defined by user e.g. SSIZE or DSIZE fields in TCDx_ATTR registers.

NOTE: In a case of FlexSPI memory data has been read from FlexSPI dedicated memory and write into the DTCM. This must be considered when making conclusion from benchmark results below.



8-bit READ - WRITE	16-bit READ - WRITE	32-bit READ - WRITE	BURST 4 READ - WRITE
n	n	n	n
m	m	m	m
n+1	n+2	n+4	n+16
m+1	m+2	m+4	m+16
n+2	n+4	n+8	n+32
m+2	m+4	m+8	m+32
n+3	n+6	n+12	n+48
m+3	m+6	m+12	m+48
.	.	.	.
n+8191	n+8190	n+8188	n+8188
m+8191	m+8190	m+8188	m+8188

Fig. 6 READ then WRITE approach

Memory benchmark results

Memory benchmark results show data throughput in MB/s for each memory type and each memory access size.

xTCM memory represents single cycle memory that is why these memory has not other options. NOTE: Access mode for xTCM memory controller has not been changed and it was configured as default for fast access.

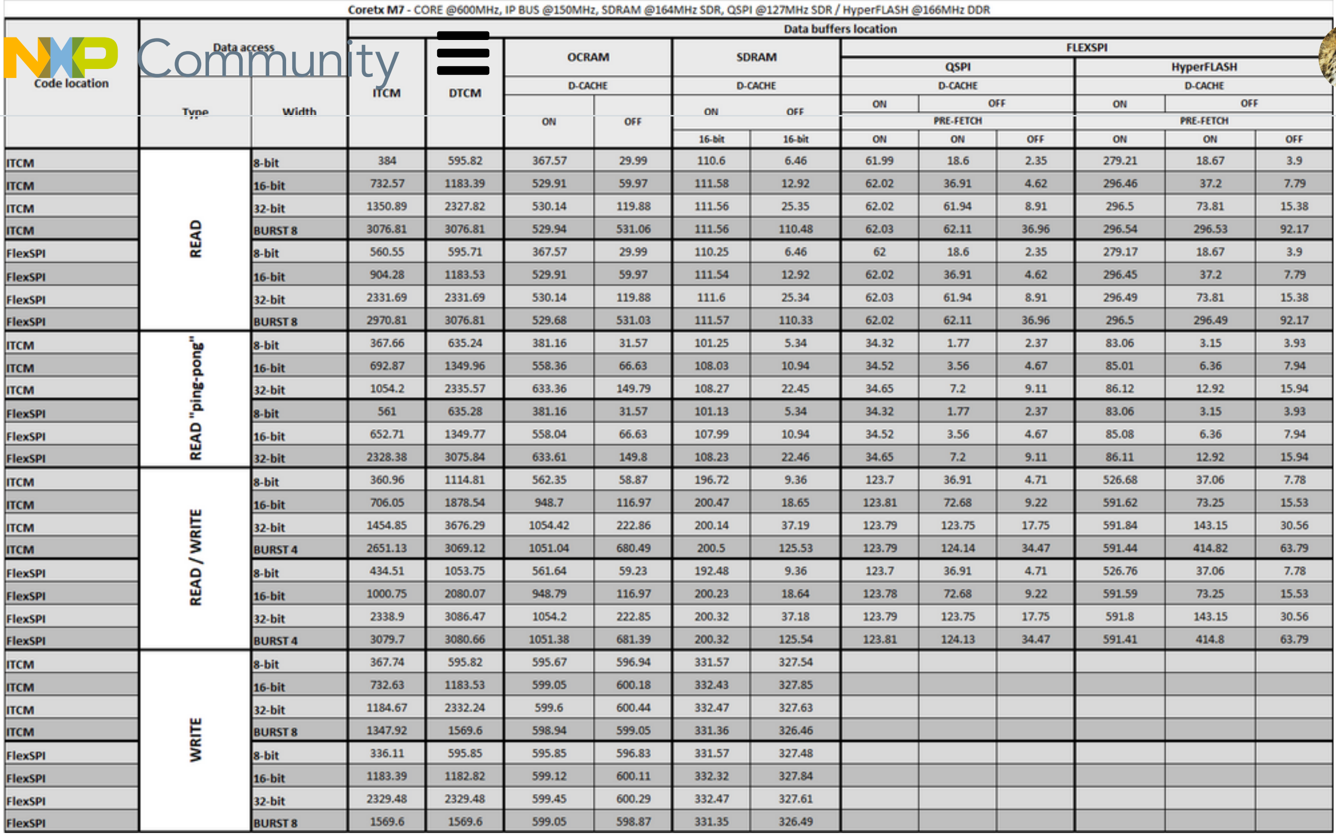
OCRAM memory is connected as slave to interconnect bus fabric (NIC) accessible by 64-bit AXI system bus which accesses can be cachable. This can potentially generates wait-states when accessing this memory. The results in below tables are divided into two columns which represents D-CACHE enabled and disabled.

SDRAM memory is also connected as a slave port (32-bit bus width) on NIC therefore results are divided for D-CACHE enabled and disabled.

FlexSPI dedicated memory is similarly connected as slave port (64-bit bus width) on NIC the results are represented in D-CACHE enabled or disabled. The key feature on FlexSPI controller especially for read access is pre-fetch. Hence also pre-fetch buffer influence is presented in tables.

NOTE: In benchmarks below all results under D-CACHE / SYSTEM-CACHE ON does not represents the cache performance. It represent enabled cache during transfer but after each transfer the cache has been invalidated. At the end it means every CACHE miss represents BURST type request on system AXI/AHB bus. Every CACHE hit assumed as single cycle access to CACHE.

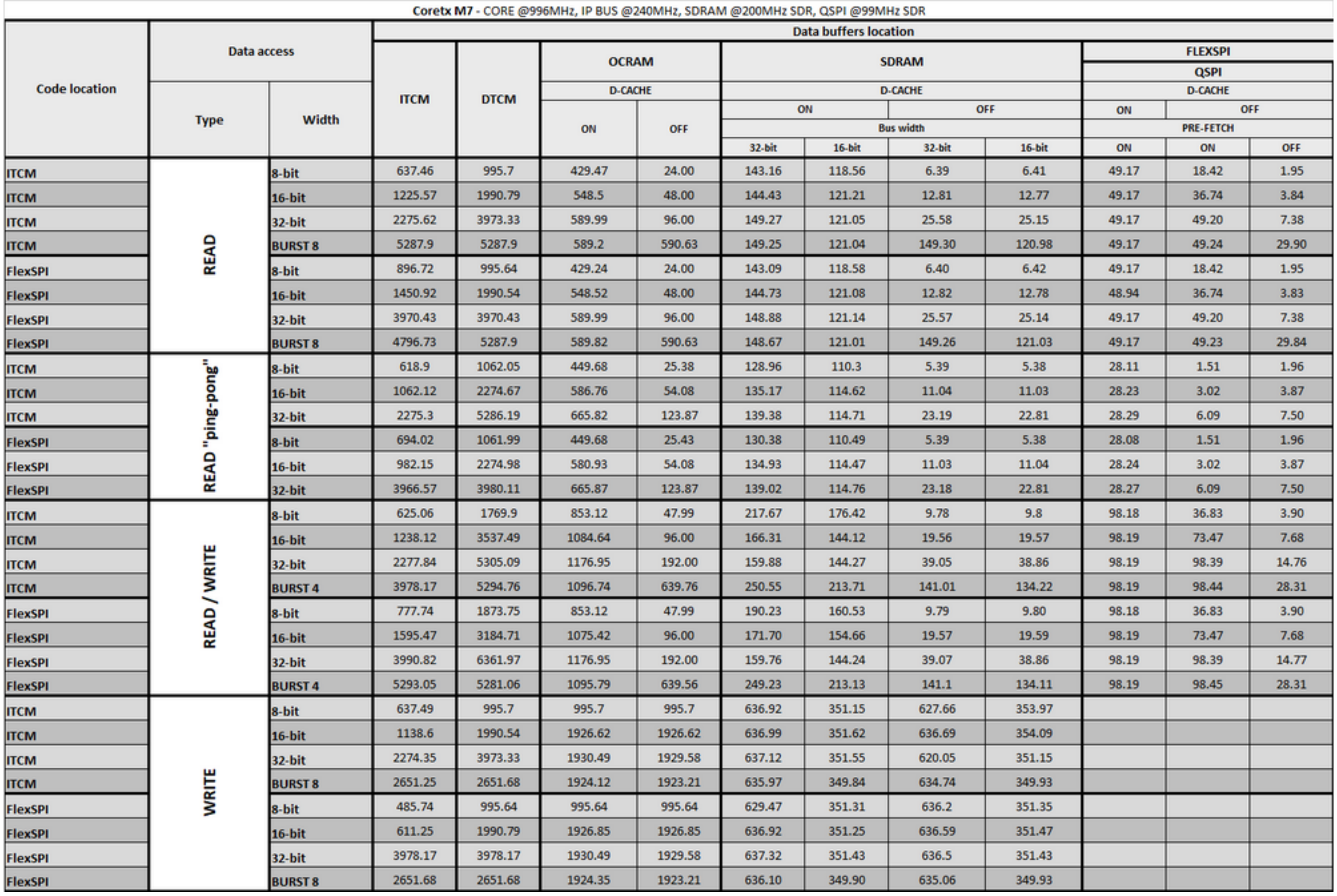
NOTE: **RT11x CM7**: Based on above note statement and considering L1 D-CACHE line size as 32B it is clear why the results for BURST 8 with CACHE OFF or ON are giving very similar results.



Coretx M7 - CORE @600MHz, IP BUS @150MHz, SDRAM @164MHz SDR, QSPI @127MHz SDR / HyperFLASH @166MHz DDR

Code location	Data access		Data buffers location											
			ITCM	DTCM	OCRAM		SDRAM		FLEXSPI					
									QSPI			HyperFLASH		
					D-CACHE		D-CACHE		D-CACHE			D-CACHE		
	Type	Width			ON	OFF	ON	OFF	ON	OFF		ON	OFF	
										PRE-FETCH			PRE-FETCH	
							16-bit	16-bit	ON	ON	OFF	ON	ON	OFF
ITCM	READ	8-bit	384	595.82	367.57	29.99	110.6	6.46	61.99	18.6	2.35	279.21	18.67	3.9
ITCM		16-bit	732.57	1183.39	529.91	59.97	111.58	12.92	62.02	36.91	4.62	296.46	37.2	7.79
ITCM		32-bit	1350.89	2327.82	530.14	119.88	111.56	25.35	62.02	61.94	8.91	296.5	73.81	15.38
ITCM		BURST 8	3076.81	3076.81	529.94	531.06	111.56	110.48	62.03	62.11	36.96	296.54	296.53	92.17
FlexSPI		8-bit	560.55	595.71	367.57	29.99	110.25	6.46	62	18.6	2.35	279.17	18.67	3.9
FlexSPI		16-bit	904.28	1183.53	529.91	59.97	111.54	12.92	62.02	36.91	4.62	296.45	37.2	7.79
FlexSPI		32-bit	2331.69	2331.69	530.14	119.88	111.6	25.34	62.03	61.94	8.91	296.49	73.81	15.38
FlexSPI		BURST 8	2970.81	3076.81	529.68	531.03	111.57	110.33	62.02	62.11	36.96	296.5	296.49	92.17
ITCM	READ "ping-pong"	8-bit	367.66	635.24	381.16	31.57	101.25	5.34	34.32	1.77	2.37	83.06	3.15	3.93
ITCM		16-bit	692.87	1349.96	558.36	66.63	108.03	10.94	34.52	3.56	4.67	85.01	6.36	7.94
ITCM		32-bit	1054.2	2335.57	633.36	149.79	108.27	22.45	34.65	7.2	9.11	86.12	12.92	15.94
FlexSPI		8-bit	561	635.28	381.16	31.57	101.13	5.34	34.32	1.77	2.37	83.06	3.15	3.93
FlexSPI		16-bit	652.71	1349.77	558.04	66.63	107.99	10.94	34.52	3.56	4.67	85.08	6.36	7.94
FlexSPI		32-bit	2328.38	3075.84	633.61	149.8	108.23	22.46	34.65	7.2	9.11	86.11	12.92	15.94
ITCM	READ / WRITE	8-bit	360.96	1114.81	562.35	58.87	196.72	9.36	123.7	36.91	4.71	526.68	37.06	7.78
ITCM		16-bit	706.05	1878.54	948.7	116.97	200.47	18.65	123.81	72.68	9.22	591.62	73.25	15.53
ITCM		32-bit	1454.85	3676.29	1054.42	222.86	200.14	37.19	123.79	123.75	17.75	591.84	143.15	30.56
ITCM		BURST 4	2651.13	3069.12	1051.04	680.49	200.5	125.53	123.79	124.14	34.47	591.44	414.82	63.79
FlexSPI		8-bit	434.51	1053.75	561.64	59.23	192.48	9.36	123.7	36.91	4.71	526.76	37.06	7.78
FlexSPI		16-bit	1000.75	2080.07	948.79	116.97	200.23	18.64	123.78	72.68	9.22	591.59	73.25	15.53
FlexSPI		32-bit	2338.9	3086.47	1054.2	222.85	200.32	37.18	123.79	123.75	17.75	591.8	143.15	30.56
FlexSPI		BURST 4	3079.7	3080.66	1051.38	681.39	200.32	125.54	123.81	124.13	34.47	591.41	414.8	63.79
ITCM	WRITE	8-bit	367.74	595.82	595.67	596.94	331.57	327.54						
ITCM		16-bit	732.63	1183.53	599.05	600.18	332.43	327.85						
ITCM		32-bit	1184.67	2332.24	599.6	600.44	332.47	327.63						
ITCM		BURST 8	1347.92	1569.6	598.94	599.05	331.36	326.46						
FlexSPI		8-bit	336.11	595.85	595.85	596.83	331.57	327.48						
FlexSPI		16-bit	1183.39	1182.82	599.12	600.11	332.32	327.84						
FlexSPI		32-bit	2329.48	2329.48	599.45	600.29	332.47	327.61						
FlexSPI		BURST 8	1569.6	1569.6	599.05	598.87	331.35	326.49						

Tab. 2 RT10xx - Memory data throughput in MB/s



Coretx M7 - CORE @996MHz, IP BUS @240MHz, SDRAM @200MHz SDR, QSPI @99MHz SDR

Code location	Data access		Data buffers location											
			ITCM	DTCM	OCRAM		SDRAM				FLEXSPI		
											QSPI		
					D-CACHE		D-CACHE				D-CACHE		
	Type	Width			ON	OFF	ON		OFF		ON	OFF	
							Bus width					PRE-FETCH	
							32-bit	16-bit	32-bit	16-bit	ON	ON	OFF
ITCM	READ	8-bit	637.46	995.7	429.47	24.00	143.16	118.56	6.39	6.41	49.17	18.42	1.95
ITCM		16-bit	1225.57	1990.79	548.5	48.00	144.43	121.21	12.81	12.77	49.17	36.74	3.84
ITCM		32-bit	2275.62	3973.33	589.99	96.00	149.27	121.05	25.58	25.15	49.17	49.20	7.38
ITCM		BURST 8	5287.9	5287.9	589.2	590.63	149.25	121.04	149.30	120.98	49.17	49.24	29.90
FlexSPI		8-bit	896.72	995.64	429.24	24.00	143.09	118.58	6.40	6.42	49.17	18.42	1.95
FlexSPI		16-bit	1450.92	1990.54	548.52	48.00	144.73	121.08	12.82	12.78	48.94	36.74	3.83
FlexSPI		32-bit	3970.43	3970.43	589.99	96.00	148.88	121.14	25.57	25.14	49.17	49.20	7.38
FlexSPI		BURST 8	4796.73	5287.9	589.82	590.63	148.67	121.01	149.26	121.03	49.17	49.23	29.84
ITCM	READ "ping-pong"	8-bit	618.9	1062.05	449.68	25.38	128.96	110.3	5.39	5.38	28.11	1.51	1.96
ITCM		16-bit	1062.12	2274.67	586.76	54.08	135.17	114.62	11.04	11.03	28.23	3.02	3.87
ITCM		32-bit	2275.3	5286.19	665.82	123.87	139.38	114.71	23.19	22.81	28.29	6.09	7.50
FlexSPI		8-bit	694.02	1061.99	449.68	25.43	130.38	110.49	5.39	5.38	28.08	1.51	1.96
FlexSPI		16-bit	982.15	2274.98	580.93	54.08	134.93	114.47	11.03	11.04	28.24	3.02	3.87
FlexSPI		32-bit	3966.57	3980.11	665.87	123.87	139.02	114.76	23.18	22.81	28.27	6.09	7.50
ITCM	READ / WRITE	8-bit	625.06	1769.9	853.12	47.99	217.67	176.42	9.78	9.8	98.18	36.83	3.90
ITCM		16-bit	1238.12	3537.49	1084.64	96.00	166.31	144.12	19.56	19.57	98.19	73.47	7.68
ITCM		32-bit	2277.84	5305.09	1176.95	192.00	159.88	144.27	39.05	38.86	98.19	98.39	14.76
ITCM		BURST 4	3978.17	5294.76	1096.74	639.76	250.55	213.71	141.01	134.22	98.19	98.44	28.31
FlexSPI		8-bit	777.74	1873.75	853.12	47.99	190.23	160.53	9.79	9.80	98.18	36.83	3.90
FlexSPI		16-bit	1595.47	3184.71	1075.42	96.00	171.70	154.66	19.57	19.59	98.19	73.47	7.68
FlexSPI		32-bit	3990.82	6361.97	1176.95	192.00	159.76	144.24	39.07	38.86	98.19	98.39	14.77
FlexSPI		BURST 4	5293.05	5281.06	1095.79	639.56	249.23	213.13	141.1	134.11	98.19	98.45	28.31
ITCM	WRITE	8-bit	637.49	995.7	995.7	995.7	636.92	351.15	627.66	353.97			
ITCM		16-bit	1138.6	1990.54	1926.62	1926.62	636.99	351.62	636.69	354.09			
ITCM		32-bit	2274.35	3973.33	1930.49	1929.58	637.12	351.55	620.05	351.15			
ITCM		BURST 8	2651.25	2651.68	1924.12	1923.21	635.97	349.84	634.74	349.93			
FlexSPI		8-bit	485.74	995.64	995.64	995.64	629.47	351.31	636.2	351.35			
FlexSPI		16-bit	611.25	1990.79	1926.85	1926.85	636.92	351.25	636.59	351.47			
FlexSPI		32-bit	3978.17	3978.17	1930.49	1929.58	637.32	351.43	636.5	351.43			
FlexSPI		BURST 8	2651.68	2651.68	1924.35	1923.21	636.10	349.90	635.06	349.93			

Tab. 3 RT1176 - Cortex M7 core: Memory data throughput in MB/s

Tab. 3 RT1176 - Cortex M4 core: Memory data throughput in MB/s

NOTEs:

1. Read instruction sequence in Soc:

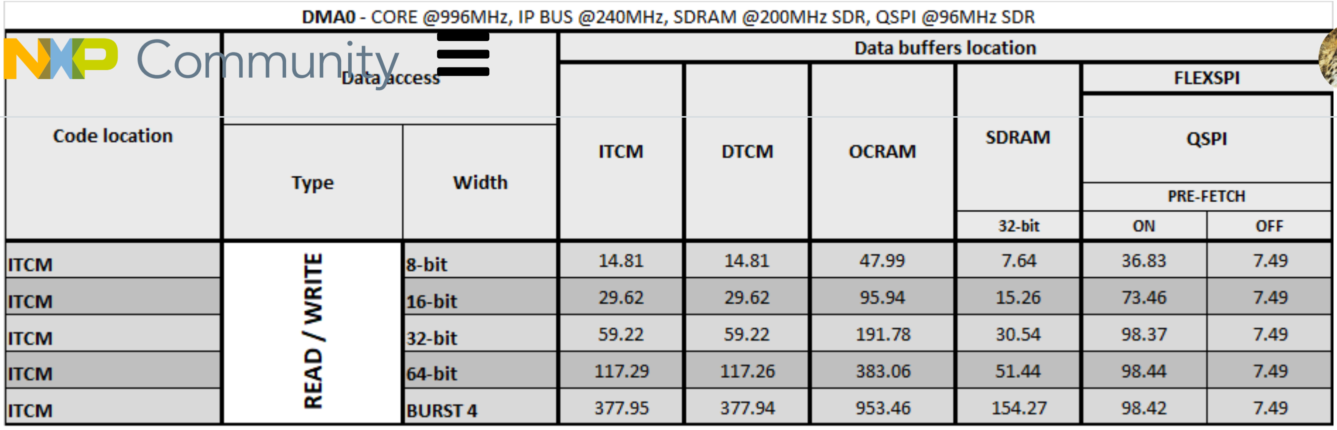
   core issue read request(T0) -> read instruction arrives SEMC via internal bus(T1) -> SEMC processes the read request(T2) -> SDRAM bus read timing start by burst length(T3) -> SEMC get the value (at first read clock of the burst) and return to core(T4) -> core get the response(T5) -> core can issue next read request(T6)

   - Core can just issue another read request after T6, from T0 to T6, it just wait the response.
   - The default burst length is 8 words, for each read request, the SEMC will always read the burst length from SDRAM, and return the request value ASAP, doesn't wait all burst clock finishing.
   - After T4, read clocks are still working on SDRAM bus to align the burst length, of course the SEMC may drop the undesired words if core doesn't request more.

   So

   - The sequence core issues read instructions is the key point of the low read performance.
   - About 16bit VS 32bit SDRAM bus, for a 4 bytes core read request, it's "2 read clock VS 1 read clock" on bus timing, the incremental one read clock is so insignificant comparing with core read issuing delay (T0 – T6) and the burst mode, so the performance results are similar.

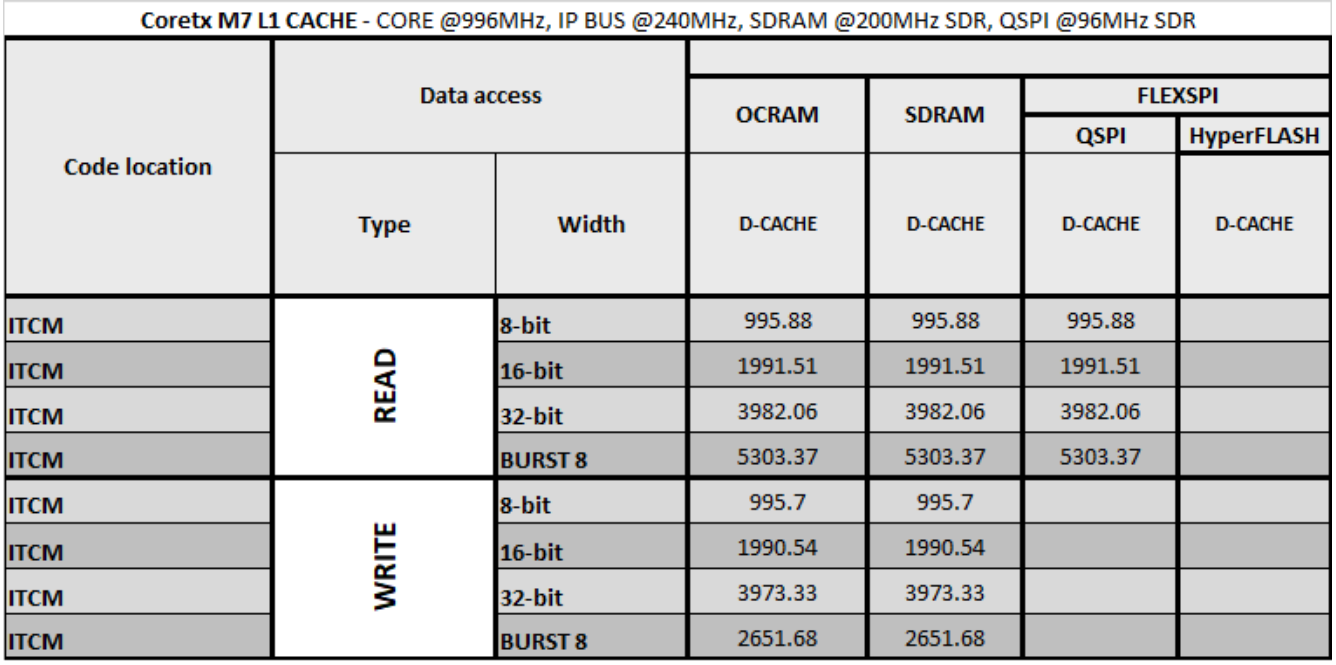
1. Write instruction sequence from core can be in burst mode without cache, and response is also unnecessary, so the improvement of 32bit VS 16bit will be significant for the write performance.
2. Also you can find the write performance results are similar between cache enable and cache disable due to the above reason.
3. For SDRAM bus timing, one read timing is also longer than one write timing.



DMA0 - CORE @996MHz, IP BUS @240MHz, SDRAM @200MHz SDR, QSPI @96MHz SDR

Code location	Data access		Data buffers location					
							FLEXSPI	
	Type	Width	ITCM	DTCM	OCRAM	SDRAM	QSPI	
							PRE-FETCH	
						32-bit	ON	OFF
ITCM	READ / WRITE	8-bit	14.81	14.81	47.99	7.64	36.83	7.49
ITCM		16-bit	29.62	29.62	95.94	15.26	73.46	7.49
ITCM		32-bit	59.22	59.22	191.78	30.54	98.37	7.49
ITCM		64-bit	117.29	117.26	383.06	51.44	98.44	7.49
ITCM		BURST 4	377.95	377.94	953.46	154.27	98.42	7.49

Tab. 4 RT117x - DMA0 bus master: Memory data throughput in MB/s (NOTE: FlexSPI represents read from flexspi location and write to DTCM location)



Coretx M7 L1 CACHE - CORE @996MHz, IP BUS @240MHz, SDRAM @200MHz SDR, QSPI @96MHz SDR

Code location	Data access					
			OCRAM	SDRAM	FLEXSPI	
					QSPI	HyperFLASH
	Type	Width	D-CACHE	D-CACHE	D-CACHE	D-CACHE
ITCM	READ	8-bit	995.88	995.88	995.88	
ITCM		16-bit	1991.51	1991.51	1991.51	
ITCM		32-bit	3982.06	3982.06	3982.06	
ITCM		BURST 8	5303.37	5303.37	5303.37	
ITCM	WRITE	8-bit	995.7	995.7		
ITCM		16-bit	1990.54	1990.54		
ITCM		32-bit	3973.33	3973.33		
ITCM		BURST 8	2651.68	2651.68		

Tab. 5 RT117x - L1 CACHE data throughput in MB/s (NOTE: for write only Write-Back policy has been tested)

References

[1] - https://www.nxp.com/docs/en/application-note/AN12437.pdf

[2] - https://www.nxp.com/docs/en/application-note/AN12042.pdf

Labels : BL: Auto BL: non-Auto

NXP Community

Add tags

15 Kudos Was this article helpful? YES NO

EDIT

COMMENT

SHARE

COMMENTS

vincent_aubinea 12-21-2020 11:

Hi Rastislav,

Thank you for this page, do you know why in i.MXRT1176 M4's benchmark table results seems so sloooow vs M7 for instance?

BR

Vincent

rastislav_pavlanin 03-04-2021 01

Hi @vincent_aubinea ,

sorry I just realized your comment now. Well, what we should consider here is:

- core frequency: CM7 996MHz vs. CM4 393MHz, ratio = 2.5
- bus widths:
  - CM7 ITCM 64-bit vs CM4 code TCM 32-bit
  - CM7 DTCM 2x32-bit (32-bit interleaved fashion -> 64-bit) vs CM4 system TCM 32-bit
  - to access non-TCM bus slaves as are FlexSPI or SDRAM or OCRAM: CM7 uses cached

    64-bit AXI vs CM4 the same bus as used to access system TCM it means system bus 32-bit
    - accesses to FlexSPI, SDRAM, OCRAM is done through one NIC on CM7
    - accesses to FlexSPI, SDRAM, OCRAM is done via XB, NIC (multiple arbitrations)
- core pipeline: this can affect performance on single cycle memories
    - CM7 uses super scalar pipeline: allow execute 2 instruction in one cycle
    - CM4 uses 3 stage pipeline: max. 1 instruction in 1 cycle, considering LDR/STR instruction which are mostly considered in my benchmark, it takes 2 cycles
- branching: in my benchmark I doing 32B transfers then jump back and do it again.
    - CM7: uses predictive branching -> I believe that allows to do branching (short jump) in 1 cycle
    - CM4: needs to re-fill pipeline at each branch, it means branch instruction will takes 3 cycles

regards

R.

alex_peck

01-11-2022 07:

Hi Rastislav,

Thank you for this excellent information. Can it be shared with customers?

Thanks,

Alex Peck

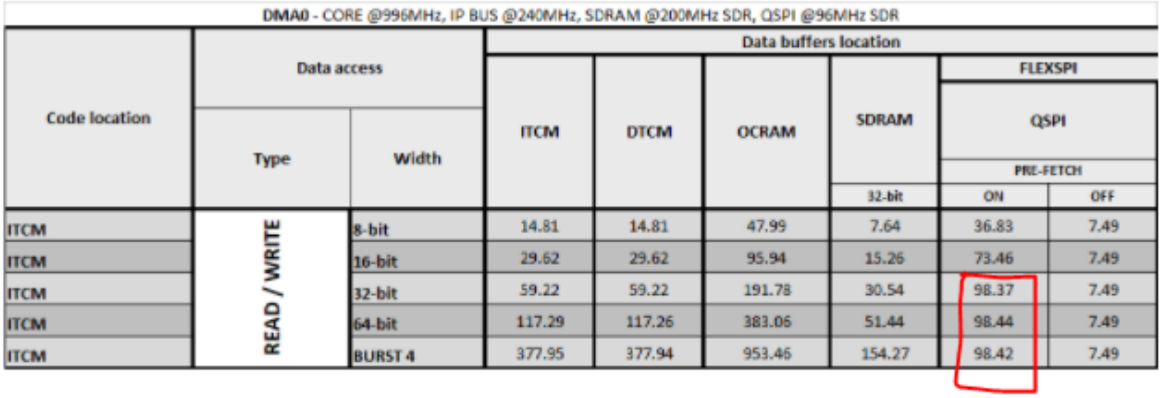
vincent_aubinea

01-26-2022 11:

Hi Rasdislav,

Are you sure the QSPI Flash is at 96MHz and is it a quad or an octal? because the max theoretical throughput for a QSPI SDR@96MHz is 48MB/s.

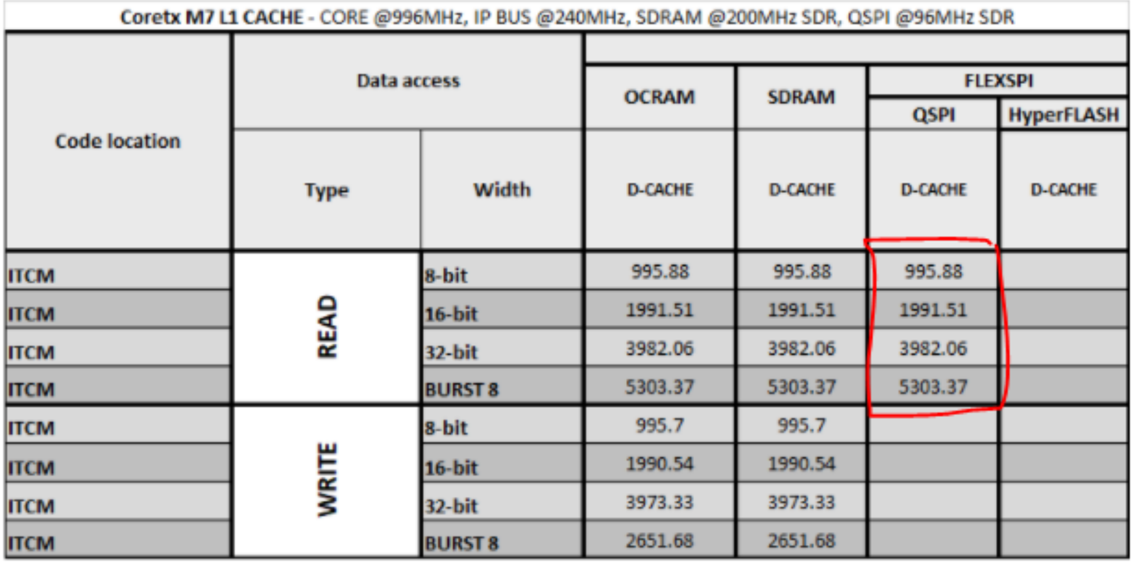
With the results you have I would say you have an Octal @99MHz (half of the SDRAM frequency):



DMA0 - CORE @996MHz, IP BUS @240MHz, SDRAM @200MHz SDR, QSPI @96MHz SDR

Code location	Data access		Data buffers location					
							FLEXSPI	
	Type	Width	ITCM	DTCM	OCRAM	SDRAM	QSPI	
							PRE-FETCH	
						32-bit	ON	OFF
ITCM	READ / WRITE	8-bit	14.81	14.81	47.99	7.64	36.83	7.49
ITCM		16-bit	29.62	29.62	95.94	15.26	73.46	7.49
ITCM		32-bit	59.22	59.22	191.78	30.54	98.37	7.49
ITCM		64-bit	117.29	117.26	383.06	51.44	98.44	7.49
ITCM		BURST 4	377.95	377.94	953.46	154.27	98.42	7.49

Tab. 4 RT117x - DMA0 bus master: Memory data throughput in MB/s (NOTE: FlexSPI represents read from flexspi location and write to DTCM location)



Coretx M7 L1 CACHE - CORE @996MHz, IP BUS @240MHz, SDRAM @200MHz SDR, QSPI @96MHz SDR

Code location	Data access					
			OCRAM	SDRAM	FLEXSPI	
					QSPI	HyperFLASH
	Type	Width	D-CACHE	D-CACHE	D-CACHE	D-CACHE
ITCM	READ	8-bit	995.88	995.88	995.88	
ITCM		16-bit	1991.51	1991.51	1991.51	
ITCM		32-bit	3982.06	3982.06	3982.06	
ITCM		BURST 8	5303.37	5303.37	5303.37	
ITCM	WRITE	8-bit	995.7	995.7		
ITCM		16-bit	1990.54	1990.54		
ITCM		32-bit	3973.33	3973.33		
ITCM		BURST 8	2651.68	2651.68		

Tab. 5 RT117x - L1 CACHE data throughput in MB/s (NOTE: for write only Write-Back policy has been tested)

If you still have your code, can you check the clocks config with my little clock dump function (may need to be adapted depending of the SDK version):

https://community.nxp.com/t5/RT-4-Digit-10xx-11xx-12xx/Clock-dump-on-i-MXRT1170/ta-p/1104443

Vincent

Comment

PREVIEW

Hint: @ links to members, content

☑ Email me when someone replies

CANCEL

POST YOUR COMMENT

Version history

Revision #: 7 of 7
Last update: 01-26-2022 05:18 AM
Updated by: rastislav_pavlanin

View Article History