Flash Layer Default Performance and Power Consumption Models

Pierre Olivier <pierre.olivier@univ-brest.fr>

September 16, 2014

Contents

CHAPTER

1 Modeling Performance and Power Consumption of a NAND Flash Storage Subsystem

1 Introduction

The equations presented here are used to model the performance (time taken) and energy consumed for the follwoing flash commands:

1. Legacy read, write, erase ;
2. Copy-back ;
3. Cache read, cache write ;
4. Multi plane read, write, erase ;
5. Multi plane copy-back, multi plane cache read, multi plane cache write ;
6. Lun-interleaved operations ;
7. Multi channel operations.

To do so, we use the variables and constants defined in Table 1.1.

name	description	unit	variable ?
TIN	Transfer In Nand, time taken to transfer of data (a page) from the page register to the NAND array in a plane.	μs	yes
TON	Transfer Out of Nand, time taken to transfer of a page from the NAND array to the page register.	μs	yes
IO	Time taken to transfer a page on the I/O bus.	μs	no
$BERS$	Block ERaSe, time taken to erase a flash block.	μs	no
$PTIN$	Power during TIN.	W	no
$PTON$	Power during TON.	W	no
PIO	Power during IO.	W	no
$PBERS$	Power during block erase.	W	no

Table 1.1: Variable and constants of our model

We assume the following:

- The time for sending commands opcodes and address is considered negligible, so it is not modeled ;
- Transfering a page from the page register on the I/O bus (read) takes the same time as transferring a page on the I/O bus to the page register (write). Time taken for these two operation is represented by IO ;
- TIN and TON can be variable, according to the index of the addressed page in its block (in particular in MLC NAND). So in fact we have terms like TIN_{addr} and TON_{addr} representing the time taken to transfer a

page to / from the nand array according to that page index within its containing block.

In this document, the unit of time is the microsecond. The unit of power is the watt, and the unit of energy is the microjoule.

2 Models

2.1 Legacy Read

An example of legacy read operation execution is presented on Figure 1.1.

Performance

$$T_{LegacyRead}(addr) = TON_{addr} + IO \tag{1.1}$$

$addr$ is the address of the page being read, so TON_{addr} represents the time to transfer this page from the NAND array to the page register.

Energy Consumption

$$E_{LegacyRead}(addr) = PTON * TON_{addr} + PIO * IO \tag{1.2}$$

2.2 Legacy Write

An example of legacy write operation execution is presented on Figure 1.1.

Performance

$$T_{LegacyWrite}(addr) = IO + TIN_{addr} \tag{1.3}$$

Energy Consumption

$$E_{LegacyWrite}(addr) = PIO * IO + PTIN * TIN_{addr} \tag{1.4}$$

2.3 Legacy Erase

An example of legacy erase operation execution is presented on Figure 1.1.

Performance

$$T_{LegacyErase} = BERS \tag{1.5}$$

Energy Consumption

$$E_{LegacyErase} = PBERS * BERS \tag{1.6}$$

2.4 Copy-Back

An example of copy back operation execution is presented on Figure 1.2.

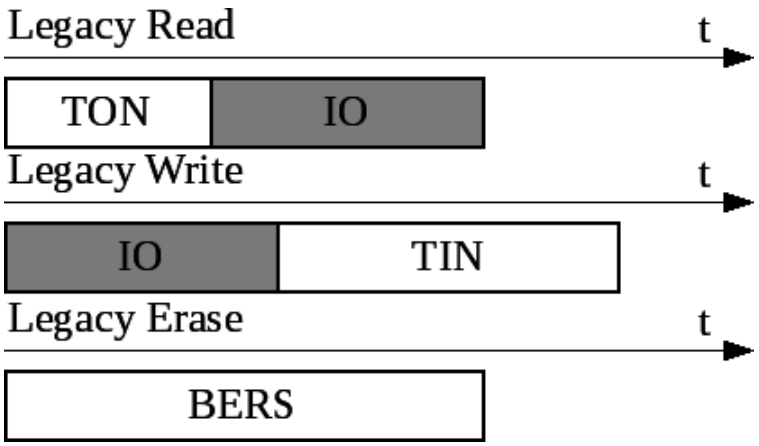

Figure 1.1: Legacy read, write and erase operations execution

Performance

$$T_{CopyBack}(src, dest) = TON_{src} + TIN_{dest} \tag{1.7}$$

src and $dest$ are the source (page read) and destination (page written) addresses of the copy-back operation.

Energy Consumption

$$E_{CopyBack}(src, dest) = PTON * TON_{src} + PTIN * TIN_{dest} \tag{1.8}$$

2.5 Cache Read

An example of cache read operation execution is presented on Figure 1.2.

Performance

$$T_{CacheRead}(addr_1, addr_2, ..., addr_n) = TON_{page_1} + \sum_{i=2}^{n} max(TON_{addr_i}, IO) + IO \quad \text{with } n \geq 2 \tag{1.9}$$

Energy Consumption

$$E_{CacheRead}(addr_1, addr_2, ..., addr_n) = \sum_{i=1}^{n} (PTON * TON_{addr_i} + PIO * TIO) \quad \text{with } n \geq 2 \tag{1.10}$$

$addr_1, ..., addr_n$ are the addresses of the different pages being read sequentially during the cache read operation. Note that n must be at least 2.

2.6 Cache Write

An example of cache write operation execution is presented on Figure 1.2.

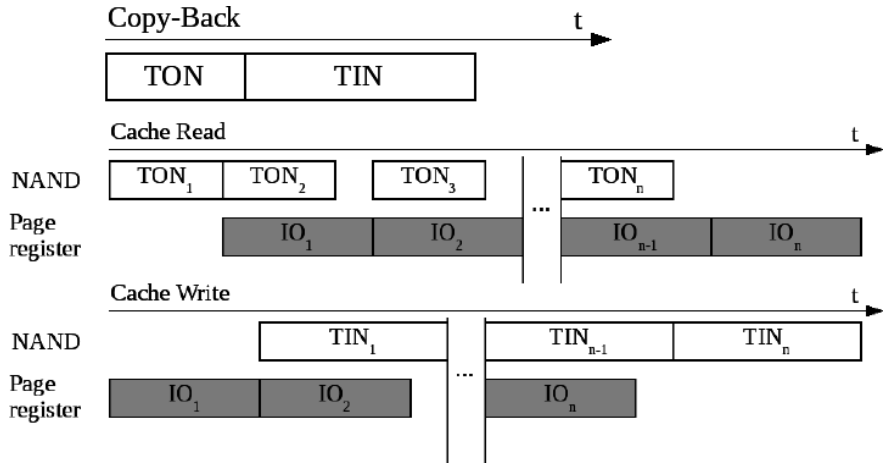

Figure 1.2: Copy-back, cache read and cache write operations execution

Performance

$$T_{CacheWrite}(addr_1, addr_2, ..., addr_n) = IO + \sum_{i=2}^{n} max(TIN_{addr_i}, IO) + TIN_{addr_n} \quad \text{with } n \geq 2 \tag{1.11}$$

Energy Consumption

$$E_{CacheWrite}(addr_1, addr_2, ..., addr_n) = \sum_{i=1}^{n} (PTIN * TIN_{addr_i} + PIO * TIO) \quad \text{with } n \geq 2 \tag{1.12}$$

2.7 Multi Plane Read

An example of multi plane read operation execution is presented on Figure 1.3.

Performance

We define T_{P_i} as the time of one read operation in the plane i:

$$T_{P_1}(addr_1) = TON_{addr_1} + IO$$
$$T_{P_i}(addr_i) = max(T_{P_{i-1}}, TON_{addr_i}) + IO \tag{1.13}$$

$addr_i$ is the address of the page to read in the plane i. So we have:

$$T_{MultiPlaneRead}(addr_1, addr_2, ..., addr_n) = max_i(T_{P_i}) \quad \text{with } n \geq 2 \tag{1.14}$$

n is the number of planes participating in the multi plane operation.

Energy Consumption

$$E_{MultiPlaneRead}(addr_1, addr_2, ..., addr_n) = \sum_{i=1}^{n}(PTON * TON_{addr_i}) + n * PIO * IO \quad \text{with } n \geq 2 \tag{1.15}$$

2.8 Multi Plane Write

An example of multi plane write operation execution is presented on Figure 1.3.

Performance

T_{P_i} is the time of one write operation in the plane i. It is defined as follows:

$$T_{P_i}(addr_i) = i * IO + TIN_{addr_i} \tag{1.16}$$

$addr_i$ is the address of the page to write in plane i. So we have:

$$\begin{aligned} T_{MultiPlaneWrite}(addr_1, addr_2, ..., addr_n) &= max_i(T_{P_i}) \\ &= max_i(i * IO + TIN_{addr_i}) \\ &\text{with } n \geq 2 \end{aligned} \tag{1.17}$$

n is the number of planes participating in the multi plane operation.

Energy consumption

$$E_{MultiPlaneWrite}(addr_1, addr_2, ..., addr_n) = n * PIO * IO + \sum_{i=1}^{n}(PTIN * TIN_{addr_i}) \quad \text{with } n \geq 2 \tag{1.18}$$

2.9 Multi Plane Erase

An example of multi plane erase operation execution is presented on Figure 1.3.

Performance

$$T_{MultiPlaneErase}(addr_1, addr_2, ..., addr_n) = BERS \quad \text{with } n \geq 2 \tag{1.19}$$

n is the number of planes participating in the multi plane operation.

Energy Consumption

$$E_{MultiPlaneErase}(addr_1, addr_2, ..., addr_n) = n * PBERS * BERS \quad \text{with } n \geq 2 \tag{1.20}$$

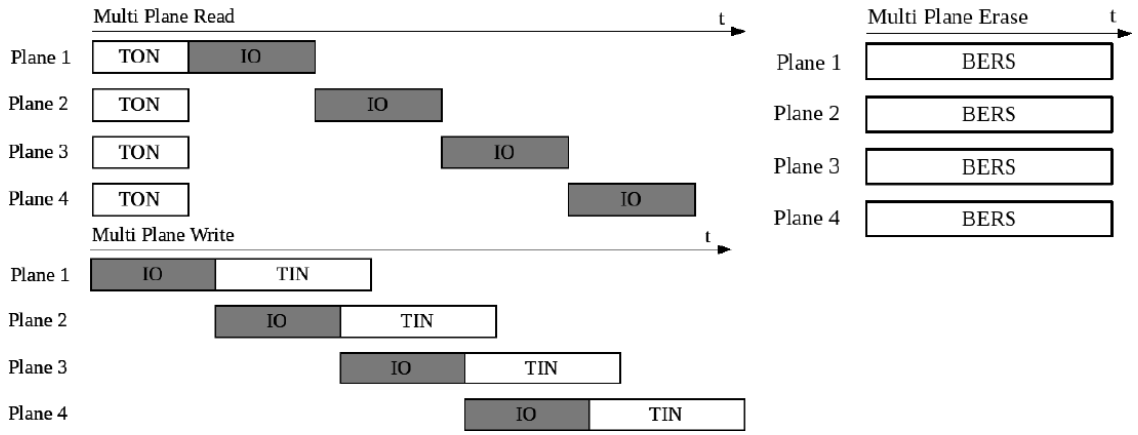

Figure 1.3: Multi plane read, write and erase operations execution

2.10 Multi Plane Copy-Back

Performance

$$T_{MultiPlaneCopyBack}(src_1, src_2, ..., src_n, dest_1, dest_2, ..., dest_n) = max_i(TON_{src_i} + TIN_{dest_i}) \quad \text{with } n \geq 2 \tag{1.21}$$

n is the number of planes participating in the multi plane operation.

Energy Consumption

$$E_{MultiPlaneCopyBack}(src_1, src_2, ..., src_n, dest_1, dest_2, ..., dest_n) = \sum_{i=1}^{n}(PTON * TON_{src_i} + PTIN * TIN_{dest_i}) \quad \text{with } n \geq 2 \tag{1.22}$$

2.11 Multi Plane Cache Read and Multi Plane Cache Write

The multi plane cache read and write operations can be caracterized as a set of lists of addresses. Each list represent a cache read / write operation in one plane. So the is one list per plane participating in the multi plane cache operation. Each list contains the addresses of the pages to read or write sequentially for the corresponding cache operation. For example, a multi plane cache operation on four plane with three pages read / written sequentially in each plane can be described as follows:

$$[addr_{1,1}, addr_{1,2}, addr_{1,3}, addr_{2,1}, addr_{2,2}, addr_{2,3}, addr_{3,1}, addr_{3,2}, addr_{3,3}, addr_{4,1}, addr_{4,2}, addr_{4,3}] \tag{1.23}$$

$addr_{i,j}$ represents the address of the j-th page to read / write in plane i. The multi plane cache read operation has one $TON_{i,j}$ and one $IO_{i,j}$ related to each page read. Similarly, the multi plane cache write operation has one $IO_{i,j}$ and one $TIN_{i,j}$ for each page written. Those values are depicted on the example of multi plane cache read and write on Figure 1.4 and Figure 1.5. Note that IO is a constant, therefore each $IO_{i,j}$ is actually equal to IO (with the exeception when $addr_{i,j}$ is $none$, as explained below). However, $TON_{i,j}$ and $TINi, j$ are variables.

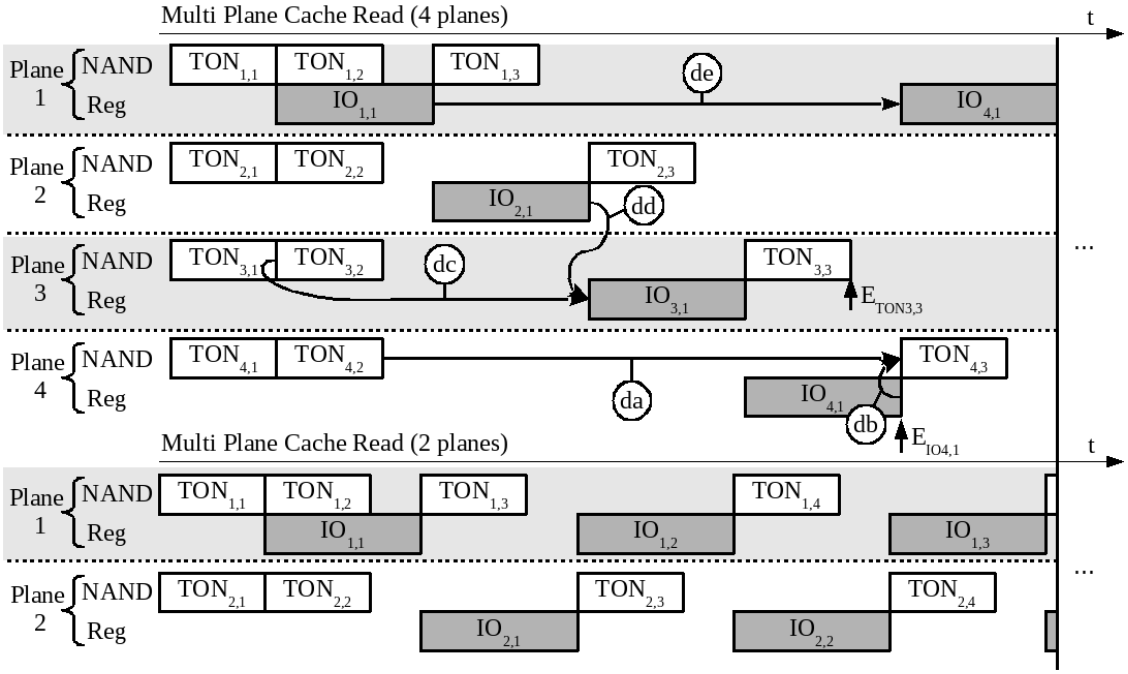

Figure 1.4: Multi plane cache read operation execution

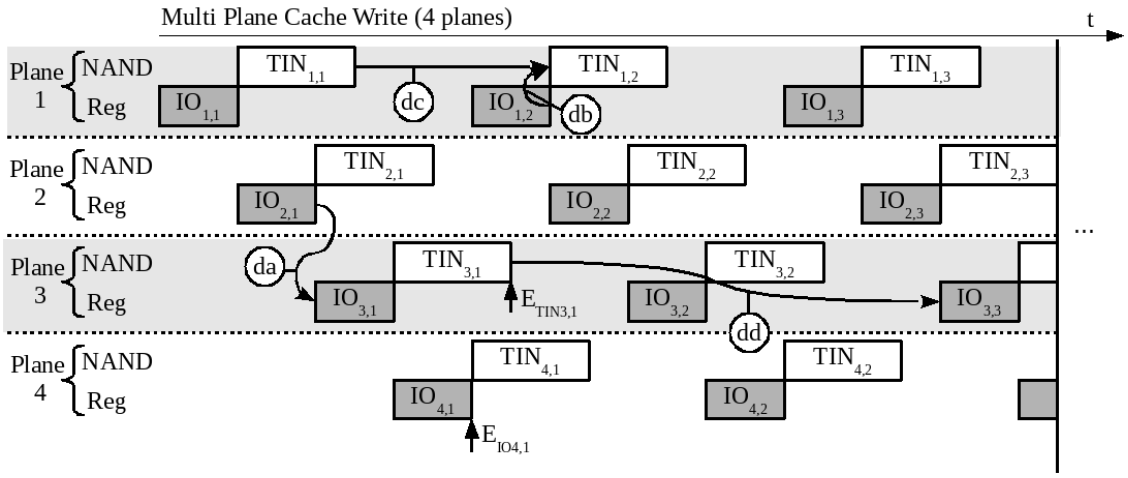

Figure 1.5: Multi plane cache write operation execution

In the case which the number of page to read / write sequentially is different from one plane to another, the size of each list is set to the size of the biggest list. We introduce the term *none*, used to fill the 'holes' in smaller lists. For example, a multi plane cache operation on two planes, with four pages accessed sequentially in the first plane, and only two pages accessed sequentially in the second, will be represented as follows:

$$[addr_{1,1}, addr_{1,2}, addr_{1,3}, addr_{1,4}, addr_{2,1}, addr_{2,2}, addr_{2,3} = none, \\ addr_{2,4} = none] \tag{1.24}$$

none terms will be useful in the performance / power consumption computation: we define that TON, IO and TIN values for an address equal to *none* as 0:

$$TON_{i,j} = \begin{cases} TON_{addr_{i,j}} & \text{if } addr_{i,j} \neq none \\ 0 & \text{otherwise} \end{cases} \tag{1.25}$$

$$TIN_{i,j} = \begin{cases} TIN_{addr_{i,j}} & \text{if } addr_{i,j} \neq none \\ 0 & \text{otherwise} \end{cases} \tag{1.26}$$

$$IO_{i,j} = \begin{cases} IO & \text{if } addr_{i,j} \neq none \\ 0 & \text{otherwise} \end{cases} \tag{1.27}$$

We also define, for multi plane cache read / write operations, the terms $E_{TON_{i,j}}$, $E_{IO_{i,j}}$ and $E_{TIN_{i,j}}$. They are respectively the moments in time were the TON, IO and TIN operation just finished for the related page read / write. $E_{TON_{3,3}}$ and $E_{IO_{4,1}}$ are illustrated as an example on Figure 1.4. $E_{TIN_{3,1}}$ is illustrated on Figure 1.5.

In the following equations, n is the number of planes participating in the multi plane cache read / write operation, and m_i is the number of pages read / written in plane i. m_max is the number of page read / written in plane containing the most important set of pages accessed. For our example presented in the expression 1.24, $n = 2$, $m_1 = 4$, $m_2 = 2$, and $m_{max} = max_i(m_i) = m_1 = 4$.

Multi Plane Cache Read - Performance

$E_{TON_{i,j}}$ depends of:

1. $E_{IO_{i,j-2}}$ (db on Figure 1.4): the cache register containing page $(i, j-2)$ must be first emptied. It is then filled with page $(i, j-1)$, coming from the page register. As the page register is emptied, $TON_{i,j}$ can occur ;
2. $E_{TON_{i,j-1}}$ (da on Figure 1.4) ;
3. $TON_{i,j}$.

$E_{IO_{i,j}}$ depends of:

1. $E_{IO_{i-1,j}}$ (dd on Figure 1.4): the IO bus must be free ;
2. $E_{TON_{i,j}}$ (dc on Figure 1.4): the page to output must be in the cache register ;
3. $E_{IO_{i,j-1}}$ (de on Figure 1.4) ;
4. $IO_{i,j}$.

According to this, the execution time of the multi plane cache read operation can be modelled as follows:

$$\begin{aligned} E_{IO_{1,1}} &= TON_{1,1} + IO \\ E_{IO_{1,j}} &= max(E_{IO_{n,j-1}}, E_{TON_{1,j}}, E_{IO_{i,j-1}}) + IO && \text{with } j \neq 1 \\ E_{IO_{i,j}} &= max(E_{IO_{i-1,j}}, E_{TON_{i,j}}, E_{IO_{i,j-1}}) + IO && \text{with } j \neq 1 \text{ and } i \neq 1 \end{aligned} \tag{1.28}$$

$$\begin{aligned} E_{TON_{i,1}} &= TON_{i,1} \\ E_{TON_{i,2}} &= E_{TON_{i,1}} + TON_{i,2} \\ E_{TON_{i,j}} &= max(E_{IO_{i,j-2}}, E_{TON_{i,j-1}}) + TON_{i,j} && \text{with } j > 2 \end{aligned} \tag{1.29}$$

Finally:

$$T_{MultiPlaneCacheRead}(addr_{1,1}, addr_{1,2}, ..., addr_{1,m_{max}}, addr_{2,1}, ..., addr_{n,m_{max}}) = max_i(E_{IO_{i,m_{max}}}) \quad \text{with } n \geq 2 \text{ and } m_{max} \geq 2 \tag{1.30}$$

Multi Plane Cache Read - Energy Consumption

$$E_{MultiPlaneCacheRead}(addr_{1,1}, addr_{1,2}, ..., addr_{1,m_{max}}, addr_{2,1}, ..., addr_{n,m_{max}}) = \sum_{i=1}^{n} [\sum_{j=1}^{m_{max}} (TON_{addr_{i,j}} * PTON + IO_{i,j} * PIO)] \quad \text{with } n \geq 2 \text{ and } m_{max} \geq 2 \tag{1.31}$$

Multi Plane Cache Write - Performance

$E_{TIN_{i,j}}$ depends of:

1. $E_{TIN_{i,j-1}}$ (dc on Figure 1.5) ;
2. $E_{IO_{i,j}}$ (db on Figure 1.5): IO transfer from the host must be complete to then load the page in the NAND array ;
3. $TIN_{i,j}$.

$E_{IO_{i,j}}$ depends of:

1. $E_{IO_{i-1,j}}$ (da on Figure 1.5): the IO bus must be free for the IO transfer of page (i, j) ;
2. $E_{TIN_{i,j-2}}$ (dd on Figure 1.5): when the page $(i, j-2)$ is fully transfered to the NAND array, the page register is empty. So the page $(i, j-1)$ can be placed in that page register, coming from the cache register. Then, as the cache register is empty, $TIN_{i,j}$ can occur. ;
3. $IO_{i,j}$.

The execution time of the multi plane cache write operation can the be modelled as follows:

$$\begin{aligned}
E_{IO_{1,1}} &= IO_{i,j} \\
E_{IO_{1,j}} &= E_{IO_{n,j-1}} + IO_{i,j} && \text{with } 2 \leq j \leq 3 \\
E_{IO_{1,j}} &= max(E_{IO_{n,j-1}}, E_{TIN_{1,j-2}}) + IO_{i,j} && \text{with } j > 3 \\
E_{IO_{i,j}} &= E_{IO_{i-1,j}} + IO_{i,j} && \text{with } i \neq 1 \text{ and } 2 \leq j \leq 3 \\
E_{IO_{i,j}} &= max(E_{IO_{i-1,j}}, E_{TIN_{i,j-2}}) + IO_{i,j} && \text{with } i \neq 1 \text{ and } j > 3
\end{aligned} \tag{1.32}$$

$$\begin{aligned}
E_{TIN_{1,1}} &= E_{IO_{1,1}} + TIN_{1,1} \\
E_{TIN_{i,j}} &= max(E_{IO_{i,j}}, E_{TIN_{i,j-1}}) + TIN_{i,j} && \text{with } i \neq 1 \text{ and } j \neq 1
\end{aligned} \tag{1.33}$$

Once again here n is the number of planes participating in the multi plane operation. Finally we have:

$$T_{MultiPlaneCacheWrite}(addr_{1,1}, addr_{1,2}, ..., addr_{1,m_{max}}, addr_{2,1}, ..., addr_{n,m_{max}}) = max_i(E_{TIN_{i,m_{max}}}) \quad \text{with } n \geq 2 \text{ and } m_{max} \geq 2 \tag{1.34}$$

Multi Plane Cache Write - Energy Consumption

$$E_{MultiPlaneCacheWrite}(addr_{1,1}, addr_{1,2}, ..., addr_{1,m_{max}}, addr_{2,1}, ..., addr_{n,m_{max}}) = \sum_{i=1}^{n} [\sum_{j=1}^{m_{max}} (TIN_{addr_{i,j}} * PTIN + IO_{addr_{i,j}} * PIO)]$$
$$\text{with } n \leq 2 \text{ and } m_{max} \leq 2 \tag{1.35}$$

2.12 LUN-interleaved commands

Performance

Energy Consumption

2.13 Multi Channel

Performance

$$T_{MultiChannel}(cmd_1, cmd_2, ..., cmd_n) = max_i(T_{cmd_i})$$
$$\text{with } n \geq 2 \tag{1.36}$$

n is the number of channels participating in the multi channel operation.
$cmd_1, ..., cmd_n$ are the commands sent in parallel amongst different channels. T_{cmd_i} is the time required to process cmd_i, computed using one of the previously presented equations.

Energy Consumption

$$E_{MultiChannel}(cmd_1, cmd_2, ..., cmd_n) = \sum_{i=1}^{n} E_{cmd_i}$$
$$\text{with } n \geq 2 \tag{1.37}$$

E_{cmd_i} represents the energy consumed during the execution of cmd_i, computed using the previously presented equations.