Memory resource

KEK IPNS E-sys
Ryotaro Honda, Yun-Tsung Lai

Overview of this session

RAM and FIFO have already been used by many people.
Here we explain typical usage and things to be aware of.

- If you need random access: RAM。
- if you want to retrieve the stored items in order: FIFO。

Random Access Memory (RAM)

In Xilinx FPGA, block RAM is a dual-port RAM where one storage area is shared with two independent ports.

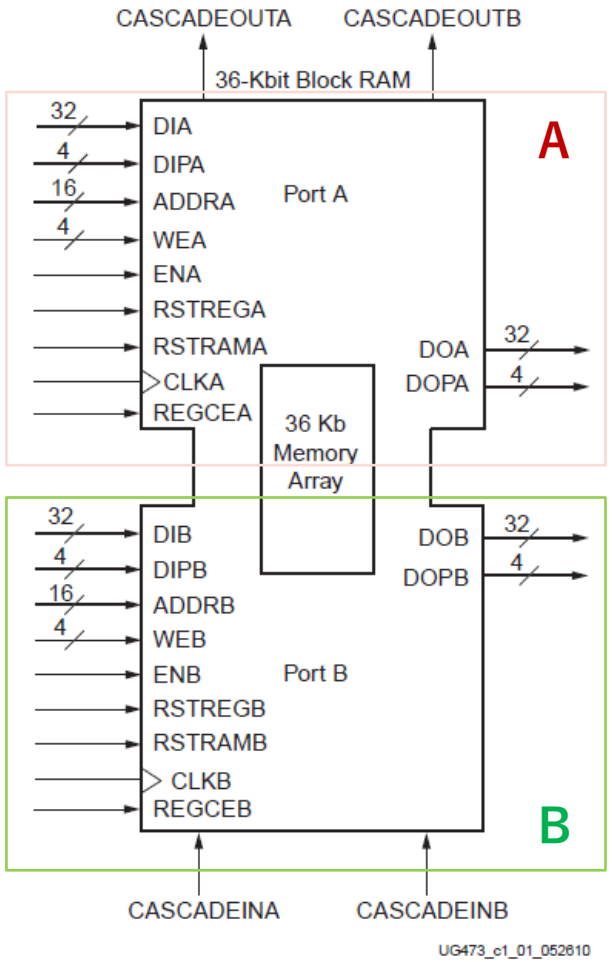


図 1-1 : RAMB36 の TDP データ フロー
RAMB36's TDP data flow

Point

True Dual Port (TDP)

- A mode where any location in one memory area can be accessed from two independent ports.
- Can be written and read independently.

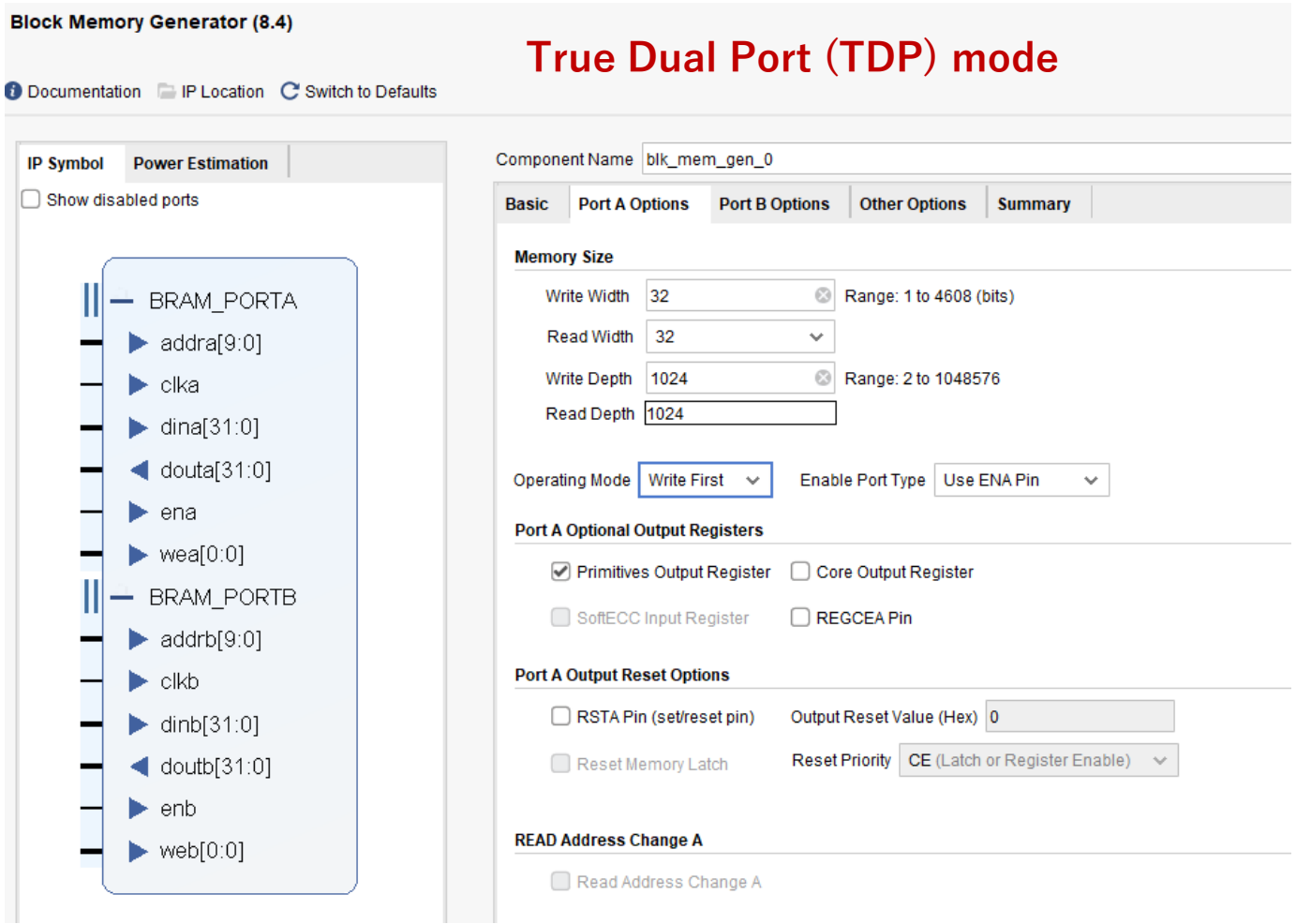
Simple Dual Port (SDP)

- A mode where port A for reading and port B for writing.
- Port A and B can be combined to double the port width.

- DI: Data input
- DIP: Parity bit (can be used as extended data)
- ADDR: Address

Block RAM is mostly generated by IP.
Here, we will use the components generated from the IP catalog to explain the key points of usage.

Block RAM (TDP mode)



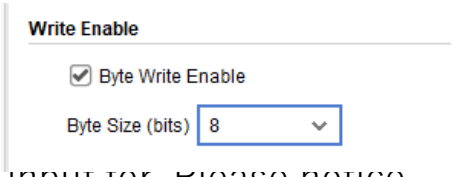
A mode that allows independent reading and writing from ports A and B to the shared memory.
Since the configuration is the closest to the BRAM primitive, it can be applied to other modes if understanding TDP well.

ENA, ENB

- **Port** enable. Don't forget to to enable it for both reading and writing.

WEA, WEB

- The BRAM's raw port is byte write enable. If generated in the IP catalog, the entire data width will be written at once (normal write enable) unless the byte write enable mode is checked.



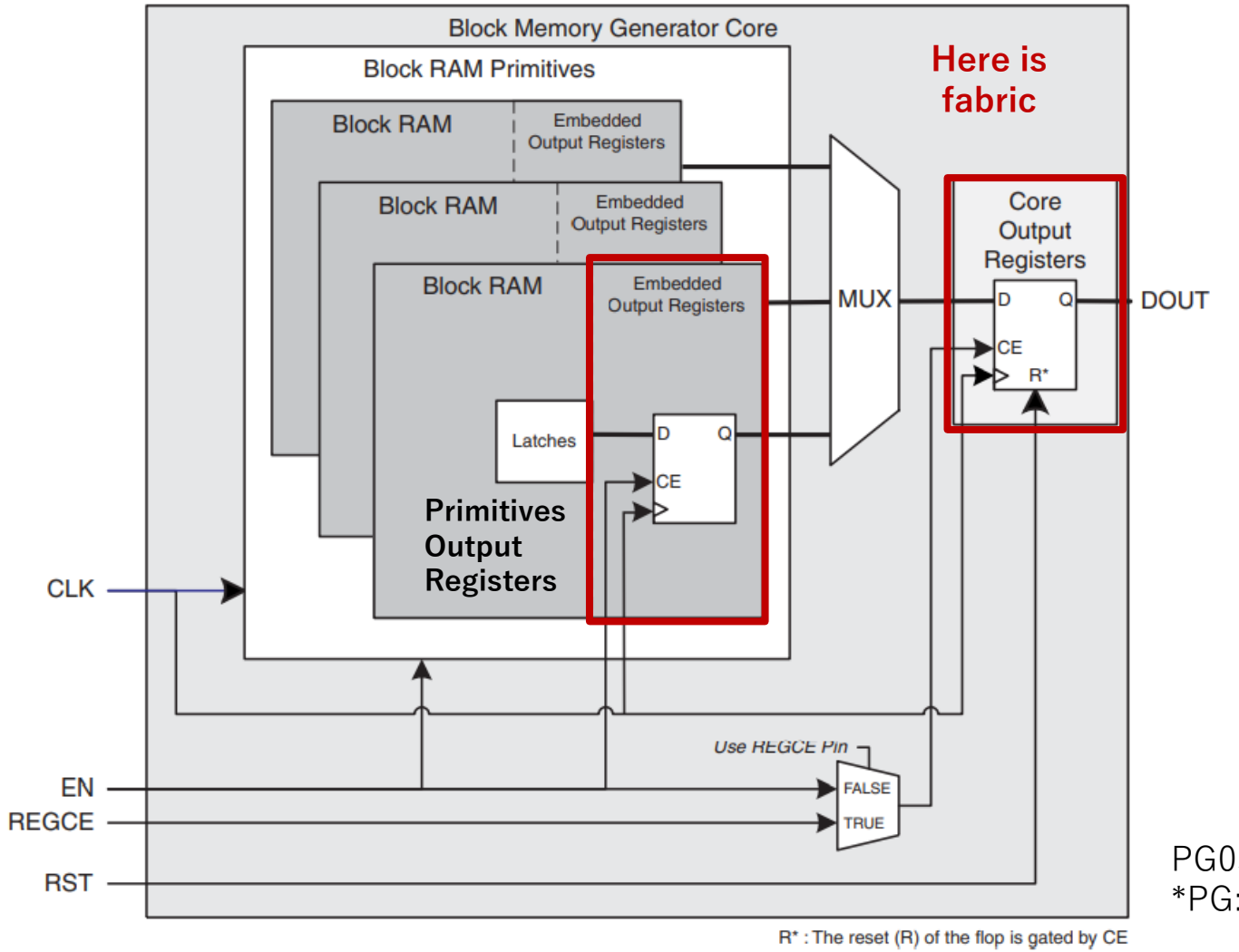
RSTA (B)

- **Output register**'s reset input for. Please notice that "initial values are given to memories, but there is no dynamic mass reset signal".

Output registers

- Next page

Output registers



Since the IP core may bundle multiple BRAM primitives, registers can be set even after MUX.

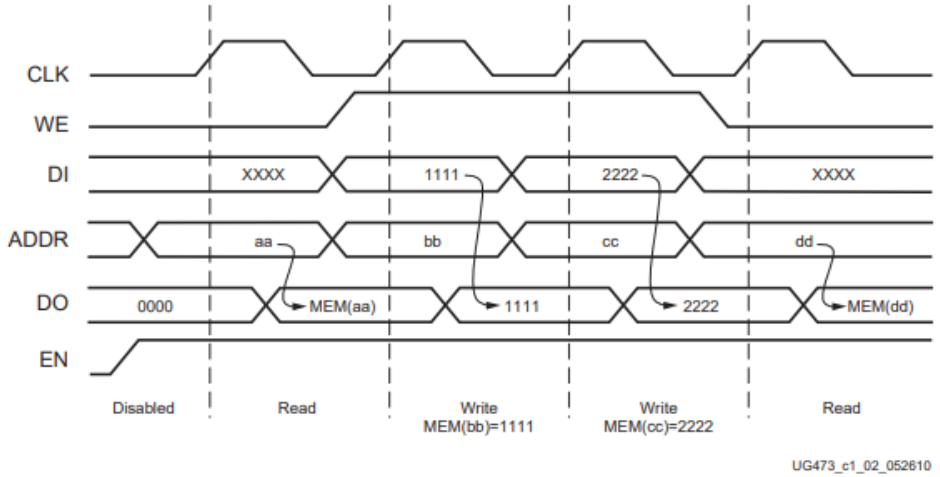
Since the MUX is a combinational circuit, if you want to operate at high speed, it is easier to increase the throughput by implementing both.

PG058
*PG: product guide

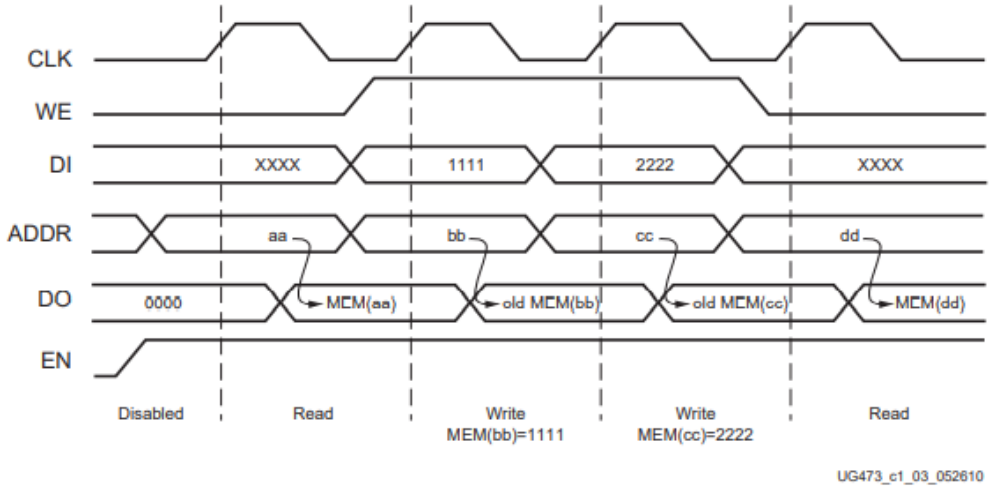
Writing mode

To write to the memory, ENA(B) is set to HIGH, which inevitably causes reading at the same time.
Let's use these 3 types properly based on the purpose.

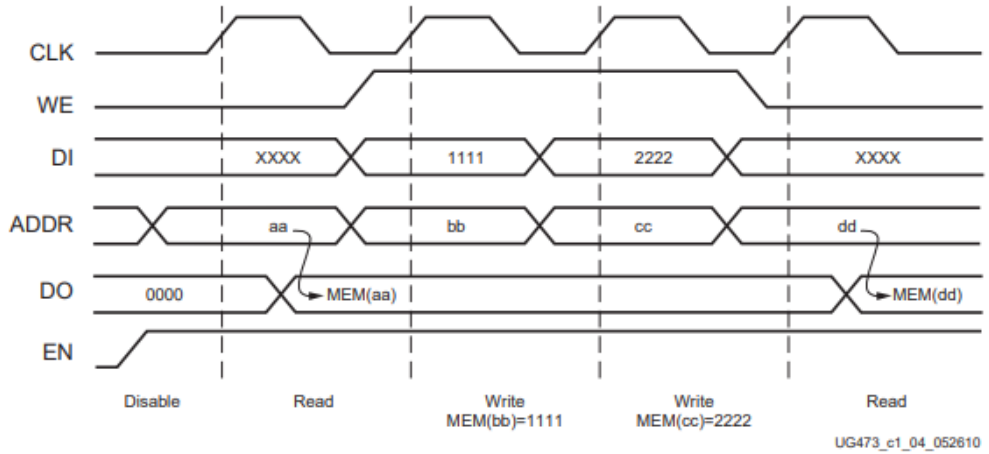
WRITE_FIRST (passthrough mode)



READ_FIRST

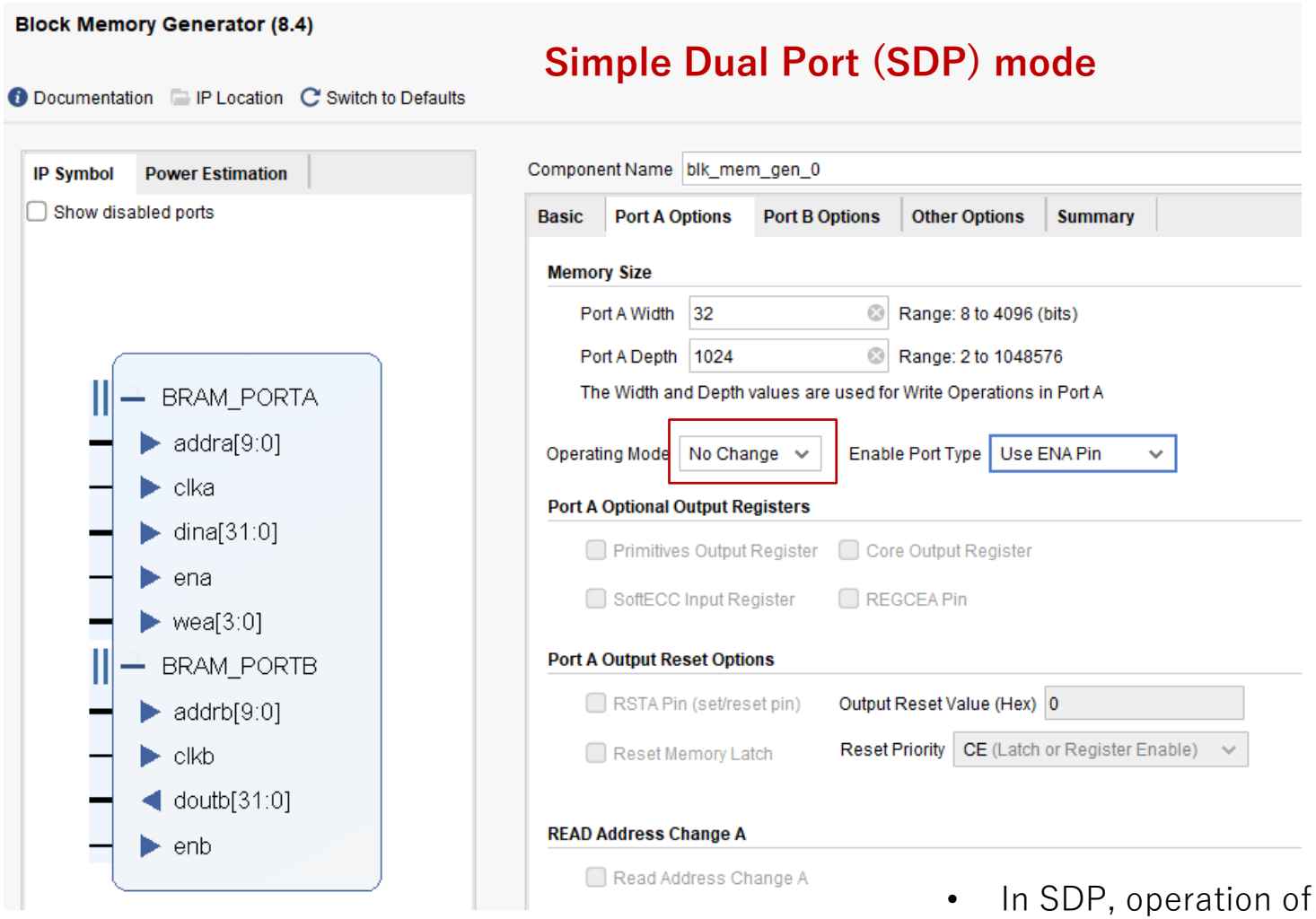


NO_CHANGE



Do not read during writing

Block RAM (SDP mode)



A mode where port A is for write-only and port B is for read-only. Address assignment is independent.

ENA, ENB

- **Port** enable. Don't forget to to enable it for both reading and writing.

WEA

- Same as TDP, but exists only on port A.

RSTA (B)

- **Output register's** reset input. Please notice that "initial values are given to memories, but there is no dynamic mass reset signal".

Output registers

- Same as TDP

- In SDP, operation of port B is fixed to WRITE_FIRST.
- Since reading is hidden for port A, it seems meaningless to select the write mode. But if you select NO_CHANGE, it can save power a little bit.
  ⇒ BRAM operates internally by itself.

Example of using Block RAM

Here are what I can think of now. Please give comments if you have other ideas.

Histogram
- The address is the bin number, and the value is the count number.

Correction table
- If the correction function f(x) is very complex, the correction function can be given as a discrete table.
- The address is the x-axis, and the value is the correction constant.

Big encoder/decoder
- Similar to the correction table. Get the converted value from the address value.

Ring buffer
- If the pointer position for reading can be changed dynamically.
- If just want to read the written data in order, FIFO can be considered.

In either case, it is necessary to specify the address randomly, and it can be done only using RAM.

TDP and SDP

This is also what I can think of.

TDP

- A pattern where reading and writing to the same memory occur from two different clock domains.
  - Group the slow and fast ADC data together (place them on consecutive addresses).
  - Two different clock domains are using the same correction table.
- When accessing from the same clock domain, improvement in memory efficiency and throughput can be expected.

If you need 2 ports for either writing or reading: TDP.

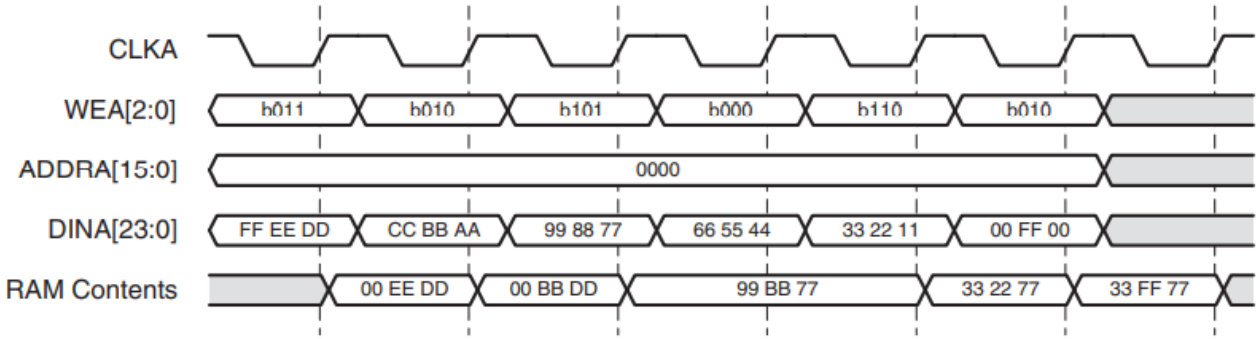
SDP

- When the separation of reading and writing is decided.
- In many cases, ring buffer can be implemented in SDP.

If there is benefit to use TDP, go with TDP.
Otherwise, SDP.

Recommendation for byte write

Example of byte write



For a function to change the contents of memory every 8(9)-bit, TDP is good.

Example of utilization

- Correct only the lower 8-bit of 64-bit data later.
- Handle the data whose length is changeable

- Byte writing is more resource efficient than expanding the address width. (because it is a function of BRAM primitives)
- Easier to speed up than extending the address length (I think)

Please be careful for the conflict in byte write

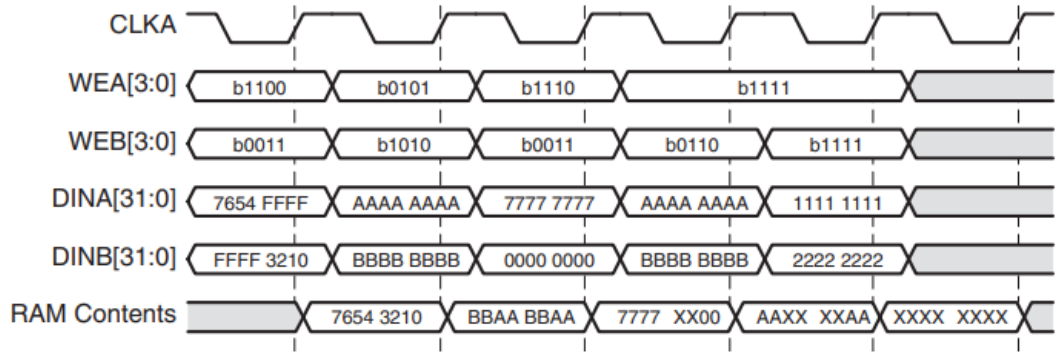


Figure 3-15: **Write-Write Collision Example**

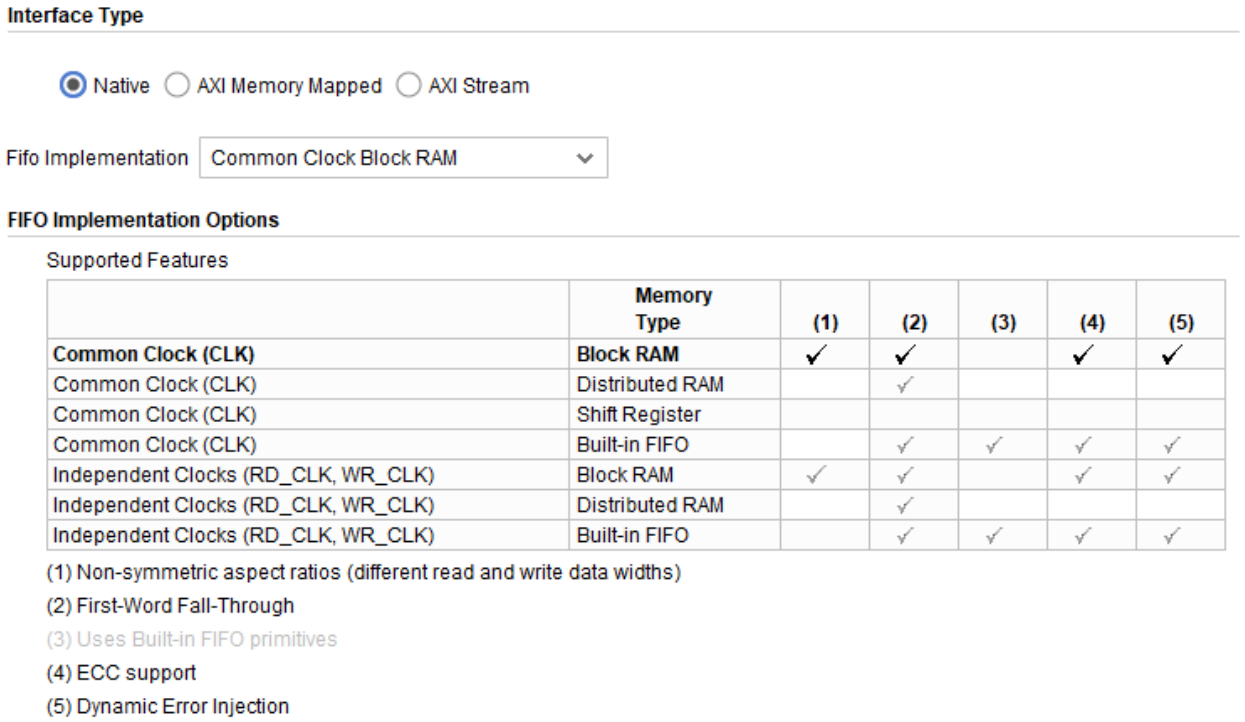
Considering using byte write enable, it is convenient for the FPGA internal data structure with 8(9)-bit units.
(resource efficiency is not optimized)

Electronics System Group

FIFO

openit
Open source consortium of Instrumentation

Generating FIFO



	Memory Type	(1)	(2)	(3)	(4)	(5)
Common Clock (CLK)	**Block RAM**	✓	✓		✓	✓
Common Clock (CLK)	Distributed RAM		✓			
Common Clock (CLK)	Shift Register					
Common Clock (CLK)	Built-in FIFO		✓	✓	✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	✓	✓		✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Distributed RAM		✓			
Independent Clocks (RD_CLK, WR_CLK)	Built-in FIFO		✓	✓	✓	✓

(1) Non-symmetric aspect ratios (different read and write data widths)
(2) First-Word Fall-Through
(3) Uses Built-in FIFO primitives
(4) ECC support
(5) Dynamic Error Injection

Option of memory blocks

Distributed RAM

- Implement memory in CLB.

Block RAM

- Use BRAM primitives.
  - ECC's support is the same as BRAM.
- The FIFO control part is fabric dependent.

Built-in FIFO

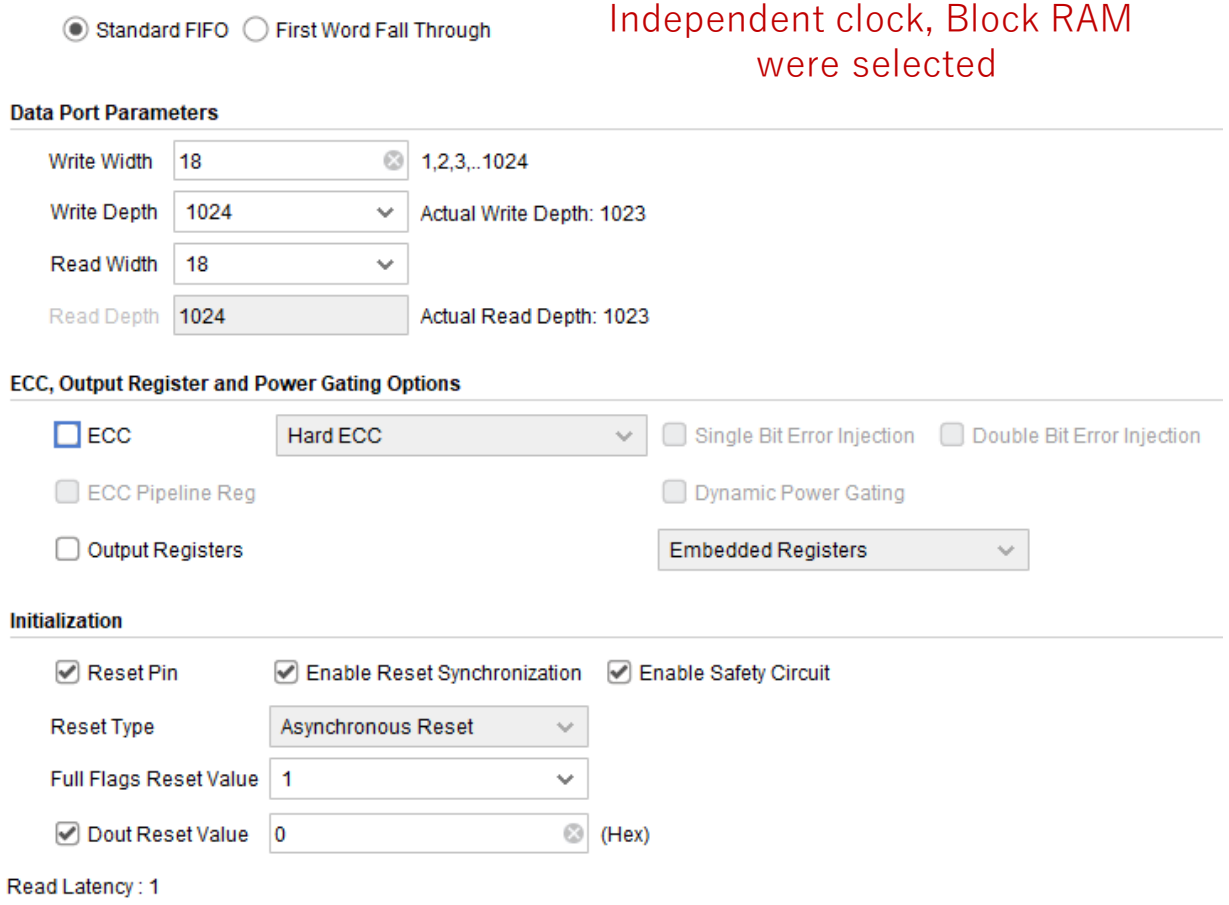
- Use BRAM primitives.
  - ECC's support is the same as BRAM.
- FIFO control part also uses dedicated block resources and is fabric-independent.
  - Easy to operate at high speed. Does not consume CLB.

Please notice that: Non-symmetric aspect ratios (FIFO with different input/data data width)
is not available for built-in FIFOs.

The experiment groups using SiTCP use this function a lot.

Generating FIFO

Independent clock, Block RAM were selected



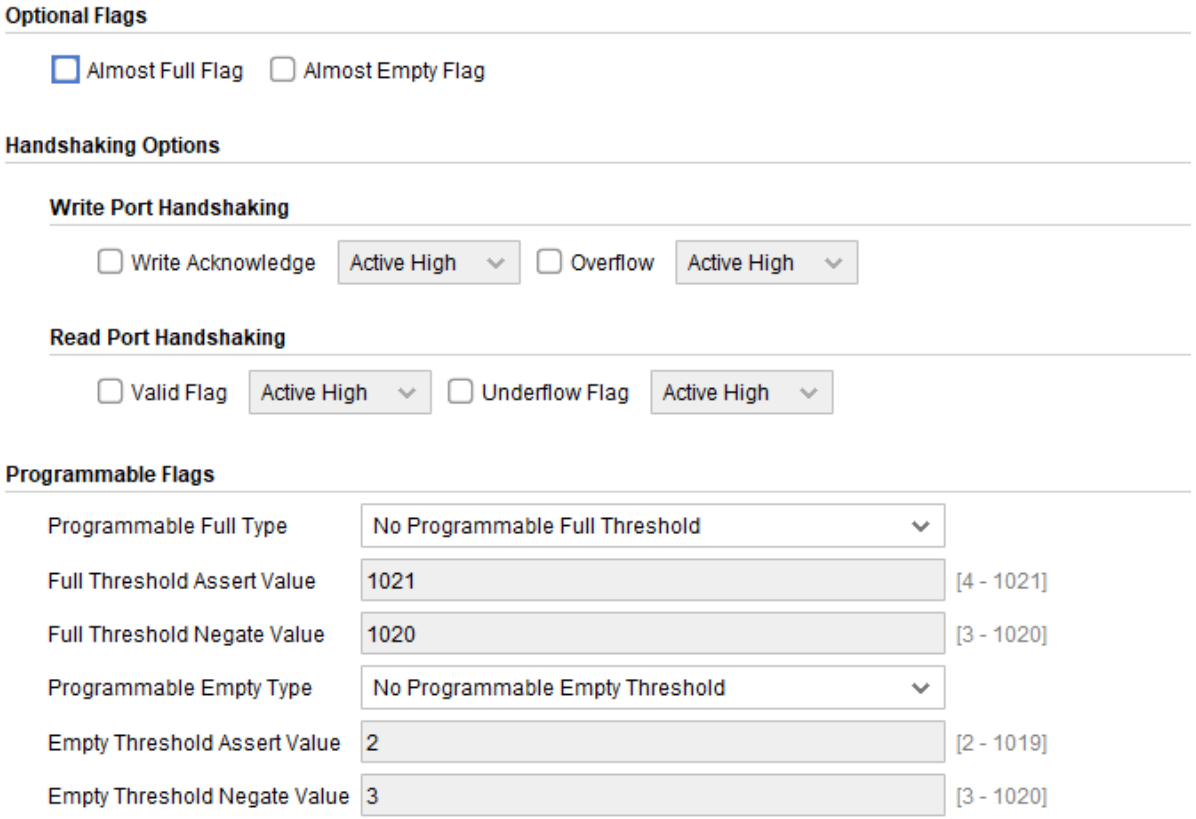
ECC and output register are the same as BRAM.

Dynamic power gating

- Only available with Built-in FIFO. When it is not used, it would sleep to reduce power consumption.

Since the contents are BRAM, there is no function to erase all the contents of the memory at once.
However, it is possible to create a state with empty FIFO.

Generating FIFO



Read valid

- A signal indicating when the data output at the same clock edge is valid. It is very important.
- Read latency depends on FIFO parameter, so we cannot make a design by thinking that read enable was set to HIGH before how many clocks ago. It would be a bug.
- It is kept as the write enable to the next FIFO.

Programmable Full Threshold

- A flag to indicate when the set value is exceeded (when it is HIGH). Essential for safe design.
- When it is full and can no longer receive data, it must request the upstream circuit to stop sending data.
- It may take several clocks to reach upstream circuit, and upstream memory will also have latency for reading.
- The flag is designed to give a grace period from issuing a stop request until write enable actually becomes low.

Example of using FIFO

WRCLK and RDCLK are the same

Upstream circuits:
write at whenever you like

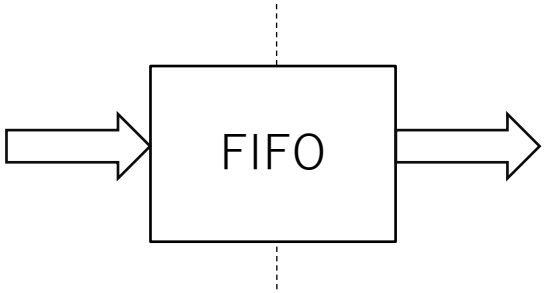
Downstream circuit:
Read at whenever you like
until it is empty

The easiest way to use it.
Temporary storage for data
For Programmable Full (pg-full)Threshold, Read Valid, Empty,
no problem if keeping an eye on it.

Digression
Empty transitions synchronously with RDCLK.
If you put Empty's reversal in read enable, it will automatically start reading.

WRCLK≠RDCLK clock transfer

Upstream circuits:
write at whenever you like



Downstream circuit:
Read at whenever you like
until it is empty

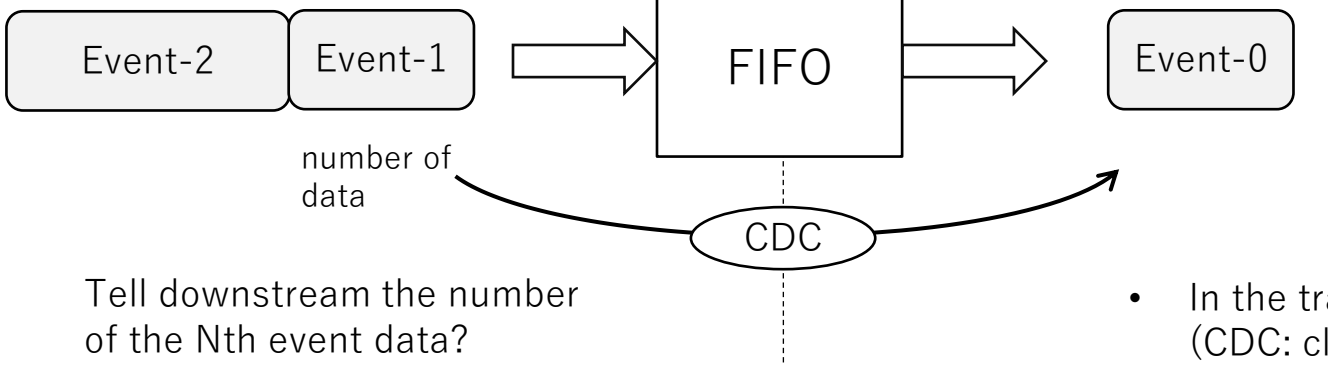
A typical way to use FIFO.
Buffer for safe clock transfer.
If pg-full and valid, empty signals are seen, not a problem

Example of using FIFO

Writing data strings with boundaries, such as variable-length event data

WRCLK≠RDCLK clock transfer

Since the downstream circuit wants to process each event, we want to stop reading once at the boundary of the event.

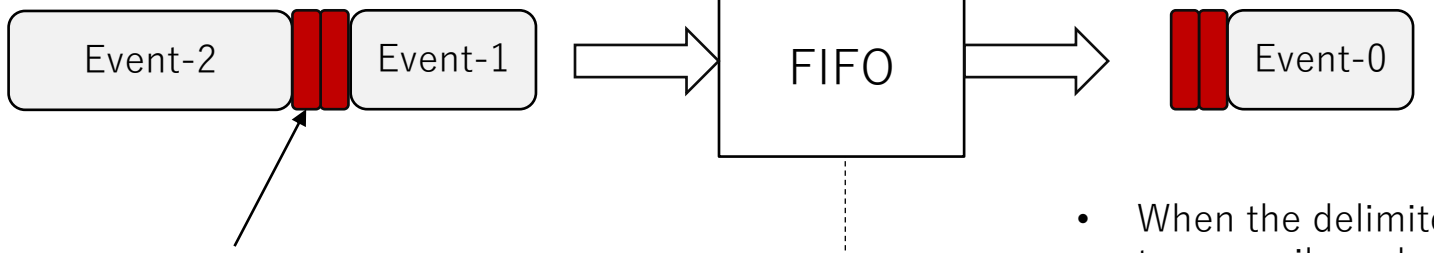


Tell downstream the number of the Nth event data?

- In the transmission of data numbers, clock transfer (CDC: clock domain crossing) happens
- It is somewhat complicated because it counts the number and controls the readout.

Example of using FIFO

Writing data strings with boundaries, such as variable-length event data



delimiter

Delimiter is put between each event.

WRCLK≠RDCLK clock transfer

Since the downstream circuit wants to process each event, we want to stop reading once at the boundary of the event.

- When the delimiter is found, reading stops temporarily and waits for the judgment from the circuit, which controls the data flow.

Processing becomes easy

Point

- If you control read enable after seeing the read result, there will always be a delay (extra data will be read)
  - Insert 2 delimiters successively if the reading latency is 1.
  - Insert 3 if the reading latency is 2.

Digression

- In First-data-fall-through mode, the following data appears in DO, but it is necessary to judge whether it is really valid data by looking at each flag. In addition, it is necessary to correctly understand the latency until flag assertion.
- It is safer to judge from the DO output with read valid.

FIFO' synchronicity

When using FIFO, the operation is usually asynchronous.
Even if WRCLK and RDCLK are synchronous, flag latency is still complicated .
If really want synchronization, read UG473 and PG057 carefully.

Case 1: Writing to an Empty FIFO

Prior to the operations performed in Figure 2-6, the FIFO is completely empty.

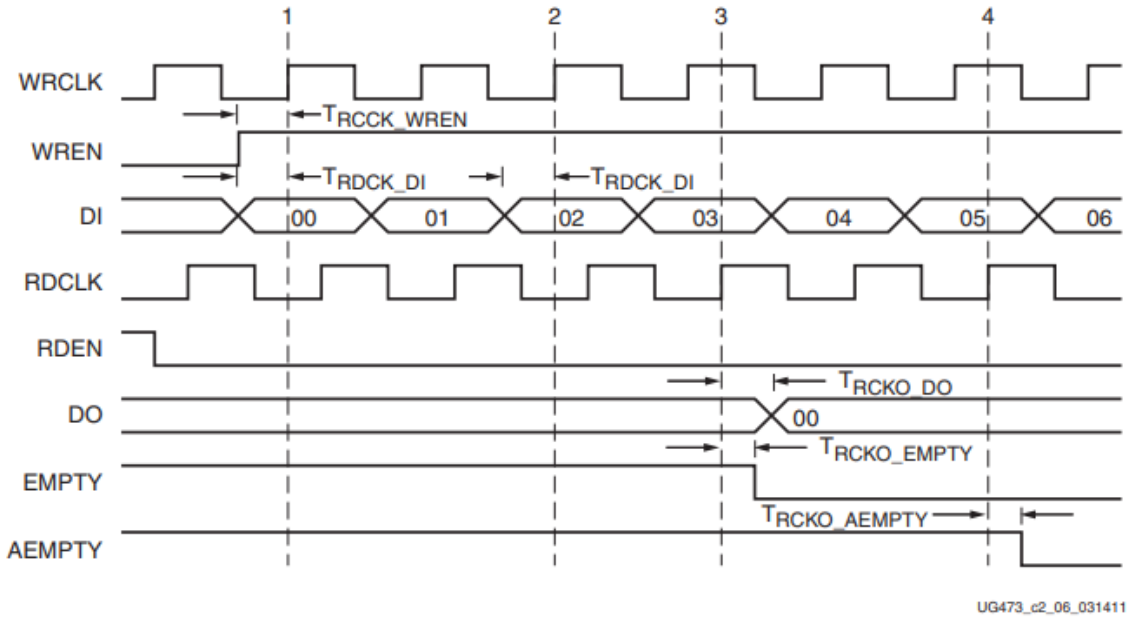


Figure 2-6: **Writing to an Empty FIFO in FWFT Mode**

Since the Built-in FIFO primitive is enabled for writing, empty is deasserted after 3 RDCLK cycles (normal mode) or 4 cycles (FDFT).

However,

If the rising WRCLK edge is close to the rising RDCLK edge, EMPTY could be deasserted one RDCLK period later.

it can be seen that it is not uniquely determined by the phase relationship.

UG473 describes the built-in FIFO primitive.
Please notice the explanation.

Please see chapter of latency in PG057 for flag latency of FIFO generated by IP.

FIFO's reset

- UG473 states that keeping the reset signal HIGH for 3 clock cycles each for WRCLK and RDCLK for all the reset flags of built-in FIFO primitives.
- Seems that PG057 doesn't mention the reset length.

It is safer to input 3-clock-cycle length for reset to FIFO.

Electronics System Group

Error Correction Code

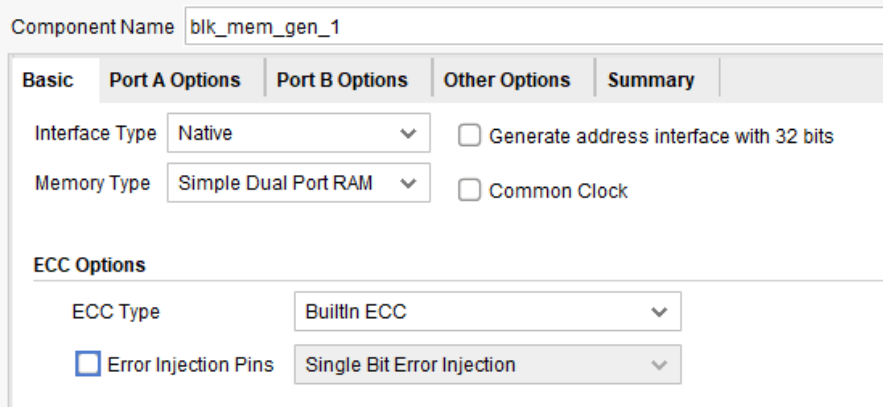
openit
Open source consortium of Instrumentation

Xilinx 7 series FPGA's ECC

Electronics System Group

- Xilinx FPGA supports Hamming code and HSIAO code for ECC (Error Correction Code).
  - (The principle is too long, so omitted)
- Both are based on the parity bits.
  - An 1-bit parity bit knows that a single bit error has occurred, but it cannot be corrected because the place of error is unknown.
  - For both Hamming code and HSIAO code, the single bit error's place can be identified, and the double bit error can only be known to the occurrence.

Block RAM's case



Since ECC supports only the SDP mode at the BRAM primitive level, ECC can only be selected in SDP mode in the IP Catalog.

Since it is configured as 72x512, resource efficiency is poor depending on the data width.

Single bit error

- Reading a memory region with bit flipping will assert sbiterr along with the data.
- **The output is error-corrected** by the ECC function. **The memory contents remain uncorrected**.

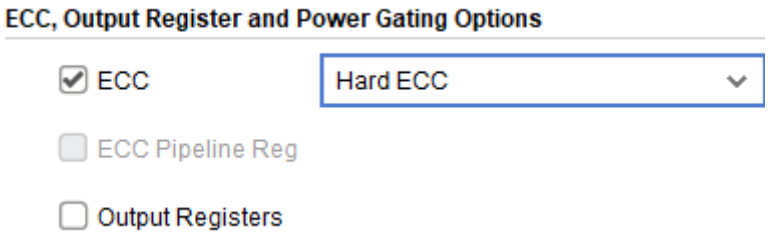
Double bit error

- Reading a memory region with bit flips will assert dbiterr along with the data.
- **Since it cannot be corrected, the output is not corrected and it remains as it is.**

rdaddrecc

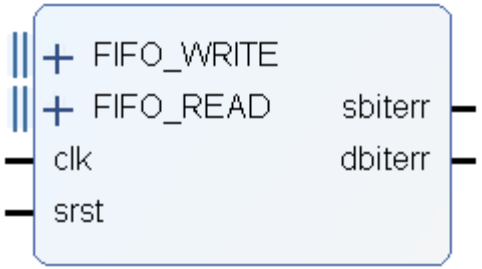
- A line representing at the address where the current data was stored.
- It is used to identify the location of bit error.

FIFO's case



Only supported for Block RAM and built-in FIFOs.
Since both the contents are BRAM primitives, they are configured as 72x512.
Again, resource efficiency is poor depending on the data width.

It is good to check how much BRAM is consumed in Summary.
Since it should fit in 18K BRAM, 36K BRAM is always selected for the data width.



Single bit error

- Reading a memory region with bit flipping will assert sbiterr along with the data.
- **The output is error-corrected** by the ECC function.

Double bit error

- Reading a memory region with bit flips will assert dbiterr along with the data.
- **Since it cannot be corrected, the output is not corrected and it remains as it is.**

If the data has been read once, it will never be read again in FIFO, so there is no way to identify the address where an error occurred.

ECC independent of FPGA memory resources

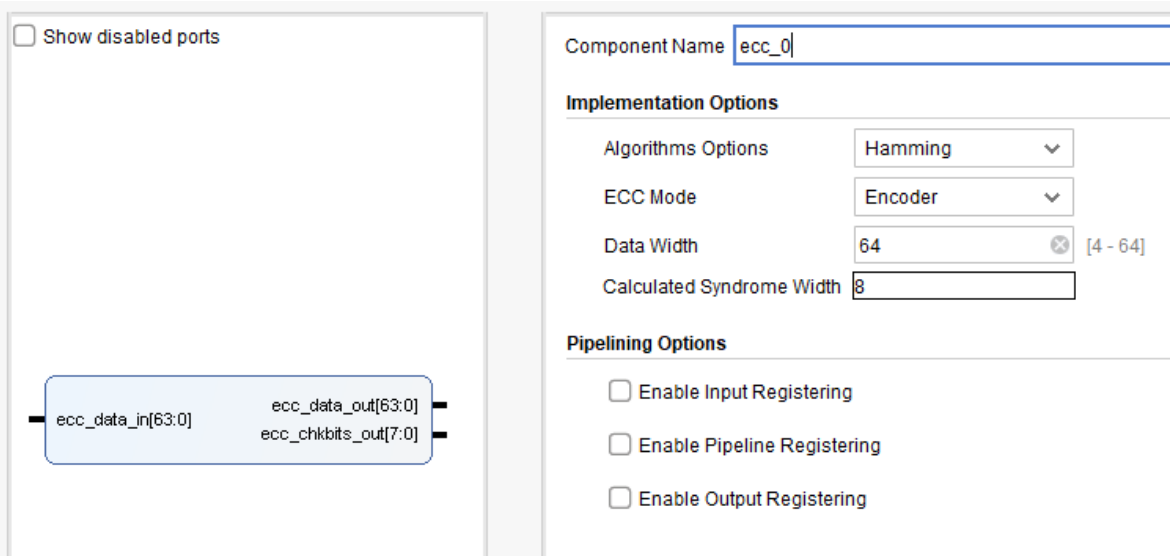
The ECC encoder/decoder is provided as an IP.
Use it when you want to add ECC to external memory or distributed RAM.

Encoder

- ECC is calculated from the input data, and a check bit string is given.
- External memory, etc, is written together with data.

Decoder

- Error detection and possible correction are performed from the check bit and data.



Since the ECC encoder/decoder is a combinational circuit, it is generated by consuming CLB.

Pipelining is applied for high-speed operation.

See ECC's actual action

Let's see BRAM's built-in ECC's functionality and ECC encoder/decoder's functionality in action.

BRAM

- Generate BRAM in SDP mode and enable Built-in ECC.。
- Enable Error Injection Pins and select Single and Double Error injection.
  - A signal to inserts data corruption intentionally.
  - Use it to check how ECC works and whether downstream circuit can handle it.
  - Input to injection pins with an one shot pulse.
- Input data and address shall be given by VIO.

ECC encoder/decoder

- Generate Encoder and Decoder respectively and connect them directly.
- To create intentional data corruption, a bit-reversed pattern is generated from VIO and XORed with the data output by the encoder.
- Input data shall be given by VIO.

Exercise
EX3