High-speed concentration of sorted data streams for triggerless HEP experiments

presented by: Marek Gumiński

Triggerless High Energy Physics experiments

In classical HEP experiments particles are grouped into bunches, that collide at preset time. Data acquisition may be triggered at that time.

Certain experiments benefit from continuous particle flow resulting in “continuous” collision.

With continuous particle flow, there is no external trigger source, DAQ must be ready to accept data at any time.

Readout data in triggerless experiments consists of continuous stream of small (ie 32 bits) data packets.

Typical data packet contain actual measurement (ADC value) and its timestamp. Other information may be also included.

Data Acquisition

Efficient data processing requires readout data to be encapsulated in packets containing entire data from certain time period.

Online creation of time packets requires strictly sorted data stream.

Each FEE sends data whenever it is acquired, producing rising timestamp data stream.

Certain samples may be slightly out of order (maximum disorder should be defined), but generally timestamp is rising.

Data of certain timestamp may be shifted in different FEE streams.

A0 A1 A3 A7
B0 B4 B7 B9 ➡ A0 B0 A1 A3 B4 A7 B7 B9

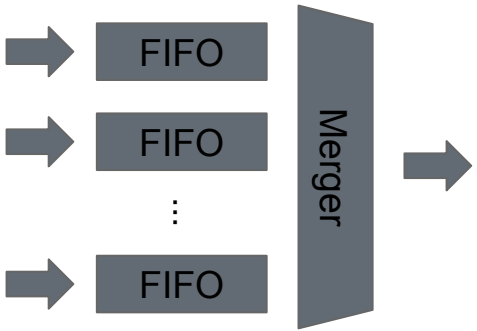
If FEE data streams are not sorted, it has to be done before stream merging. Not included in this presentation.

Merging

Buffer (with FIFOs) S inputs to account for shifts in different data streams.

Select N oldest sample from all input data FIFOs.

Make sure that no FIFO is empty. New sample on previously empty input may turn out to be older than previously output.



Maximum stream bandwidth is equal to the number of samples going through the pipeline in parallel when all samples are valid (no empty samples).

$$B_{max} = N * f_{clk}$$

Often pipeline has lower bandwidth than the maximum possible (multiple empty cycles).

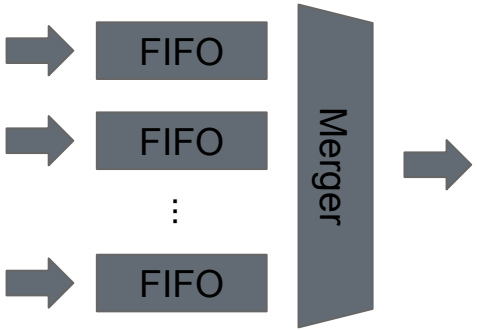
$$B = K * N * f_{clk}, K \leq 1$$

Maximum output FIFO bandwidth must be higher or equal to the sum of input FIFO bandwidth.

$$B_{out} \geq \sum_{in=0}^{S-1} B_{in}$$

Number of input streams

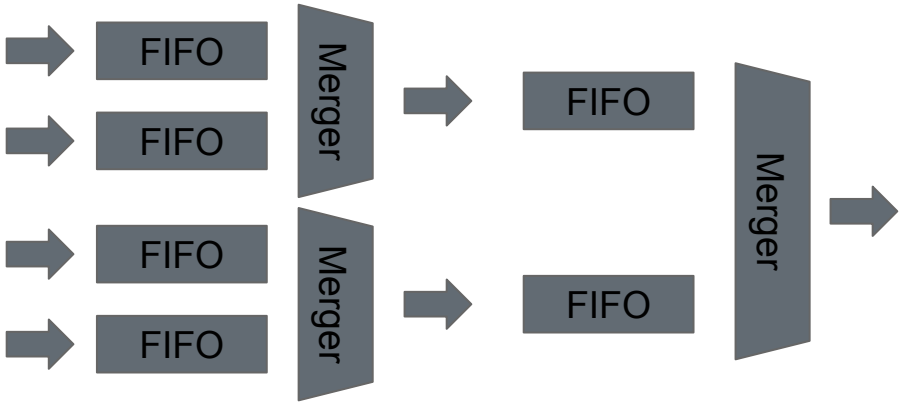
As long as bandwidth condition is fulfilled, the merger may have any number of inputs.



The more inputs, the more comparisons are required to find the oldest sample.

In order to get highest bandwidth, binary tree of two input mergers must be used.

Binary tree require more FIFOs (memory).



Bandwidth limitation

Maximum merger bandwidth

$$B_{max} = N * f_{clk}$$

Until now mergers of N=1 were considered.

In order to increase bandwidth, clock frequency had to be increased.

FPGA has clock frequency limitation, where register setup and hold time requirements cannot be fulfilled, causing latching incorrect values in registers.

In practice it is difficult to drive programmable logic blocks with clock frequency higher than 200 Mhz (actual value depend on numerous factors).

When hardware clock frequency limit is reached, increasing the number of output samples per clock cycle is only way to increase merger bandwidth..

Multiple samples per clock cycle

In some cases merging require reading one sample from each FIFO

A0 A1
B0 B1

➡ **A0 B0**
A0 B0 **A1 B1**

In some cases merging require reading two samples from one FIFO.

A1 A1
B0 B0

➡ **B0 B0**
B0 B0 **A1 A1**

In FPGA FIFO may be implemented as block or distributed memory.

It is possible to implement FIFO that allows reading/writing a variable number of samples in distributed memory, but their number is limited.

Block memory FIFOs may only be read/written with predefined number of samples per clock cycle.

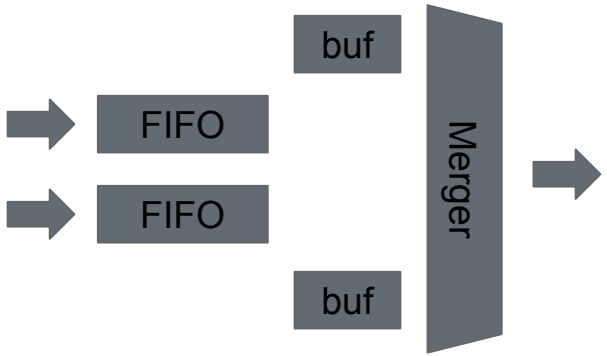
Proposed solution

Two word FIFOs are required.

Basic data buffering should be done in block memory FIFOs.

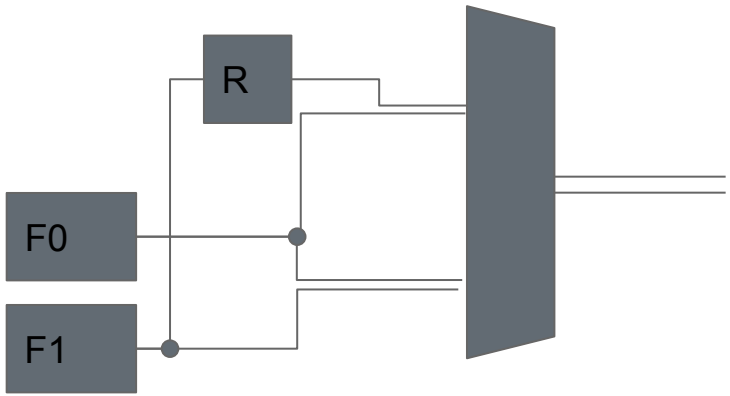
Distributed FIFO between block FIFO, and actual merger. Enables reading variable number of words.

Further investigation show that if FWFT block FIFO is used, distributed FIFO may be reduced to a register capable of storing single sample.



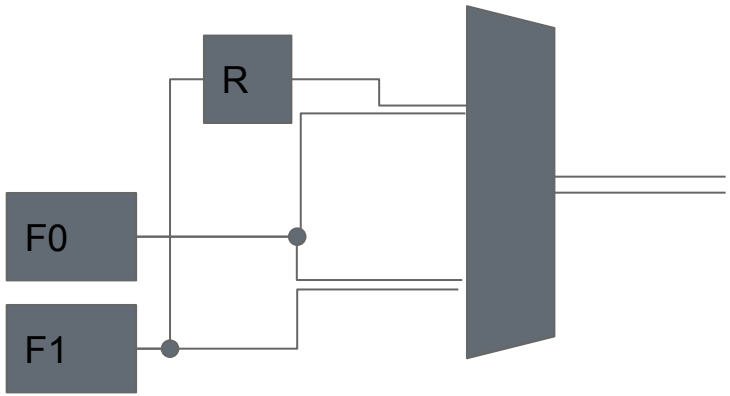
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	**A0**	R = F0 = F1 =	F0 = F1 =	Wait FIFO	
B	**B0**	R = F0 = F1 =	F0 = F1 =		



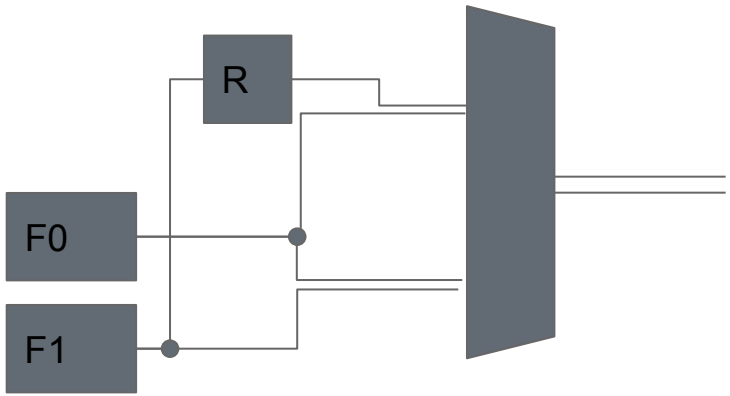
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	**A0 A2**	R = F0 = **A0** F1 = **A2**	F0 = F1 =	Wait FIFO	
B	**B0 B1**	R = F0 = **B0** F1 = **B1**	F0 = F1 =		



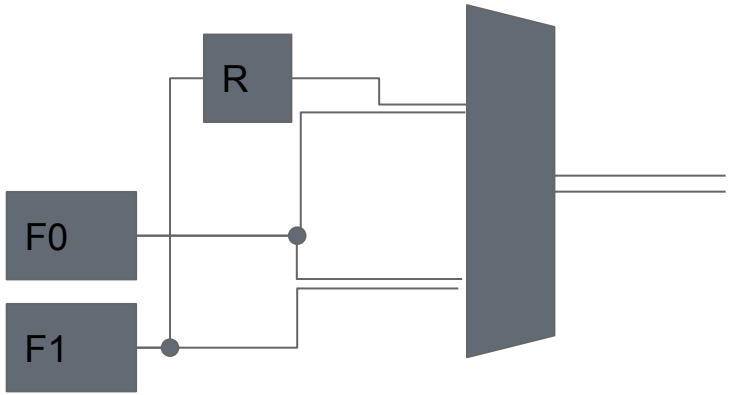
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	**A0 A2**	R = F0 = A0 F1 = A2	F0 = **A0** F1 = **A2**	**Read A,B**	
B	**B0 B1**	R = F0 = B0 F1 = B1	F0 = **B0** F1 = **B1**		



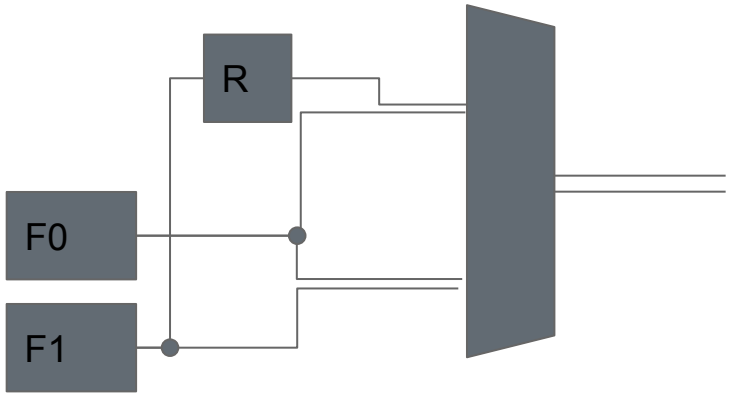
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2	R = **A2** F0 = F1 =	F0 = F1 =	Wait FIFO	**A0 B0**
B	B0 B1	R = **B1** F0 = F1 =	F0 = F1 =		



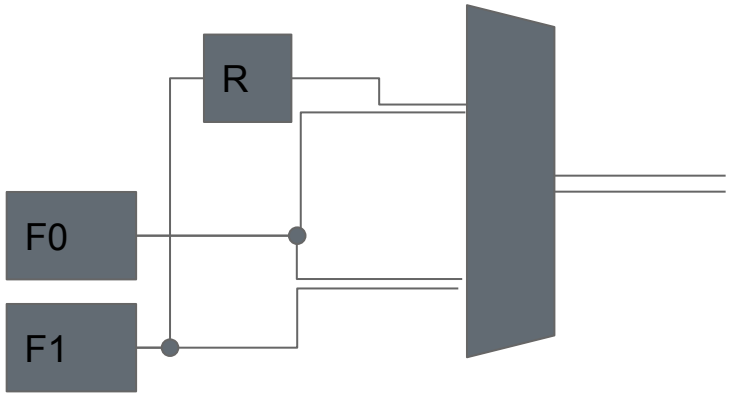
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3**	R = A2 F0 = F1 =	F0 = F1 =	Wait FIFO	A0 B0
B	B0 B1 **B1**	R = B1 F0 = F1 =	F0 = F1 =		



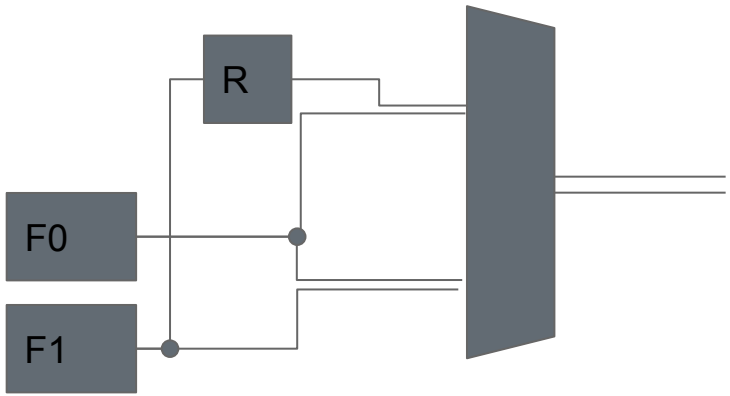
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3**	R = A2 F0 = **A3** F1 = **A3**	F0 = F1 =	Wait FIFO	A0 B0
B	B0 B1 **B1 B2**	R = B1 F0 = **B1** F1 = **B2**	F0 = F1 =		



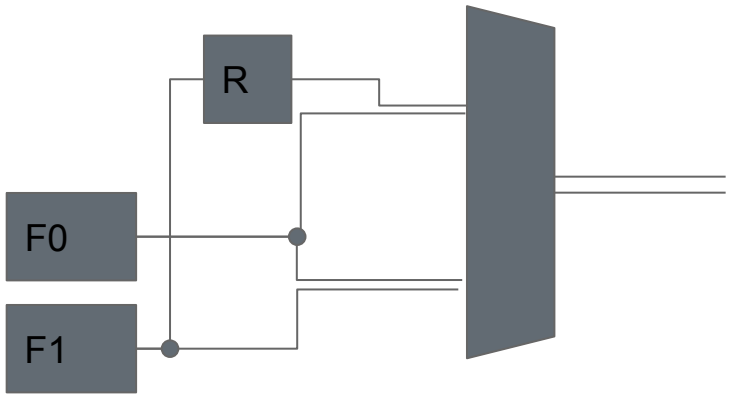
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3**	R = A2 F0 = A3 F1 = A3	F0 = **A2** F1 = **A3**	**Read B**	A0 B0
B	B0 B1 **B1 B2**	R = B1 F0 = B1 F1 = B2	F0 = **B1** F1 = **B1**		



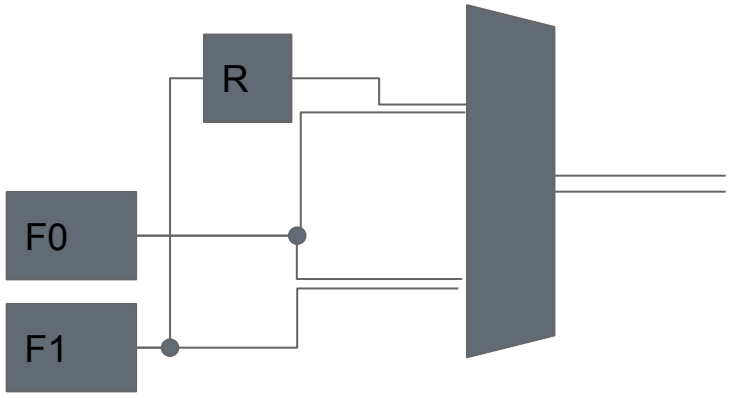
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3**	R = A2 F0 = A3 F1 = A3	F0 = A2 F1 = A3	Wait FIFO	A0 B0 **B1 B1**
B	B0 B1 B1 B2	R = **B2** F0 = F1 =	F0 = F1 =		



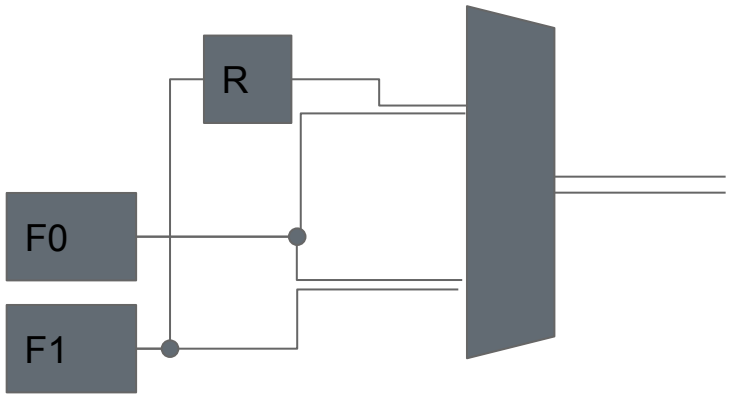
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3 A99 A99**	R = A2 F0 = A3 F1 = A3	F0 = A2 F1 = A3	Wait FIFO	A0 B0 B1 B1
B	B0 B1 B1 B2 **B99 B99**	R = B2 F0 = **B99** F1 = **B99**	F0 = F1 =		



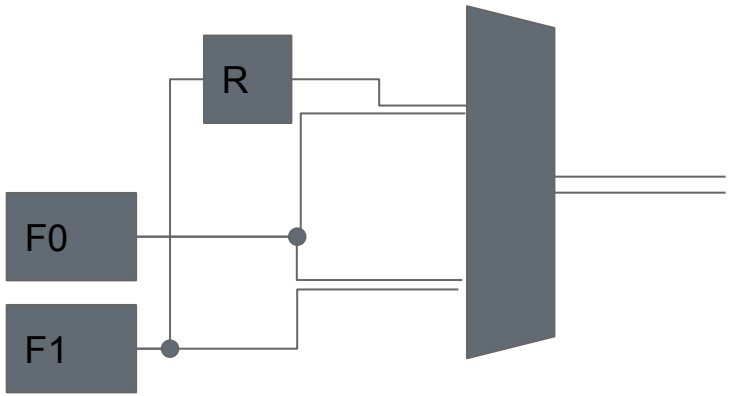
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3 A99 A99**	R = A2 F0 = A3 F1 = A3	F0 = A2 F1 = A3	Wait FIFO	A0 B0 B1 B1
B	B0 B1 B1 B2 **B99 B99**	R = B2 F0 = B99 F1 = B99	F0 = **B2** F1 = **B99**		



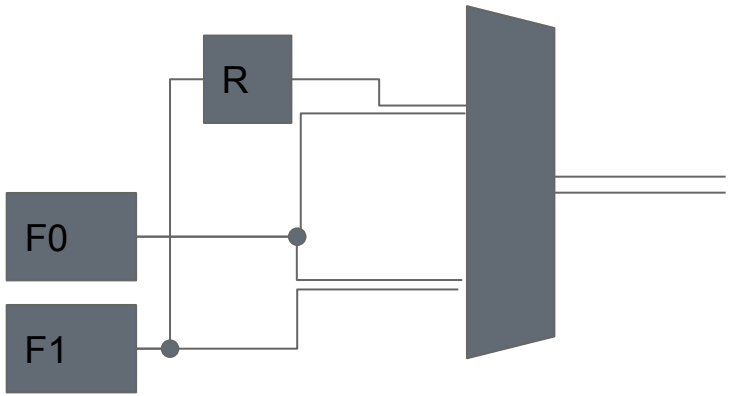
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3 A99 A99**	R = A2 F0 = A3 F1 = A3	F0 = A2 F1 = A3	**Read A,B**	A0 B0 B1 B1 **A2 B2**
B	B0 B1 B1 B2 **B99 B99**	R = B2 F0 = B99 F1 = B99	F0 = B2 F1 = B99		



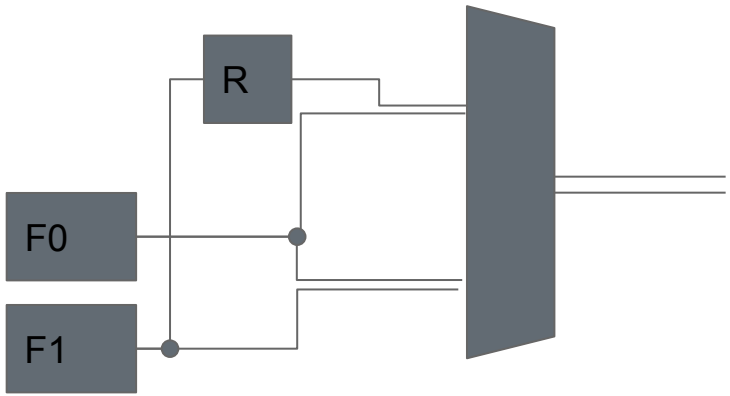
Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 **A3 A3 A99 A99**	R = F0 = A3 F1 = A3	F0 = **A3** F1 = **A3**	Wait	A0 B0 B1 B1 A2 B2
B	B0 B1 B1 B2 **B99 B99**	R = F0 = B99 F1 = B99	F0 = **B99** F1 = **B99**		



Example

	Samples	FIFO and Register content	Samples seen by merger	Operation	Output stream
A	A0 A2 A3 A3 A99 A99	R = F0 = A3 F1 = A3	F0 = A3 F1 = A3	**Read A**	A0 B0 B1 B1 A2 B2 **A3 A3**
B	B0 B1 B1 B2 **B99 B99**	R = F0 = B99 F1 = B99	F0 = B99 F1 = B99		



Simple comparison

Finding two oldest samples out of four may seem like a complicated task (at high clock frequency).

Three comparisons required by binary method may be reduced to two, when utilising the fact that input streams are sorted.

$$S[n] \geq S[n-1]$$

So in two sample FIFO

$$F0 \leq F1$$

Simple comparison

If newer of two oldest samples from stream A is older than oldest sample from stream B then both samples from stream A are older than any sample from stream B.

$$AF1 \leq BF0 \equiv$$
$$AF0 \leq AF1 \leq BF0 \leq BF1$$

Merger should read two samples from stream A.

$$F0 \leq F1$$

If the opposite

$$BF1 \leq AF0 \equiv$$
$$BF0 \leq BF1 \leq AF0 \leq AF1$$

Merger should read two samples from stream B.

If none of above is true, merger should read one sample from each stream.

Additional comparison of AF0 and BF0 is required to determine, with one is older.

Results

Presented merger was implemented in VHDL

Successful validation on random data

No timing issues on Xilinx Kintex 7 FPGA, working with 160 MHz clock.

Extras

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

Data streams must be buffered:

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

A0 **A1**
B0 **B0**

A0 B0 ..
A0 B0 **B0 A1** ..

Data streams must be buffered:

A0 **A1**
B0 B0

A0
A0 **B0** ..

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

A0 A1 **A1**
B0 B0 **B0**

A0 B0 ..
A0 B0 B0 A1 ..
A0 B0 B0 A1 **B0** **A1** ..

Data streams must be buffered:

A0 **A1 A1**
B0 **B0 B0**

A0
A0 B0 ..
A0 B0 **B0** ..

Stream merging - clock per cycle

If merger outputted two samples each clock cycle:

A0 A1 A1 **A7**
B0 B0 B0 **B5**

A0 B0 ..
A0 B0 B0 A1 ..
A0 B0 B0 A1 B0 A1 ..
A0 B0 B0 A1 B0 A1 **B5 B7** ..

Data streams must be buffered:

A0 **A1 A1 A7**
B0 B0 **B0 B5**

A0
A0 B0 ..
A0 B0 B0 ..
A0 B0 B0 **B0** ..

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

A0 A1 A1 **A7**
B0 B0 B0 **B5**

A0 B0 ..
A0 B0 B0 A1 ..
A0 B0 B0 A1 B0 A1 ..
A0 B0 B0 A1 B0 A1 **B5 B7** ..

Data streams must be buffered:

A0 **A1 A1 A7**
B0 B0 B0 **B5**

A0
A0 B0 ..
A0 B0 B0 ..
A0 B0 B0 B0 ..
A0 B0 B0 B0 **A1** ..

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

A0 A1 A1 **A7**
B0 B0 B0 **B5**

A0 B0 ..
A0 B0 B0 A1 ..
A0 B0 B0 A1 B0 A1 ..
A0 B0 B0 A1 B0 A1 **B5 B7** ..

Data streams must be buffered:

A0 A1 **A1 A7**
B0 B0 B0 **B5**

A0
A0 B0 ..
A0 B0 B0 ..
A0 B0 B0 B0 ..
A0 B0 B0 B0 A1 ..
A0 B0 B0 B0 A1 **A1** ..

Stream merging - clock per cycle

If merger outputs two samples each clock cycle:

A0 A1 A1 **A7**
B0 B0 B0 **B5**

A0 B0 ..
A0 B0 B0 A1 ..
A0 B0 B0 A1 B0 A1 ..
A0 B0 B0 A1 B0 A1 **B5 B7** ..

Data streams must be buffered:

A0 A1 A1 **A7**
B0 B0 B0 **B5**

A0
A0 B0 ..
A0 B0 B0 ..
A0 B0 B0 B0 ..
A0 B0 B0 B0 A1 ..
A0 B0 B0 B0 A1 A1 **B5 A7** ..

Waiting for two valid samples

Merger outputting a sample on each clock cycle

Merger waiting for non empty stream FIFOs

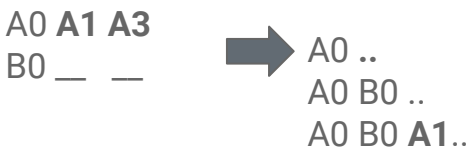
Waiting for two valid samples

Merger outputting a sample on each clock cycle

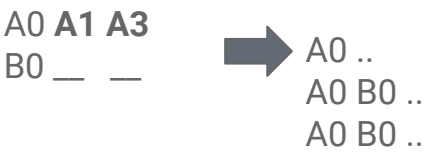
Merger waiting for non empty stream FIFOs

Waiting for two valid samples

Merger outputting a sample on each clock cycle

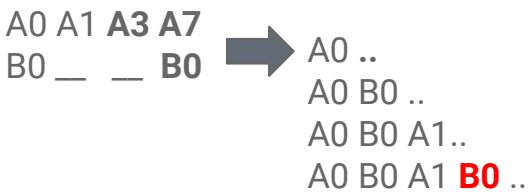


Merger waiting for non empty stream FIFOs

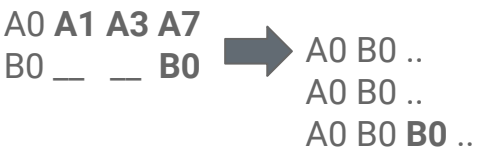


Waiting for two valid samples

Merger outputting a sample on each clock cycle

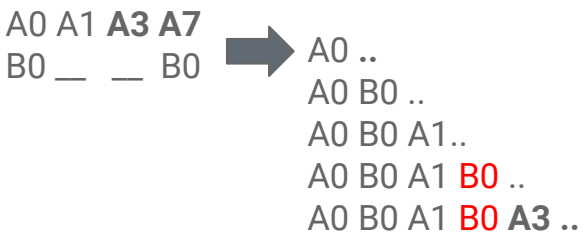


Merger waiting for non empty stream FIFOs

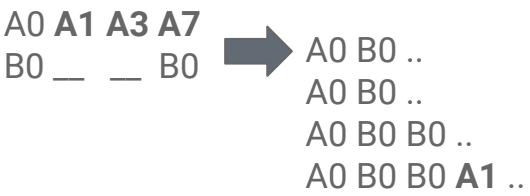


Waiting for two valid samples

Merger outputting a sample on each clock cycle

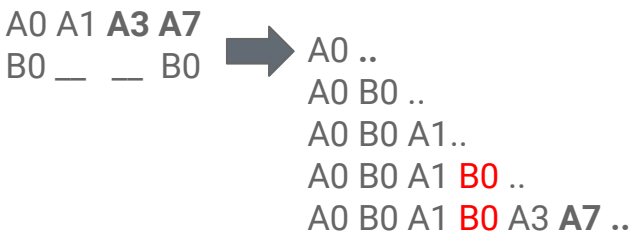


Merger waiting for non empty stream FIFOs

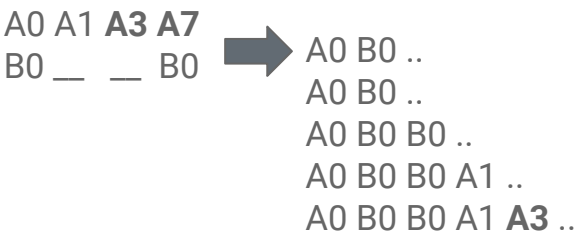


Waiting for two valid samples

Merger outputting a sample on each clock cycle

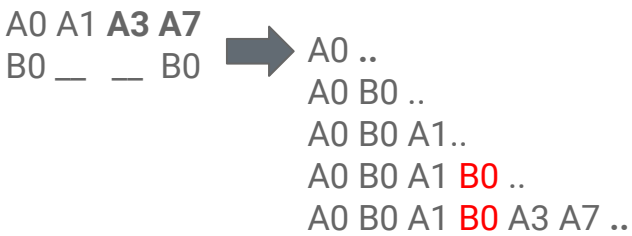


Merger waiting for non empty stream FIFOs



Waiting for two valid samples

Merger outputting a sample on each clock cycle



Merger waiting for non empty stream FIFOs