BittWare a molex company

White Paper

HBM2 performance boost for 2D FFT using FPGAs and OpenCL

Introduction

The new FPGAs with HBM2 memory can match GPU performance on far more algorithms than FPGAs without HBM2. This fact makes it possible to configure some specialized new systems with just FPGA acceleration and no GPU chips.

To demonstrate the capability of this new bandwidth, BittWare created a 2D FFT example that is traditionally memory bandwidth limited. We used the Intel FPGA SDK for OpenCL™ targeting a BittWare 520N-MX PCIe card. OpenCL is portable, but as this work illustrates, achieving peak performance requires OpenCL tailored to the target hardware.

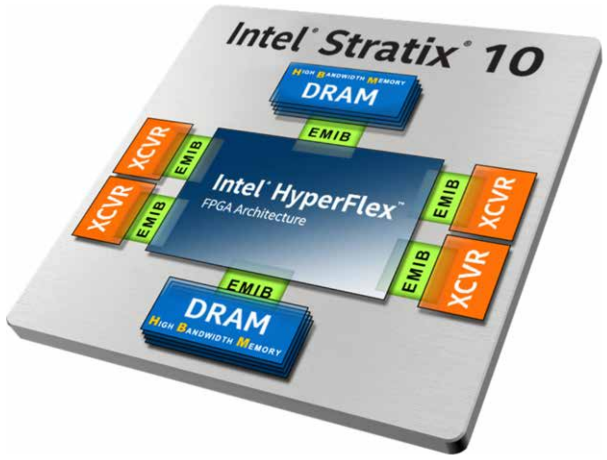

Figure 1: Stratix 10 with two HBM2 memories connected on the die

HBM2 Performance Improvement

HBM2 is a collection of stacked DDR memories accessible in parallel to give a total bandwidth that is the aggregate of the total.

The Intel® Stratix® 10 MX device used on the Bittware 520N-MX card has two HBM2 memory stacks. Each memory stack supports up to 8 independent physical channels (or 16 independent pseudo channels), with each physical channel providing 8Gbit of capacity. This translates to 8GB of capacity for each HBM2 memory stack giving a total of 16GB of HBM2 memory in the Intel® Stratix® 10 MX device. On the -2 speed grade device, the maximum bandwidth from the HBM2 devices is 409 GBytes/sec. – a 5x improvement over previous FPGA cards.

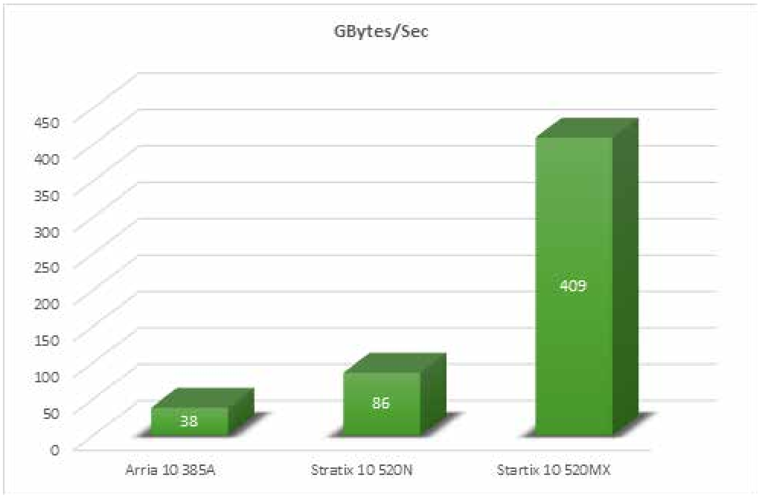

Figure 2: Comparison of memory bandwidth for previous OpenCL enabled FPGA cards

HBM2 performance boost for 2D FFT using FPGAs and OpenCL

Device Layout

The HBM2 memories are exposed directly to the user FPGA logic, with no crossbar interconnect between memory channels. Each memory is therefore treated as an independent resource to user application. This reduces latency but increases the complexity of a design if data needs to be shared between any of the 16 independent pseudo channels in an HBM2 memory stack, or if the HBM2 memory stack is to be used as one single large memory space. A total of 16 GBytes of HBM2 memory is available on the 520N-MX card.

Using HBM2 in OpenCL

The HBM2 memory is presented as 32 independent read/write ports in OpenCL. The image in the center of Figure 3 is the floor plan of a simple memory test design where data is read, modified and written back to each of the 32 HBM2 interfaces. In this example, data is shared between the memories. This achieves more than 87% of the theoretical peak memory bandwidth.

Using the OpenCL tools to program the Stratix 10 MX requires certain conditions to be met if the full potential of the MX device is to be realized:

- Ensuring the OpenCL kernel clock frequency is equal to or greater than the 400 MHz required to keep each HBM2 memory controller busy.
- Each OpenCL HBM2 port is 256 Bits wide, requiring that data types should be aligned to this width if possible.
- Finally, as is true with all DDR memory, data must be burst into the device to ensure row and column address changes do not slow the performance of the memory. For the Stratix 10 MX HBM2, this is a burst of 16, 32 Byte words.

Therefore, it is necessary to structure code in the FPGA to meet as many of these criteria as possible if the best performance is to be achieved.

2D FFT Example

A 2D FFT can be composed of multiple 1D FFTs, first applied to the rows of a 2D matrix and then on the columns. Therefore, it is logical to perform multiple 1D FFTs on multiple rows in parallel, each assigned to a different HBM2 interface.

As an example, a fully pipelined 1D FFT with 1024 complex inputs requires approximately 0.16 Bytes/flop/clock cycle.

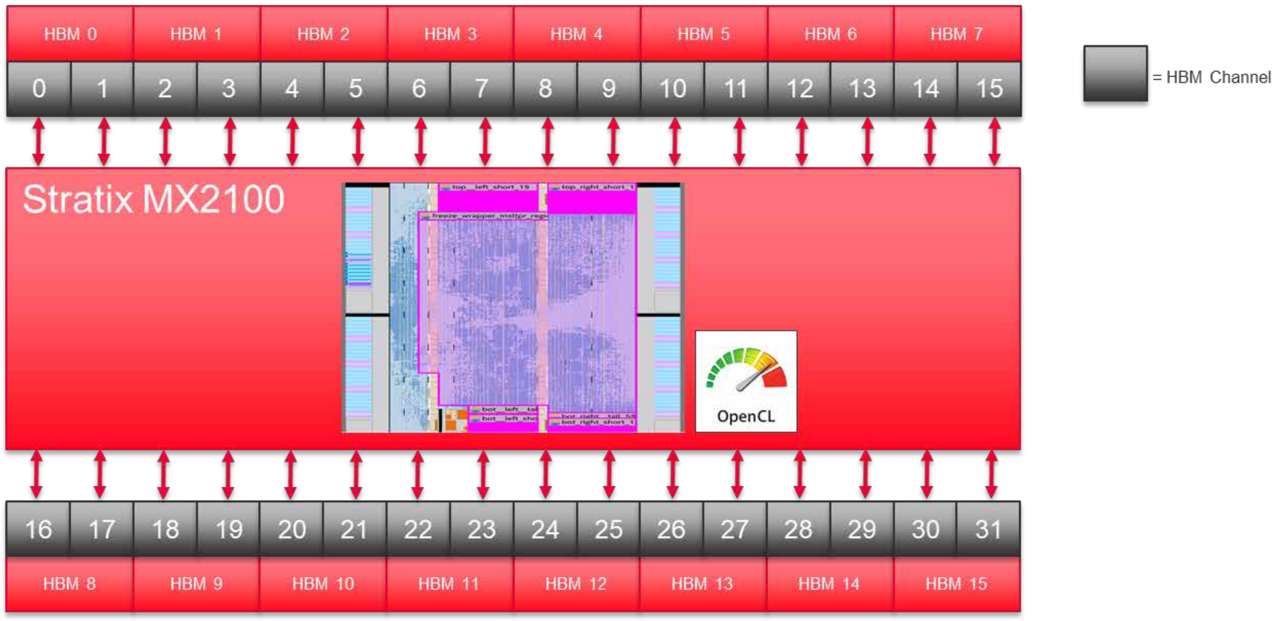

Figure 3 : Arrangement for HBM memory channels relative to FPGA fabric.

This means that for the Stratix-10 MX2100 we would require approximately 500 GByte/sec to saturate all the DSP logic on the device. Therefore, the ratio of compute to bandwidth is close to the ideal for this problem when using the HBM2 memories.

However, striping data across HBM2s causes issues for the transposition part of the calculation. At this point data must be shared between HBM2 memory channels, but there is no inbuilt crossbar on the chip to help. In this scenario, the user must manually copy data from one memory channel to another. Fortunately, FPGAs have a large amount of internal memory and register space that can be used to efficiently create sliding windows and other double buffer techniques to facilitate this. Combining both approaches allows hardware to be generated that transposes data between channels without impacting the efficiency of memory accesses.

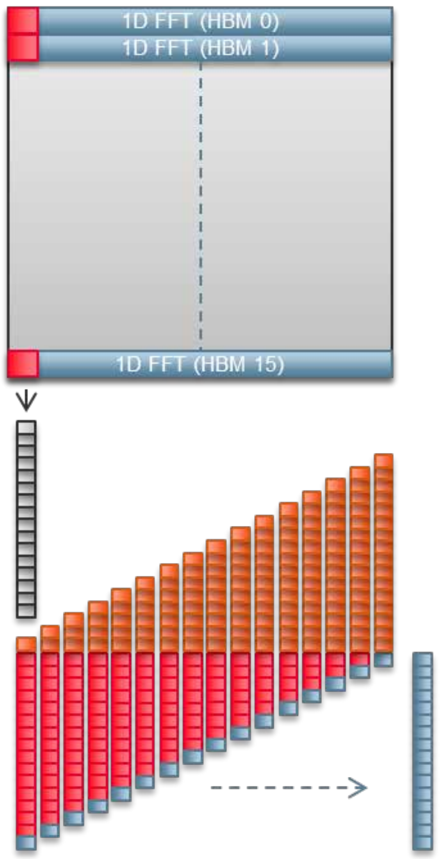

Figure 4 : Delaying output of FFTs to create a transposition in parallel to the transform calculation

Implementing a Perfect Transpose

The example illustrated here (Figure 4) delays the output from each row in 16 different delay buffers, implemented as sliding windows. After 16 clock cycles, the data is available to stripe back into the HBM2 memory transposed. There is then no need to perform an extra transpose stage as this is handled in parallel to the FFT calculation, making the transposition effectively free in terms of compute cycles.

Each of the blocks in Figure 4 represents a complex pair of words; therefore there is not enough data available to produce a full 16 word burst into the DDR memory for peak efficiency. To achieve best performance, multiple rows of 16 complex pairs are buffered until there is enough locally cached data to achieve a full burst. This can be achieved by using the M20K memories on the FPGA to buffer four rows worth of output. This is in effect 64 FFT results, which need a sizable buffer requiring 512 M20K memories (12.5% of the MX2100 device) for 1024-point 1D FFT (see Figure 5).

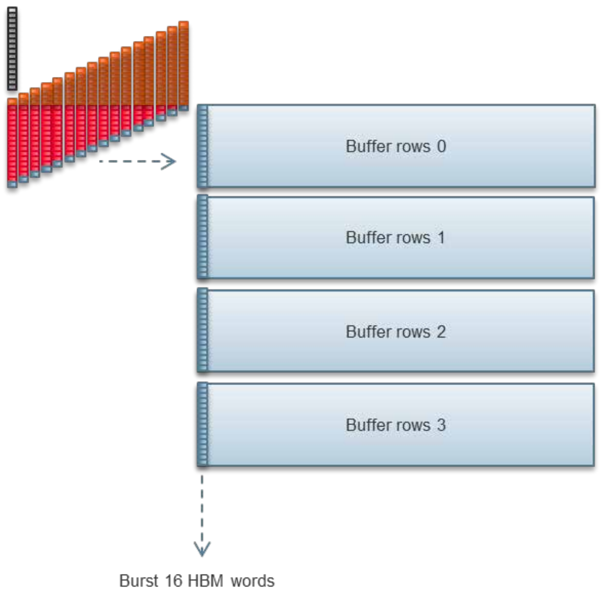

Figure 5 : Buffering multiple transpose output to achieve full burst

HBM2 performance boost for 2D FFT using FPGAs and OpenCL

Using the local memories to increase the burst length from 4 to 16 yields 60% performance boost (Figure 6). The bandwidth achieved with all the HBM2 optimizations applied is close to the theoretical peak of 409 GBytes/s.

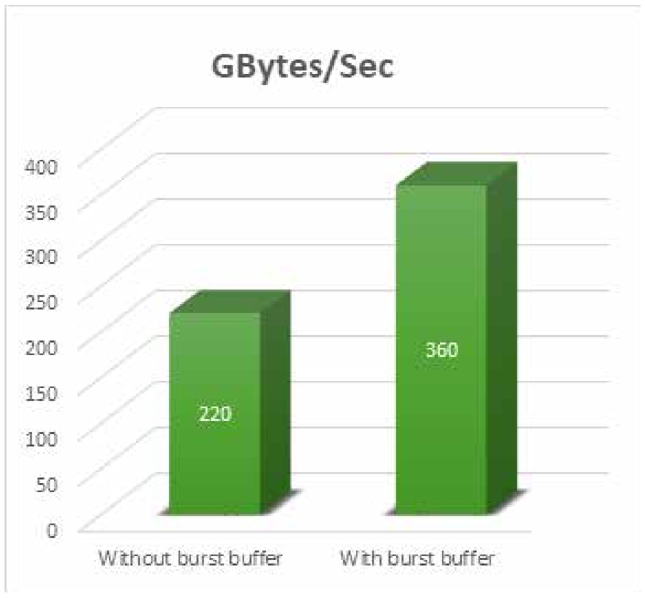

Figure 6: Performance improvement achieved by utilizing buffering

Conclusion

HBM2 memories allow FPGA logic to be fully utilised in designs that were memory bound on other FPGA platforms; however, they do require careful programming when partitioning data across the many memory ports. The good news is FPGAs have significant on-device logic that can be used to provide enough locality to solve access issues as can be seen for this 2D FFT example.

To learn more, visit the BittWare website at www.bittware.com.

Rev 2020.03.17 | March 2020

BittWare
a molex company