

SWITCHED SHARED MEMORY DATA PROCESSING SYSTEM

The present invention relates to efficient data processing within devices. For instance, image and video processing within a mobile phone. In this and other cases the processing can be split into a pipeline with data being fed in one end, processed by a number of distinct stages, and output the other end. The data may travel in various ways, for instance when processing an image, the data may appear one pixel at a time, with each pixel taking a certain time, commonly known as a clock cycle. Furthermore, when processing an image, the pixels could form a grid and a number of consecutive pixels could constitute a row of that grid, with subsequent pixels forming the row below and so on.

Within each stage of the pipeline typically one would find memories used for various purposes including that of storing portions of the data to allow computations that require access to those patches of the image. Such computations being impossible without storage of the stream of pixels because the computation also involves knowledge of the values of pixels in the rows above and below.

It is very common for a device to have a number of such pipelines for processing data of various types and for the activity of some of these pipelines to be mutually exclusive. For instance, one pipeline in a mobile phone might be dedicated to processing photographs while another is dedicated to processing video, and while it is desirable that the mobile phone can capture both photographs and video, it is never necessary to capture both at once. In another scenario the mobile phone could have particular

pipelines dedicated to particular types of photography, for example low-light or slow motion the activities of which are also mutually exclusive.

Thus, it can be seen that building the data processing pipelines in this way is wasteful of memory, because only a portion of the memories are being used at any one time. If the amount of memory could be reduced, the design would be cheaper in terms of the area of silicon required in the device's silicon chips and would require less power overall, power reduction being an important goal for designers of certain devices such as mobile phones.

10

Thus, we propose an architecture where the memory, consisting of a number of RAM blocks is logically centralised and connected to the data processing pipelines by a switching fabric which can route requests from the individual stages of the active pipeline to RAM blocks that have been pre-allocated to a particular stage during the design process. Each stage within the pipelines may generate a number of requests for RAMs, and thus can be considered to act as a number of requesters to the fabric.

15

For instance, the first stage of the first pipeline could be pre-allocated the first five RAMs, while the first and second stages are pre-allocated the first two and subsequent five respectively. A configuration selector will then select between the case when the first pipeline is active and the case where the second pipeline is active. Thus, the fabric is configured to connect the RAMs to the requesters in different patterns depending on the pipeline activity. Within a particular selected configuration, the fabric will perform switching to connect each requester to the pre-allocated RAMs belonging to that requester.

20

25

One possible operation of the fabric can be seen more clearly with an example. In this example the pipeline stage is required to perform a computation acting on a 5x5 block of pixels in the input image to produce each corresponding pixel in the output image.

5 The input pixels arrive in rows and the output pixels are generated in rows by sliding the 5x5 window across the input image. Clearly a particular output pixel requires the corresponding input pixels from five rows to have arrived and been stored to allow the computation. This can be achieved by storing the input pixels as they arrive into a RAM, with pixels being written at increasing addresses within the RAM and each RAM
10 storing one row. Once enough input pixels have arrived, we can read all five RAMs simultaneously, storing the results in a small temporary buffer, until we have a block of 5x5 and can perform the computation. This temporary buffer can be arranged so that new reads are added on one side and previous reads are shifted down so that in each clock cycle the 5x5 buffer advances 1 pixel to the right.

15

We can also see in the example that once a particular location has been read those pixel values are no longer needed and can be replaced by new input pixels. Thus, we can reuse the RAM for the next row and only require 5 RAMs to process the entire image no matter how many rows it consists of.

20

Note also in this example that two pixels are written and read from each RAM in each clock cycle, such an arrangement sometimes being necessary to achieve the required throughput of pixels given a particular clock cycle time. Of course, if two pixels are read in each cycle then it is necessary to perform two pixels calculations in each cycle,

25 thus producing two pixels to be outputted.

In an extension to the previous example, we can consider that sometimes the RAMs used will only be able to perform a single read or a single write in one clock cycle. Therefore, we cannot read from a RAM that we are also writing to. This problem can
5 be circumvented by increasing the number of RAMs by one, at the cost of an extra rows delay in producing the output. It can be seen that five reads can occur in five of the RAMs while one write occurs in the sixth.

Considering how this example could be implemented by the fabric we can see the
10 stage could implement a write request sequencer (WRS) which generates both the increasing RAM address and a bank address which selects the particular RAM being written for this row. A switching function in the fabric can then route the request's data and address to the correct RAM based on the bank address. Alternatively, the switching function can multicast the address and data to including the bank address
15 to all of the possible RAMs, each RAM being configured with a register storing its particular bank address so only one RAM, whose bank address matches the incoming bank address, will actually perform the write.

On the read side, the pipeline stage operates a read request sequencer (RRS), which
20 of course must be synchronised with the WRS. In this case no bank select logic is required and the address is simply multicast to all the RAMs within the group. Once the read has happened the data is returned through the fabric to the requester through
5 separate paths one for each RAM.

In the case where an extra RAM is used in order to emulate dual port memory, the fabric must perform an additional function, this being a switching function to choose between pairs of adjacent RAMs so that 5 RAMs out of the 6 can be selected for reading in one particular cycle. Note that in the accompanying figure only one such switching function is shown for clarity and it will be understood that switching functions exist also for the remaining read data ports.

Note that in both the previous arrangements the rows that are returned to the read data ports will be in different orders depending on the position in the image. For one position the topmost row might appear on read data port 0 whereas for another position the topmost row might appear on read data port 1. As the computation would usually require different handling of each row depending on its relative position the pipeline stage would have to include extra processing to rearrange the rows into the correct order before performing the computation. To avoid this extra complexity in the pipeline stage this rearrangement could be performed in the fabric. In this case, the fabric switching function selecting between all six RAMs in order to route the topmost RAM's read data to read data port 0. Note that in the accompanying figure only one such switching function is shown for clarity and it will be understood that switching functions exist also for the remaining read data ports.

Another possible arrangement could be used in the case where a single row does not fit in one RAM, and two are required. In this case the fabric could route the addresses to all the RAMs within one group, either A or B, and also performs a switching function to return the read data from the correct group, such routing being determined by the bank address.

A further possible arrangement would consist of RAMs of different sizes where some RAMs can fit the whole row and some cannot. In this case for the larger RAMs no switching function is required for the read data but part of the bank address must be
5 routed to the larger RAMs as part of the read address.

In some implementations we could also duplicate certain portions of the WRS and RRS within the fabric or within the RAM block to avoid transporting addresses about, such duplicates being synchronised to the WRS within the pipeline stage using signals
10 between the original and duplicate locations.

Sometimes there may be cases where multiple active pipeline stages' memory requirements sum to less than a single RAM, both in terms of memory size and memory bandwidth. In this case the stages can share a single RAM. Such sharing can
15 be done by carefully sequencing the accesses of each stage so they never request at the same time. However, this approach may be hard to achieve in practice, so the fabric may also provide an arbitration function to choose which requester is given access when two requests arrive at the same time. Given this arbitration, it will be necessary to store the unsuccessful request so that it may be fulfilled later. Such
20 storage can be achieved using a FIFO for each requester which stores the requests until they can be fulfilled. The arbitration can be made sensitive to the number of requests stored in each FIFO so that it may prevent any FIFO from overflowing.

The fabric may also manage the addressing within the RAMs using a base address for each requester to ensure that requesters only access the area of RAM allocated to them.

- 5 Another aspect that must be considered is data packing, where more than one element of data is written into each RAM location. This may be necessary to obtain enough bandwidth to the RAM or also for efficiency reasons as some RAM dimensions are more efficient than others, narrow tall RAMs being typically less efficient in terms of area used and/or power consumed.

10

For instance in image processing pipelines each pixel may be 8, 10, 12, 14 or 16 bits with different stages using different pixel sizes. For example, if a stage is using 14-bit pixels we may decide to store these in a 28-bit wide RAM with two pixels stored in each round word. One possible way of achieving this is for the requester and for all 28
15 bits to be transferred across the fabric at once. Another way is for the requester to send single pixels across the fabric and for logic in the RAM block to pack each pair of pixels into a RAM word. In this case it is fairly easy for the requester to always send an even number of pixels so that the RAM always has enough pixels to finish writing a word.

20

Of course, the two techniques may also be used in tandem with some packing happening at the requester and further packing happening at the RAM block.

A further complication arises when we allow the RAM blocks to pack data words, such as pixels, into RAM words where the RAM size is not an exact multiple of the data
25 word. In order to efficiently pack the data into the RAM words we have to accumulate

one or more data words in a buffer until we have enough data to write a whole RAM word. Of course, this packing approach only works if the data is written in a sequential manner. If the requester is aware of the size the RAM, and the sizes are chosen carefully, the requester can ensure that it always writes a number of pixels that correspond to a whole number of RAM words. If however the requester is not aware of the size of the RAM words, it is possible to have an extra signal that tells the RAM to write a partial word at the end of a sequential batch of writes. Alternatively, the write can be triggered by the arrival of an out of sequence write address. Reads may be non-sequential provided the frequency of reads allows time for the extra accesses to be done when data words span two RAM words. Sequential reads are handled by loading the RAM word into a temporary buffer and shifting out the data words one by one.

The switching fabric logically consists of a set of cross-points that serve to selectively connect the inputs to the outputs, a structure known as a crossbar. In practice the crossbar is often implemented using a multiplexor for each output which chooses which input the output is connected to. In the present invention it may be advantageous to build a big fabric out of a number of smaller fabrics. One possible arrangement consists of 3 layers of fabrics and is known as a Clos network.

As well as connecting the pipeline stages to the RAMs the fabric may also connect other functions. For instance, if a RAM is used as a lookup table containing values that map from a pixel value used as the address to a preconfigured number stored in the RAM then there must be a path for some external entity such as a CPU to write the preconfigured data into the Ram. It also may be advantageous to provide read access

to all RAMs from a CPU for debugging purposes. A further function maybe to route connections from built-in self-test units (BIST) to groups of RAMs for use during manufacturing testing.

- 5 Considering the physical layout of the various circuit elements that make up the pipeline stages, the fabric and RAMs, it may be advantageous to split the fabric into a number of stages and the RAMs into several groups. One such arrangement is shown in the figure where the fabric is split into three stages, the RAMs are partitioned into four groups, the image processing pipelines also being partitioned into four groups
10 and said groups being distributed around the central switching stage of the fabric.

Figure 1 shows Image Processing Pipelines

- Image processing blocks with internal memory
- Multiple pipelines used one at a time
- 15 ➤ Waste of memory

Figure 2 shows centralise memory and share through fabric

- Fabric can theoretically connect any Requester to any memory. In practice we can prune the fabric to only contain needed connections

Figure 3 shows example connections

- 20 ➤ Requester 1 uses 5 RAMs

Figure 4 shows example active connections

- Image processing pipeline 1
- Image processing pipeline 2

Figure 5 shows example Image Processing block

- 25 ➤ Pixels arrive in sequentially in rows

- Use 5x5 input pixels to calculate each output pixel

Figure 6 shows example memory write/read pattern

- Write pixels in pairs, one memory for each row
- When 5 rows have been stored can start reading all five memories and load into buffer to do 5x5 calculation

5

- Reads of 2 pixels from each row -> 2 output pixels calculated

Figure 7 shows dual port emulation

- Extra RAM so don't need to read and write to same memory in one cycle

Figure 8 shows Requester 1 Write path

10

- Requester Write Sequencer generates write addresses
- Sequencer can be duplicated in Fabric and/or in RAM to save wiring
- Switch can unicast to selected RAM, or multicast to whole RAM group (including bank address, so RAM knows if write is for it)

Figure 9 shows Requester 1 Read path

15

- Requester Read Sequencer generates read addresses
- RRS must be synchronized with RWS
- Switch multicasts address to all RAMs

Figure 10 shows Requester 1 Read path dual port emulation

- Switch multicasts address to all five RAMs being read

20

- One switch for each Read Data port selects a subset of the RAM read data to return

Figure 11 shows Requester 1 Read path row rearrangement

- To make the computation easier the fabric can rearrange the rows so that the rows appear in order from top to bottom at the Read Data ports

25 Figure 12 shows Read path if row takes 2 RAMs

- Switch multicasts address to all RAMs in either A or B group

Figure 13 shows Read path heterogeneous RAMs

For larger RAMs:

- Some Bank address bit are added to RAM address
- 5 ➤ Read data path is different

Figure 14 shows Dynamic RAM sharing

- Requesters combined is less than total RAM size and bandwidth
 - FIFOs mean no sync needed
 - Arbiter controls access to shared RAM
- 10 ➤ Fabric can apply offset to guarantee no overlapping

Figure 15 shows Pixel Packing at Requester

- Pixels can be different sizes e.g., 8,10,12,14 or 16 bits

Figure 16 shows Pixel Packing at RAM

- Requester is aware of packing, so sends pairs of pixels
- 15 ➤ Fabric ports and Memory widths can be heterogeneous

Figure 17 shows Pixel Packing across words

Figure 18 shows Switch Fabrics

Figure 19 shows other accesses

- RAMs may act as lookup tables; data needs to be written by CPU (only useful if lookup shared)
- 20 ➤ Read access may be needed for debugging (also possibly write)
- Self-test units (BIST) may access RAMs across the fabric

Figure 20 shows Multistage fabric for efficient layout.

23E0154

Claims

1. A system, apparatus or method substantially as herein described as shown in Figures 1 to 20.

Abstract

SWITCHED SHARED MEMORY DATA PROCESSING SYSTEM

5 The present invention relates to efficient data processing within devices. For instance,
image and video processing within a mobile phone. In this and other cases the
processing can be split into a pipeline with data being fed in one end, processed by a
number of distinct stages, and output the other end. The data may travel in various
ways, for instance when processing an image, the data may appear one pixel at a
10 time, with each pixel taking a certain time, commonly known as a clock cycle.
Furthermore, when processing an image, the pixels could form a grid and a number
of consecutive pixels could constitute a row of that grid, with subsequent pixels forming
the row below and so on.

15 (Figure 1)