"Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA"
Abstract

One of the main problems for underwater terrain navigation according to the correlation method is the processing time. When the algorithm is used on a regular PC the processing time is far too long for real-time applications. The Swedish navy is interested in a non-revealing terrain navigation system that can perform in real-time.

This thesis examines some reports on different navigation methods. It will make conclusions on which method is the best suited for this application.

In the second part of the report an implementation of a navigation system into a hardware accelerator (in this case a FPGA) is described. The calculations have 16 bits resolution and the use of a FPGA gives parallelism and shortens the computing time considerably. The actual computing time has been decreased with over 100 times compared with similar application on a PC and the investigation show that even faster calculation speeds can be reached. The implementation is downloaded into a Virtex®-II Xilinx device and has been written in VHDL.
Acknowledgments

Several people have been helpful in making this report possible. Without them this thesis could not have been done. We want to direct a special thanks to all of them.

A big thank you to

our supervisors Carl-Johan Andersson at FMV
and Ingemar Nygren at FOI
our examiner Lennart Lindh at Mälardalens University
Johnny Holmberg at Mälardalens University
Magnus Lindblad at Xilinx Sweden AB
And a big special thanks to Stefan Stjernen at Realfast who has been a big support in our work.
# List of contents

1. Introduction .......................................................................................................................... 5  
   1.1. Background .................................................................................................................... 5  
   1.2. Earlier Work ................................................................................................................. 6  
   1.3. Thesis objective ............................................................................................................ 6  

2. Report description ............................................................................................................. 7  

PART I - INVESTIGATION ............................................................................................... 8  

3. Different navigation methods ......................................................................................... 8  
   3.1. Terrain correlation navigation method ......................................................................... 8  
   3.2. Recursive underwater navigation .............................................................................. 9  
   3.3. Image recognition of artificial underwater landmarks ............................................. 10  
   3.4. Image analysis ........................................................................................................... 11  
   3.5. Comparison between the methods ............................................................................ 11  
   3.6. Conclusions ............................................................................................................... 12  

4. Implementation methods ............................................................................................... 13  
   4.1. Problem definition ..................................................................................................... 13  
      4.1.1. Correlation method ............................................................................................ 13  
      4.1.2. Data definition .................................................................................................. 14  
      4.1.3. Future performance in FPGA ........................................................................... 14  
   4.2. Algorithm calculation ............................................................................................... 15  
   4.3. Conventional FPGA design ...................................................................................... 16  
      4.3.1. Memory method ............................................................................................... 16  
      4.3.2. Download method ............................................................................................. 17  
   4.4. Xilinx System Generator™ V1.1 ............................................................................. 18  
   4.5. Conclusions ............................................................................................................... 20  

5. Summary of the investigation ....................................................................................... 21  

PART II - REALISATION OF THE METHOD ............................................................. 22  

6. Examined case, test card and tools ............................................................................ 22  
   6.1. Examined case ........................................................................................................... 22  
   6.2. Test card .................................................................................................................... 22  
   6.3. Design tools ............................................................................................................. 23  

7. Architecture design ....................................................................................................... 24  
   7.1. The Correlator (Top design) .................................................................................... 24  
      7.1.1. Execution flow .................................................................................................... 24  
      7.1.2. Area and timing summary ............................................................................... 25  
   7.2. HOST ....................................................................................................................... 25  
      7.2.1. Block function .................................................................................................... 25  
      7.2.2. Execution flow ................................................................................................. 27  
      7.2.3. Area and timing summary ............................................................................... 28  
   7.3. MEM_COM ............................................................................................................. 29  
      7.3.1. Block function .................................................................................................... 29
7.3.2. Execution flow ....................................................................................................... 30
7.3.3. Area and timing summary ...................................................................................... 30
7.4. FIFO1 and FIFO2 ......................................................................................................... 31
7.4.1. Block function ........................................................................................................ 31
7.5. CORR ............................................................................................................................ 32
7.5.1. Block function ........................................................................................................ 32
7.5.2. Execution flow ....................................................................................................... 33
7.5.3. Area and timing summary ...................................................................................... 33
7.6. SEND ............................................................................................................................ 34
7.6.1. Block function ........................................................................................................ 34
7.6.2. Execution flow ....................................................................................................... 35
7.6.3. Area and timing summary ...................................................................................... 35
7.7. SDRAM_CONTROLLER ............................................................................................ 36
7.7.1. Block description .................................................................................................... 36
7.7.2. Area and timing summary ...................................................................................... 36

8. Protocols .............................................................................................................................. 37
8.1. Physical protocol ........................................................................................................... 37
8.1.1. Data Write ..............................................................................................................38
8.1.2. Data read ............................................................................................................... 39
8.2. Sending protocol ........................................................................................................... 40
8.2.1. Reset ....................................................................................................................... 41
8.2.2. Send map and mean data ........................................................................................ 41
8.2.3. Send sonar data ....................................................................................................... 42
8.2.4. Timing issues .......................................................................................................... 43

9. Loading/Sending sequence and data structure ................................................................ 44
9.1. Sending sonar ................................................................................................................ 44
9.2. Sending map and mean values ...................................................................................... 44
9.2.1. Mean data from the map ........................................................................................ 44
9.2.2. Data sequence ......................................................................................................... 45
9.3. Sending results .............................................................................................................. 49
9.4. Using multiple correlators .......................................................................................... 50

10. Results ............................................................................................................................... 51

11. Summary ........................................................................................................................... 52

12. Future work ...................................................................................................................... 53

13. List of references .............................................................................................................. 54
1. Introduction

1.1. Background

The Swedish navy needs a positioning system for underwater vehicles that has high precision, fast computation and is hard too detect. The Global Positioning System (GPS) has the disadvantage that it does not work under water, since the high frequency GPS signals cannot reach beneath the surface. This means that the underwater vehicle must surface to get a position, which would make it easy to detect. Also according to the recommendations of the Swedish navy the GPS is not to be used without a backup system during military operations. Because of this, the possibility of using a terrain navigation systems is examined.

There are several different terrain navigation methods. On land an image of the terrain can be obtained and matched with a map by using for example a camera and image processing. Underwater however, this method has big limitations because of poor visibility in most waters and is highly range limited. Therefore sonar is commonly used under water. Unfortunately, the sonar transmits acoustic pulses that can be detected by other crafts or fixed installations. For that reason the underwater vehicle cannot send out “sonar-pings” for determining its position very often. Under military operations it can be several hours between two “pings”, in between the “pings” an Inertial Navigation System (INS) is used. The INS, which is the main navigation system, is then updated with the new position given by the sonar. When updated, it is crucial that the new position has a high accuracy since the error in the update will propagate in the INS. A very high accuracy in the positioning is therefore a necessity.

The basic idea for terrain navigation is to compare measurements of the sea bottom with the bottom charts over that area. Two different principles on how to conduct this comparison are presented in this report. The first one, the terrain correlation navigation method, compares the bottom profile of a large area with a bathymetric map to get the correct position. The other method, recursive underwater navigation, compares only one measurement with the map. This generates a number of different possible positions, which are then weighted against a predicted position based on the earlier position, hence the name recursive navigation.

In addition two other systems, not based on terrain navigation, is presented to provide a wider view of the underwater navigation problem.

Since the error in the INS is built up over time, the search area has at least to be of the same size as the uncertainty of the INS. Therefore long times between INS-update will mean large areas to search for the correct position. This comparison, when using the correlation principle, will therefore take very long time for a regular PC to compute. In real time applications this is not acceptable. An earlier study has indicated that the calculation time can be greatly reduced by using parallel computing, for example an implementation in an Field Programmable Gate Array (FPGA) [1]. Different implementation methods are therefore also presented in this investigation.

In the second part of the report a hardware implementation of a terrain correlation system is presented. The system is used to evaluate the usage of a hardware accelerator for correlation calculations. For the implementation a Xilinx Virtex® –II FPGA has been used and the architecture is described in the hardware language VHDL. The design and its applications are described in detail and results from different correlation tests are presented.
1.2. Earlier Work

There are some earlier theses in this area. They have examined which systems are appropriate for underwater navigation [2]. An older work from 1989 [3] looks at the use of a terrain matching navigation system in autonomous underwater vehicle. Another work has gone deeper on how to combine the terrain matching system with the inertial system [4].

1.3. Thesis objective

The objective of this thesis is to review different terrain navigation systems and to design and implement a fast correlation machine on a FPGA.
2. Report description

The problem today with the correlation method for terrain navigation when using a large search and sonar map is the computation time. The main objective of this report is to find the best way of reducing that computation time. Therefore a comparison is made between different terrain navigation methods and implementation methods. The goal is to find a method that is suitable for implementation in hardware, since this is assumed to be the only realistic way of calculating the algorithms in a reasonable time [1] [10].

The comparison is made according to the needs and conditions of Swedish underwater operations. Conclusions are therefore made according to those needs and conditions.

This paper is organised as follows:
Part I – Investigation of the use of hardware accelerators for correlation algorithms. Section 6 describes different underwater navigation methods and discusses their advantages and disadvantages. Section 7 describes different implementation methods for the terrain correlation method and section 8 summaries the investigation.

Part II – Implementation of a hardware accelerator for calculation of correlation algorithms. Section 9 describes the tools and hardware used to implement the hardware accelerator. Section 10 describes the design of the accelerator. Section 11 describes the protocol and commands used for communication with the accelerator and section 12 describes the sequence in which data is transmitted. In section 13 the results from the tests conducted on the accelerator can be found. Finally section 14 summarizes the report with some concluding remarks and section 15 discusses directions for future work.
PART I - Investigation

Part one of the report is an investigation of different terrain navigation methods and the possibility of implementing such a system in a hardware accelerator.

3. Different navigation methods

3.1. Terrain correlation navigation method

The correlation method [1] [4] [5] is based on a correlation system which compares a high resolution and depth accuracy bathymetric map with a significantly smaller depth map of the bottom of the sea measured by sonar.

The depth values, which are measured with a multibeam sonar, has an error because the sound velocity changes when the water temperature change, which it does depending of the depth. The velocity can also change because of the salinity. To get the signal right the samples has to be compensated for these errors. This compensation is provided by the sonar equipment and does not need to be calculated by the user.

This method does not need the actual sea depth only the depth related to for example the mean value point of the sonar map since it is the bottom profile, which is correlated against the maps profile.

The authors of this report have taken the detecting risk when using sonar under consideration. Therefore the use of the sonar will be very seldom. The navigation problem that comes with this is solved with an inertial navigation system. The INS is much more complex and has much smaller error factor than a dead reckoning system. This also solves another problem, which is the lack of bathymetric map areas. The INS is therefore updated in special navigation areas, placed at suitable distance from each other, where high-resolution bathymetric maps are
available. However, because of the error in the INS the search area has at least to be of the same size as the uncertainty of the INS, i.e. longer time between the measurements results in larger search areas. This will greatly effect the computation time.

The basic idea of matching system with correlations is to compare the measured terrain profile with a same size terrain profile from a map. The absolute sum of all the pixels difference in the map makes a scale, were the smallest value is the most likely position on the search area. Dividing it with the number of sonar beams then normalizes the correlation sum. To get a more user-friendly view of the result the correlation sums are inverted, which results in peaks showing the possible positions.

The equation is

\[ e = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \beta_{m,n} |e_{m,n}| \]

were \( \beta_{m,n} \) is an compensation factor.

According to the authors it is possible to get a good correlation result with very good accuracy and low variance using only a 5 x 5 pixels sonar map. This means that the sonar does not have to be so expensive because of the small number of beams that needs to be sent out.

### 3.2. Recursive underwater navigation

This thesis [7] discusses an underwater positioning method, which combines some parts of different navigation systems. The method is called LOST2. The LOST2 method uses dead reckoning, acoustic-based, and terrain-based positioning in an integrated system. The system needs a high-resolution bathymetric map, the range to a known position, the estimated velocity of the vehicle and the depth of the sea. The system combines all this information with an estimated position from an earlier calculation, hence the name recursive navigation. The system is built on two subsystems that together estimate a position.

The first subsystem is based on two modules. The first module is a dead reckoning system, which by integrating the velocity with respect to time produces a predicted position. The second one is a terrain-matching module that generates an estimated position using the measured sea depth and the bathymetric map together with the predicted location from the first module. The second subsystem uses a Kalman filter that combines the two estimations to get an optimal linear combination of the two inputs. This result together with a slant range to a known point gives the final estimated location of the vehicle. There is also a state predictor module that tracks the unmeasured velocities.
The University of Tokyo has developed the method in [10]. The basic idea of this method is to navigate using dead reckoning and after a while find an artificial underwater landmark (AUL) to update the position given by the dead reckoning system. According to this paper the position error in their dead reckoning system is about 0.4 % of the distance travelled. If the position is corrected by an AUL the distance between the AUL’s cannot be too far because of the error in the dead reckoning system and the limitation in sight for the camera. The AUL is designed with two-meter long cables that show which way the AUV should go to find the exact point. On the exact point there is a positioning tool called Signboard system for Heading angle Reference (SHR) that contains information that the AUV needs to correct the dead reckoning error, it also contains information for heading angle reference correction. To get this information the AUV must be in an exact position, this is made with the SHR that has four coloured balls placed in a geometrical shape.

The author is using this method for AUV’s in shallow water were the visibility is about 1 meter, for example to inspect a large storage tanks.

This method has clear similarities with a missile guidance method, using recursive navigation, developed in the seventies. There are different versions of the method and interested readers might look at [8] [9] which describes the systems TERPROM and SITAN.

### 3.3. Image recognition of artificial underwater landmarks

The University of Tokyo has developed the method in [10]. The basic idea of this method is to navigate using dead reckoning and after a while find an artificial underwater landmark (AUL) to update the position given by the dead reckoning system. According to this paper the position error in their dead reckoning system is about 0.4 % of the distance travelled. If the position is corrected by an AUL the distance between the AUL’s cannot be too far because of the error in the dead reckoning system and the limitation in sight for the camera. The AUL is designed with two-meter long cables that show which way the AUV should go to find the exact point. On the exact point there is a positioning tool called Signboard system for Heading angle Reference (SHR) that contains information that the AUV needs to correct the dead reckoning error, it also contains information for heading angle reference correction. To get this information the AUV must be in an exact position, this is made with the SHR that has four coloured balls placed in a geometrical shape.

The author is using this method for AUV’s in shallow water were the visibility is about 1 meter, for example to inspect a large storage tanks.
3.4. Image analysis

The authors of this report [11] suggest using a Pulse Coupled Neural Network (PCNN) in image analysis applications such as airborne reconnaissance and missile navigation. The PCNN is an algorithm that is biologically inspired and has clear similarities with the vision system. The PCNN is suggested to be integrated with the Digital Scene Matching Area Correlator (DSMAC) navigation system that is used in missile target finding systems. The DSMAC-system locates its targets by correlating an image taken by the missile with a stored image of the target. The two images are first made binary and then correlated. Because of the influence of, for example atmospheric changes and sensor quantization effects in the images, there may be smaller or larger differences between the images, which will affect the correlation result. The authors’ solution to this problem is to use edge detection or object segmentation on the images. This would make the correlation more accurate without damaging the image. The PCNN, which can perform both edge detecting and object segmentation, is therefore suitable to be integrated in the DSMAC. The report also discusses implementation of image processing algorithms. The conclusion made, is that in order for the navigation system to work the calculations has to be done in relative real time and that implementation in hardware probably is the only way of receiving that. Therefore solutions that are suggested should be possible to implement in hardware, which is possible with the PCNN.

Figure 7. The figure shows a JAS 39 gripen. The first picture is the picture taken by the camera. The second is binary image and the third is the edge detected image.

3.5. Comparison between the methods

The different theories presented above shows that there are many different ways to approach the navigation problem. However, most of the theories are developed for a predetermined area of use and are therefore not always compatible for all applications.

The terrain matching navigation method described in 3.1, which is based on correlation between a sonar image and a bathymetric sea map, is developed for the Swedish Navy and is therefore very compatible with the requirements of Swedish UV’s. It has the advantage of being able to find a correct position without knowing at what depth the vehicle is. This is a big advantage since there can be difficulties knowing under which conditions the map was made. For example: if the operation area is in tidal water there can be differences in the sea depth between the map and the sonar image, which would result in poor correlation. The compatibility between the map and the sonar image is however still a problem because of the many parameters that affect a sonar image. The method also has the advantage/disadvantage of using sonar instead of cameras, advantage because of the superior “visibility” and disadvantage because of the detection risk. The system has due to the large sonar map a very
high precision. It is of great importance that the tails of the position error variance distribution is very short if the estimated position is going to be used for update of the INS. An inaccurate position given by the system could, obviously, have severe consequences.

The advantage of the method described in 3.2 is that it requires significantly less off board hardware than other methods since it uses much of the already existing measurement from the dead-reckoning system. This method however, needs to measure at what depth the vehicle is. That gives it a disadvantage according to the reasoning above. The method also has lower accuracy and higher variance than the method above. The positions error is up to six times bigger, with respect to the resolution of the map. Another disadvantage is the number of measurements needed. The method is based on repeated measurements throughout the entire mission. With measurement in say every 50 meters (depends on the velocity) the detection risk will be greatly increased compared to the correlation method. Furthermore the repeated measurement requires maps covering the entire operation area. The correlation method only needs maps covering a few percent of the operation area because of the use of navigation areas.

The method discussed in 3.3 has a very different approach to the problem because of the use of cameras instead of sonar and of artificial underwater landmarks instead of bathymetric maps. The Swedish Navy, because of their requirements, cannot use this method. The problem is that the cameras have poor visibility (about 1 meter) and that the range is limited because of the AUL’s. This is not a surprise because it was developed for other purposes. If you take these reasons into account the method has some great features. The use of camera instead of sonar significantly reduces the amount of hardware needed, which makes it possible to build the vehicle small. When updating the position the calculation time is greatly reduced, because of the use of landmarks instead of maps.

The PCNN system described in 3.4 has the same disadvantage with the camera as the method above. The interesting thing with this method is the use of edge detecting. The use of edge detecting could be applicable in underwater navigation. But the problem with the difference between two images of the same area is much smaller when using sonar. Therefore the edge-detecting algorithm probably would not improve the result of the correlation in an underwater environment. One might also think that a regular correlation, in waters with flat bottom, would be superior to the edge detecting correlation.

### 3.6. Conclusions

The conclusion of this investigation is that the best method for underwater terrain navigation is the terrain matching method present in 3.1, which uses traditional correlation with some modifications. All methods are possible to implement in hardware but only this one fulfils the Swedish Navy requirements.
4. Implementation methods

4.1. Problem definition
The implementation methods that were examined are all for an implementation into an FPGA.

4.1.1. Correlation method
There is one basic algorithm that the authors’ in 3.1 are using. They use a few different versions of it but for this purpose the basic one is the most suited. The algorithm is:

\[ e = \frac{1}{M \cdot N} \sum_{m=1}^{M} \sum_{n=1}^{N} \beta_{m,n} \cdot |e_{m,n}| = \frac{1}{M \cdot N} \sum_{m=1}^{M} \sum_{n=1}^{N} \beta_{m,n} \cdot |(x_{m,n} - \overline{x}) - (y_{m,n} - \overline{y})|, \]

where \( \{x\} \) is the sonar matrix and \( \overline{x} \) is the mean value of it. \( \{y\} \) is a part from the map and \( \overline{y} \) is the mean value of it. This method needs a map and a sonar image. To compute the correlation, the absolute sum of the pixel value differences of the sonar image and the corresponding part of the map is calculated. This is made for every correlation area. The area with the lowest sum then becomes the likely position.

![Figure 8. Correlation principle. The green area in the map is the area that is compared with the sonar image i.e. the correlation area. This comparison is made throughout the entire map.](image)

To make this even easier a mean value of the part from the map and a mean value from the sonar are computed and subtracted from the sonar pixels and the map pixels respectively. This so the correlation does not need the depth. This is a major advantage since the sea level can differ from the reference level of the map. To prove this one can use the version of the algorithm when the square error-sum is calculated instead of the absolute error-sum.

The difference between the pixel values can be written as follows (the map and the sonar image is now represented by a vector):

\[ e_i = y_i - (x_{s,i} + x_u + z) = y_i - (x_{s,i} + x_u) - z \]

Where \( e \) is the difference. \( y \) is the depth from the map. \( x_s \) is the sonar measured depth. \( x_u \) is the constant depth of the vehicle and \( z \) is the difference between the reference level of the map and the actual level of the map.

The algorithm now can be written:

\[ V(z) = \frac{1}{N} \sum_{i=1}^{N} e_i^2 \]

This is a square function of \( z \) and the best agreement between the reference map and the current sea level is found when the function is at its minimum. Using the derivative you find that the minimum is when:

\[ 2 \frac{1}{N} \sum_{i=1}^{N} \left[ y_i - (x_{s,i} + x_u) - z \right] = 0 \]
Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA

Kim Andersson and Filip Traugott MDH / FMV / FOI

\[ z = \frac{1}{N} \sum_{i=1}^{N} (y_i - (x_{si} + x_{ui})) = y_{mv} - x_{mv} - x_u. \] Where \( mv \) is the mean value.

Using this in the first equation gives:

\[ V(z) = \frac{1}{N} \sum_{i=1}^{N} [y_i - x_{si} - y_{mv} + x_{mv}]^2. \] Which can be written as:

\[ V(z) = \frac{1}{N} \sum_{i=1}^{N} [(y_i - y_{mv}) - (x_{si} - x_{mv})]^2. \]

This reasoning can be approximated to the case where absolute sum is used, since a square sum can be approximated with an absolute sum in the region near the minimum value. This will result in a better approximation when the difference between the pixel values in the sonar image and the map is small, which is when good accuracy is best needed.

4.1.2. Data definition

To calculate the correlation, a large amount of data needs to be transmitted between the computer and the hardware. The data consists of:

- One bathymetric map ranging from 0.5Mbit to about 5Mbit, depending on the resolution.
- One “mean value map” containing the mean values of every correlation area in the map i.e. the mean value of the areas in the map that is compared with the sonar. The size of this data is also about 0.5Mbit to 5Mbit, depending on the resolution.
- One sonar image ranging from about 0.4kbit to 160kbit depending on the resolution, which already is subtracted with the mean value
- One correlation result, which is transmitted from the hardware to the computer. This data has logically the same size as the mean value map.

Because of the large amount of data the communication between the FPGA and the PC need to be quite fast. One probable interface is USB, which today is a standard interface in PC’s. The USB interface is very fast, up to 12Mbit/s, but it is quite complex and is not easy to implement in FPGA. Another fast interface is FIRE-WIRE (IEEE-1394), which has bandwidths over 1Gbit/s. FIRE-WIRE however, is not yet common in regular computers and is also very complex to implement. Parallel communication is another interface, which is quite fast, easy to implement and uses only a small part in the FPGA. The standard serial communication takes very small place but it is to slow to be interesting for this project. Then there is PCI communication where the card is placed inside the computer and communicates through the system bus. This is a very fast communication and is probably the best fit, the downside is off course the lack of mobility.

The memory aspect is very important. There are some different solutions on how to implement this algorithm (shown in chapter 7.2-7.3). The size of the memory needed is between ca. 4kbit and 400Mbit, depending on implementation method.

4.1.3. Future performance in FPGA

Over the years the size of an FPGA has grown exponential according to the Mores law. The biggest FPGA today has about 8 million gates and five years ago the FPGA had approximated 10000 gates. The internal memory has also had a fast expansion, 5 years ago the internal memory were ca.100kbit and today the largest FPGA contains ca. 3.5Mbit. If the Morse law is correct the development in FPGA’s will in 5 years expand to approximate 30Mbit internal memory and it will have over 64 Million gates. But there are many people who are sceptical to this and think the expansion will slow down. The extension of internal memory is very interesting for this project because of the large amount of information.
4.2. Algorithm calculation

The advantage in using hardware when calculating an algorithm of this kind is the use of parallel implementation. The calculation time can be greatly reduced compared to regular sequential programming. There are a number of different ways to implement the algorithm and they all have their advantages and disadvantages. Two of them are described below.

The first one calculates the correlation sum for every correlation area sequential. For example one correlation area correlated with a sonar image with 10*10 pixels would need to do the addition in the loop 100 times. This calculation would then be done parallel on different correlation areas.

The problem here is the unit delay. One would need to go through the delay for every pixel and since the delay will take longer time than the access time in the memory this will be the bottleneck in the calculation. To solve this several parallel computations could be implemented. However to calculate parallel one would need to access the memory with different calculators at the same time. This is not possible, provided that there is only one memory and that the depth is the same as the word length of the pixels, when external memory is used.

The other method uses an addition tree to solve the problem with the unit delays. Here the correlation between the map pixels and the sonar pixels are calculated parallel and then added in a tree structure.
Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA

Kim Andersson and Filip Traugott MDH / FMV / FOI

4.3. Conventional FPGA design

The correlation method can be implemented in different ways. All of them have their own advantages and disadvantages. The differences are the use of memory, internal and external, and how to communicate with the PC.

4.3.1. Memory method

This method is based on that the maps are downloaded to the FPGA before sending out a ping from the sonar. When the ping is made the sonar data is post proceed to form a regular small 10 x 10 map with a 10 meters grid before the data is sent to the FPGA.

A correlation calculation is made for one position and sent out to a regular PC. Simultaneously the FPGA gets new data from the external memory to the next calculation.

This method has two different approaches. The first one has two maps, one bathymetric and one mean value map. Here the FPGA first subtract the part of the bathymetric map with the mean value. After that the correlation computation is made. This method (Figure 11) needs much memory structure and pointers, which can slow it down.

The second approach is basically the same. The difference is that the data sent to the FPGA is already computed and has new values for every calculation i.e. the same data exists in different places in the memory. This means that the data size sent to the FPGA is the number of correlations times the size of the correlation map times the word length of the pixel. With this type of data it is easy to switch to a new correlation, it is just to shift it in with first in first out (FIFO) method. This memory method needs a lot of memory, up to almost 50 times the first method. Therefore external memory needs to be used.
4.3.2. Download method

This method is based on that there are memory limitations in the hardware and that the PC is able to handle this fast communication.

The problem with the method described above is the large amount of memory needed, from 8Mbit to about 400Mbit depending on the data used. Today this amount of data cannot be stored internally on FPGA and therefore external memory has to be used. This is a problem since external memory, either static or dynamic, is significantly slower to access then the internal memory on the FPGA. To reduce this problem the following method is presented.

This method is based on the use of a fast communication between the computer and the hardware. Instead of downloading all of the information needed for the entire correlation only a small bit of the data is downloaded and then calculated. After the calculation the result is sent to the computer and new data can be downloaded to presume the correlation. If the download of new data is performed during the calculation the total time needed to perform the correlation is only the down and upload time, provided that the communication is more time-consuming then the calculation. If not the time of the correlation will depend on the calculation time.
To use this method the communication, as mentioned, need to be very fast. Therefore FIRE-WIRE or PCI communication probably is the best choice since USB is to slow (the download time is over 100 times slower than the calculation time). However FIRE-WIRE is complex to implement and is not yet standard in computers.

4.4. Xilinx System Generator™ V1.1

A small investigation on whether the application System Generator™ [12] could be used to build the terrain navigation system has been conducted.

The System Generator™ is an application for the MathWorks Simulink® in MATLAB® created by Xilinx. In Simulink there are a number of blocks that can be used to model systems; the System Generator provides additional blocks to the library in Simulink. These blocks can be used, just like other Simulink blocks, to model and simulate different systems. In addition you can generate for example VHDL code and test benches from the Simulink model, provided that you use the Xilinx blockset. The VHDL code can then be implemented in an FPGA using a synthesis tool. In Figure 13 the design flow using the System Generator is shown.

Since the component library only contains a limited variety of building blocks it is not always possible to build an entire system using only the existing blocks. Therefore a block working as a black box, not surprising that is also the name of the block, is included. This block can be defined by the users own VHDL code and instantiated with the existing blocks. Another method to integrate a system built with System Generator with a system constructed in VHDL is to generate a core, which can be integrated with the users own VHDL construction. These two features are interesting since the terrain navigation system need rather complex memory handling, which is difficult to implement using only the existing blocks. The algorithm calculation however, should be possible to model using System Generator since it is a pretty simple design with common building blocks.

Another interesting aspect is the quality of the code. Since the size and speed of the FPGA design very much depends on how the VHDL code is written it is interesting to know how optimal the code is. This have not been tested in this report but one might think that since the
application is very new on the market a more optional code would be obtained if one would write the code manually.

Figure 13. System Generator flow diagram.
4.5. Conclusions

These different methods have all their own advantages and disadvantages. It is important to look at this problem as a whole. The questions: what kind of memories are accessible and what type of communication is in use, have to be answered. There are always one or several bottlenecks in the solution. Here it is the memory access time and the data sending time.

The memory method is very fast if it would be possible to store the entire data in an internal memory, which has very fast access time. But today there is no FPGA that can contain that amount of data. Therefore an external memory has to be used and the access time is therefore greatly enlarged. However, the calculation time will still be significantly lower in comparison with regular calculation in a PC. The first priority for this method is memory capacity and therefore communication comes in second place. Because most of the data is sent down to the FPGA before the “ping”, which starts the critical time, the communication time is not that important. At least not until the memory access time can be decreased. There are two types of the memory methods presented in chapter 7.3.1 and the second one that uses pre-computed data have faster memory communication then the other. But this data will take time to download to the FPGA. So one can say that this method is the most suitable when there is external memory and time before the ping. But when looking forward into the future the internal memory capacity will enlarge and therefore the first type should be used in the future.

The second method the “download method“ has the advantage that it does not need a lot of memory space. Instead it needs very fast communication. The different communication interfaces that are presented in chapter 7.3.2 have some disadvantages. Both FIRE-WIRE and USB communication needs to register with a company who have to approve your application. These two interfaces need also quite complex programming and it will probably take a lot of space in the FPGA.

The system generator is a tool, which makes hardware design much easier and user friendly. The system makes it possible to design systems using the popular Simulink modelling system. However when constructing more complex systems, such as terrain navigation, the application can only be used in some parts of the system since the building blocks provided are still to basic for some parts of the design.
5. Summary of the investigation

After this investigation about navigation system we have found that there is several different methods that all have their advantages and disadvantages. All of the methods are working fine. But every one has their requirements and often this requirement is not compatible with another method. Every method has limitations in their system. It is therefore interesting if there comes a system that is more general so it can accomplish more requirements.

The hardware implementation is another subject that has been brought up. The biggest problem with the terrain correlation method is the large amount of data needed. Today it is only possible to implement such a system on a FPGA that is supported by external memory. The calculation time can then be reduced up to 100 times compared to calculation on a regular PC. When FPGA chips with sufficient internal memory are available within a few years the calculation time will be significantly lower. However, such a system would also need a fast communication, as FIRE-WIRE or the PCI communication, to work in relative real time.

The system generator in Simulink is a very interesting product. The program has been examined to get some ideas about its functions and possible applications in the implementation of a terrain navigation system. The investigation has shown that it seems to be possible to construct a navigation system using System Generator. However, some of the building blocks would need to be designed by the user and integrated with the existing blocks.
PART II - Realisation of the method

Part two of the report describes an implementation of a terrain correlation navigation system in a hardware accelerator. The implementation is based upon the investigation described in part one of the report.

6. Examined case, test card and tools

6.1. Examined case

This design is built with the limitation of only handling one specific size of the map matrix and sonar matrix. The use of only one map and sonar size makes it easy to compare the hardware correlation design with a software design. It is possible to make this design with different sizes of the map and sonar. It is also possible to build a generic design but it is a more complex design.

In this typical case the map is a 501 times 501 matrix and the sonar is a 10 times 10 matrix. Both has a grid of ten meter, this means that the map has a range of 5000 times 5000 meters and the sonar has a range of 90 times 90 meters.

This typical case is a good approximation of which sizes the map and sonar can be in a live situation. This means that this design can be used for live tests.

6.2. Test card

The test card for this project, shown in Figure 14, is a newly developed card with 42 I/Os, 4 LED’s, a Xilinx Virtex-II 1000ff896 FPGA and 2 Samsung SDRAM (16M x 16 bites). Some of the I/Os will be used for the parallel port.

The Xilinx Virtex-II chip has 1280 CLBs (Configurable Logic Block), one CLB consist of four slices and each slice includes two 4-input function generators, carry logic, arithmetic logic gates, wide function multiplexers and two storage elements. Each 4-input function generator is programmable as a 4-input LUT, 16 bits of distributed SelectRAM memory, or a 16-bit variable-tap shift register element. These are the actual hardware building blocks. In Figure 15 the layout of a Virtex-II CLB-slice is shown. [13]
6.3. Design tools

The software used in this project is Matlab 6.1, ModelSim EE 5.2e, Leonardo Spectrum LS2002a and Xilinx ISE 4.2. Matlab 6.1 has been used to simulate the correlation, to build up download files and for result verification. ModelSim EE 5.2e has been used to simulate the test bench, and to synthesize the block designs, the program Leonardo spectrum LS2002a has been used. The Xilinx ISE 4.2i system has been used to make “place and route” and to download the files to the FPGA. Readers who are interested in the tools used can read more about them on their websites, where user guides are available. [14][15][16][17]
7. Architecture design

In this section of the report the architecture design of the navigation system is described. First an overview of the design is presented and afterwards follows a more detailed description of the building blocks from which the accelerator is constructed.

7.1. The Correlator (Top design)

The correlator is designed to calculate the correlation between a map and a sonar image of a predetermined size. The block "CORR" performs the calculation. The parallel EPP interface between the PC and the correlator is implemented in the block “HOST”. The interface between the external memory and the correlator is implemented in the blocks “SDRAM_CONTROLLER” and “MEM_COM”. The block “BUFF” is a buffer memory between the external memory and the “CORR” and the block “SEND” sends the results from the “CORR” to the “HOST”.

A correlation can be performed by downloading the map and mean data according to the sequence described in 9.2, followed by the sonar data according to the sequence described in 9.1. The calculation begins immediately after the last sonar data has been sent. When the calculation is ready, the correlator starts to fill the buffer with the first map and mean data again. New sonar data can then be downloaded and the calculation begins again with the new sonar data and the old map and mean data. The PC can decide either to download new map and mean data or download new sonar data. But to start a correlation, sonar data has to be downloaded. Note that after a start-up or a reset, sonar data can not be downloaded before the map and the mean data, simply because there are no data for the correlation. [20]

![Figure 16. Block description of the correlator design.](image)

**7.1.1. Execution flow**

1. Wait for command from PC.

2. Execute command. If the command is send map and mean data start receiving data from the PC and sending the data to the external memory. If the command is send sonar and map and mean data is downloaded, start receiving sonar data and send to block “CORR“.
3. If data exists in the external memory start filling the buffer with the data. If sonar has been downloaded and data exists in buffer, start to calculate. Start receiving new data from buffer parallel with the calculation.

4. When calculation is finished, send the results to the PC synchronous with the start of the next calculation.

5. When the correlation is finished go back to 1.

7.1.2. Area and timing summary

Design Summary:
- Number of errors: 0
- Number of warnings: 9
- Number of Slices: 5,118 out of 5,120 99%
- Number of Slices containing unrelated logic: 238 out of 5,118 4%
- Number of Slice Flip Flops: 7,497 out of 10,240 73%
- Total Number 4 input LUTs: 8,437 out of 10,240 82%
  - Number used as LUTs: 8,391
  - Number used as a route-thru: 46
- Number of bonded IOBs: 118 out of 432 27%
- Number of Tbufs: 64 out of 2,560 2%
- Number of Block RAMs: 23 out of 40 57%
- Number of GCLKs: 1 out of 16 6%

Total equivalent gate count for design: 1,620,229
Additional JTAG gate count for IOBs: 5,664

Timing summary:
------------
Timing errors: 0  Score: 0

Constraints cover 594714 paths, 0 nets, and 49088 connections (90.6% coverage)

Design statistics:
- Minimum period: 19.878ns (Maximum frequency: 50.307MHz)

7.2. HOST

7.2.1. Block function

The block "HOST" is the interface between the PC and the FPGA. It communicates with and interprets what the PC has sent. The different commands are then executed via the connecting blocks. There are two commands which the PC can call. They are "Send map and mean data" or "Send sonar data". When the command “Send map and mean data” is received the “HOST” receives the map data from the PC and sends it to the block “MEM_COM”. When the command “Sending sonar data” is received the “HOST” receives the sonar image from the PC and sends it to the block “CORR”. When the entire sonar image is received the host expects the PC to change mode to read instead of write. This enables the “HOST” to send the result, provided by the block “SEND”, to the PC.
Table 1. Table showing entity signals in the block "HOST".

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>IN/OUT</td>
<td>Parallel data signals, can be used in both directions</td>
</tr>
<tr>
<td>nRESET</td>
<td>IN</td>
<td>When the signal is logic high the reset is activated. This signal is used as the sw_reset_n signal, which is connected to all blocks and used as a regular reset.</td>
</tr>
<tr>
<td>nWRITE</td>
<td>IN</td>
<td>This signal shows in which direction the data is transmitted. Logical low means that the block will receive data on AD and logic high that the block will send data on AD</td>
</tr>
<tr>
<td>nDATASTB</td>
<td>IN</td>
<td>The signal shows that there is new data on AD or that data can be written to AD. Logic low means new data can be written/read. The signal is set to logic high again when data is written/read.</td>
</tr>
<tr>
<td>NWAIT</td>
<td>OUT</td>
<td>The signal shows when the data is written/read. Goes high to tell that the PC that data is written/read. The signal stays logic high until &quot;nDATASTB&quot; is logic high.</td>
</tr>
<tr>
<td>DATA_HC</td>
<td>OUT</td>
<td>Sends data parallel to block “CORR”.</td>
</tr>
<tr>
<td>REQ_HC</td>
<td>OUT</td>
<td>The signal shows if there is data on &quot;DATA_HC” for the “CORR” to read. It is set to logic high when there is data on &quot;DATA_HC”. It remains logic high until &quot;READY_HC” is switched to logic high, which indicates that data is received by the “CORR”. &quot;READY_HC” must have switched to logic low before a new request on &quot;REQ_HC can be made.</td>
</tr>
<tr>
<td>READY_HC</td>
<td>IN</td>
<td>The signal show when the data in &quot;DATA_HC” can be changed. The signal is logic low and set to logic high when the data on &quot;DATA_HC” can be changed. The signal remains high until “REQ_HC” is switched to low.</td>
</tr>
<tr>
<td>SONAR_FULL</td>
<td>IN</td>
<td>The signal is set high when the entire sonar is received by the “CORR”.</td>
</tr>
<tr>
<td>DATA_SH</td>
<td>IN</td>
<td>Receive data parallel from block “SEND”.</td>
</tr>
<tr>
<td>REQ_SH</td>
<td>IN</td>
<td>The signal shows when there is data on “DATA_SH”. It is logic low and set to logic high when ”DATA_SH” can be read. The signal is high until &quot;READY_SH” is set to logic high. A new request can only be made after the signal &quot;READY_SH” is switched to logic low.</td>
</tr>
<tr>
<td>READY_SH</td>
<td>OUT</td>
<td>The signal shows when the data on &quot;DATA_SH” is read. It is logic low and set to logic high when data on &quot;DATA_SH” has been read. It remains logic high until the signal &quot;REQ_SH is set to logic low.</td>
</tr>
<tr>
<td>FINISH_SEND</td>
<td>IN</td>
<td>The signal goes high when the last result is sent to the “HOST”. Remains high until new sonar or map data is downloaded.</td>
</tr>
<tr>
<td>MEM_FULL</td>
<td>IN</td>
<td>The signal is logic high when there is a map in the memory.</td>
</tr>
<tr>
<td>DATA_HF</td>
<td>OUT</td>
<td>Sends data parallel to “FIFO1”.</td>
</tr>
</tbody>
</table>
7.2.2. Execution flow

1. Wait until the first byte has been collected.

2. Check what the command was.
   If the command was "Send map and mean data" go to "3".
   If the command was "Send sonar data" go to "5".

3. Wait for the next 5325598 bytes to come i.e. the PC is sending map and mean values. 
   Send the bytes direct when it is received via the "DATA_HF" signals to the "FIFO1" block where data can be read by the "MEM_COM" block.

4. Go to "1" and wait for next command.

5. Wait for the next 205 bytes to come i.e. the PC is sending the sonar values.
   Send the bytes direct when it is received via the "DATA_HC" signal to the "CORR" block.

6. Start looking for result data from the "SEND" block via the signal "DATA_SH" and send it to the PC.
7.2.3. Area and timing summary

Design Summary:
- Number of errors: 0
- Number of warnings: 0
- Number of Slices: 78 out of 5,120 (1%)
- Number of Slices containing unrelated logic: 0 out of 78 (0%)
- Number of Slice Flip Flops: 82 out of 10,240 (1%)
- Number of 4 input LUTs: 71 out of 10,240 (1%)
- Number of bonded IOBs: 85 out of 432 (19%)
- Number of GCLKs: 1 out of 16 (6%)

Total equivalent gate count for design: 1,121
Additional JTAG gate count for IOBs: 4,080

Timing summary:
------------------
Timing errors: 0  Score: 0

Constraints cover 448 paths, 0 nets, and 415 connections (76.7% coverage)

Design statistics:
- Minimum period: 5.589ns (Maximum frequency: 178.923MHz)
7.3. MEM_COM

7.3.1. Block function

The block "MEM_COM" is the commander for memory control. It receives map and mean data from the block “HOST” and sends it to the "SDRAM_CONTROLLER", which stores it in the external memory. The data is sent to the memory with “byte by byte” mode, i.e. the data is sent one byte at a time with handshake signals in-between.

When “MEM_COM” has sent all data into the memory it starts to read the data from the memory and sends it to the block “CORR”. This information is sent from the memory in “burst” mode, which means that it comes 2, 4 or 8 data in a row. Since this design uses two memories, the amount of data in every burst is twice as big. Thus, the burst mode makes it possible to transfer data between the external memory and the correlator at a much higher rate then the “byte by byte” mode. When the last data is sent the block sends a signal through the block “BUFF” to the “CORR” block to indicate that the last data has been sent.

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE</td>
<td>OUT</td>
<td>The signal is set to logic high when there is data in “FIFO1”. The signal is pulsed when the “MEM_COM” wants to read data.</td>
</tr>
<tr>
<td>EMPTY_FIFO</td>
<td>IN</td>
<td>The signal indicates when the fifo memory is empty. A logical one means that the memory is empty.</td>
</tr>
<tr>
<td>DATA_FM</td>
<td>IN</td>
<td>Receives data parallel from block “FIFO1”.</td>
</tr>
<tr>
<td>WE_MC</td>
<td>OUT</td>
<td>A write enable signal to the second fifo memory “FIFO2”. For every clock cycle the signal is high, the fifo memory reads one data. The “MEM_COM” block sends 16 data at a time.</td>
</tr>
<tr>
<td>FULL</td>
<td>IN</td>
<td>The signal shows when the “FIFO2” is full. When logical high the memory is full.</td>
</tr>
<tr>
<td>CLEAR_BUFF</td>
<td>OUT</td>
<td>The signal is used to clear the data in the block “FIFO2”.</td>
</tr>
<tr>
<td>DATA_IN</td>
<td>OUT</td>
<td>Sends data parallel to block “FIFO2”</td>
</tr>
<tr>
<td>READD</td>
<td>OUT</td>
<td>The signal is used to enable read or write to the external memory. Logic low means write to the memory and vice versa.</td>
</tr>
<tr>
<td>GO</td>
<td>OUT</td>
<td>The signal goes logic low when there is data for the memory to read or when the block wants to read from the memory. The signal goes high again when the signal “READY” goes high.</td>
</tr>
<tr>
<td>BURST</td>
<td>OUT</td>
<td>The signal is used to switch to burst mode in the memory. Logical high indicates burst mode.</td>
</tr>
<tr>
<td>READY</td>
<td>IN</td>
<td>The signal is set logical high when the memory has read the data or when there is data to read on “DATA”. The signal is set logical low again when “GO” is set high.</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>OUT</td>
<td>Tells the memory where in the memory the “MEM_COM” block wants to read or write the data.</td>
</tr>
<tr>
<td>DATA</td>
<td>IN/OUT</td>
<td>Sends or receives data parallel from/to &quot;SDRAM_CONTROLLER” block.</td>
</tr>
</tbody>
</table>
7.3.2. Execution flow

1. Wait for data from the “HOST”.
2. Put the data in the memory. Go back to “1” until all data has been received.
3. Get data from the memory (It will be received in burst mode with 16 bytes each time).
4. Check if buffer is full in block “BUFF”. Else send data to block “BUFF”.
5. Go to ”3” until all data has been send away then start sending the data again. If the “HOST” has new data go to 1.

Figure 18. Graf showing the execution flow in the block “MEM_COM”.

7.3.3. Area and timing summary

Design Summary:

- Number of errors: 0
- Number of warnings: 0
- Number of Slices: 436 out of 5,120 8%
- Number of Slices containing unrelated logic: 0 out of 436 0%
- Number of Slice Flip Flops: 389 out of 10,240 3%
- Number of 4 input LUTs: 748 out of 10,240 7%
- Number of bonded IOBs: 110 out of 432 25%
- Number of GCLKs: 1 out of 16 6%
Total equivalent gate count for design:  7,918  
Additional JTAG gate count for IOBs:  5,280  

Timing summary:  
------------------  
Timing errors: 0  Score: 0  

Constraints cover 25765 paths, 0 nets, and 3747 connections (91.8% coverage)  

Design statistics:  
Minimum period: 11.180ns (Maximum frequency: 89.445MHz)  

7.4. FIFO1 and FIFO2  

7.4.1. Block function  
The blocks “FIFO1” and “FIFO2” are regular fifo memories which are used as buffers. The first memory is used as a buffer between the external memory and the PC. This buffer reduces the problem with timeouts in the EPP parallel communication, which is discussed in 8.2.4. The block is about 16kbyte and uses the on chip block ram memories which can be found on the Xilinx Virtex –II 1000ff896 chip. This means that the memories and the regular design do not use the same logic, since the memories uses specific memory blocks. [22]  
The second fifo memory is used as a buffer between the external memory and the calculation (“CORR”) block. This buffer makes it possible to load a portion of the data from the external memories before the actual calculation begins. This also reduces the problem with timeouts in the EPP interface, since the correlator always has data for the calculations. The block is about 32kbyte.  

Table 3. Table showing entity signals in the block "FIFO1". The signals to the second fifo memory has the same functions as described in this table.  

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WE</td>
<td>IN</td>
<td>The signal shows if there is data on ”DATA_HF” to be sent to the fifo memory. The signal is pulsed with a logical one every time a data is written.</td>
</tr>
<tr>
<td>FULL_FIFO</td>
<td>OUT</td>
<td>The signal shows if the “FIFO1” is full or not. When “FULL_FIFO” is high the memory is full and no data can be written.</td>
</tr>
<tr>
<td>DATA_HF</td>
<td>IN</td>
<td>Parallel data from the “HOST” block.</td>
</tr>
<tr>
<td>RE</td>
<td>IN</td>
<td>The signal is set to logic high to enable read when there is data in “FIFO1”. The signal is pulsed when the “MEM_COM” wants to read data.</td>
</tr>
<tr>
<td>EMPTY_FIFO</td>
<td>OUT</td>
<td>The signal indicates when the fifo memory is empty. A logical one means that the memory is empty.</td>
</tr>
<tr>
<td>DATA_FM</td>
<td>OUT</td>
<td>Parallel data to “CORR” block.</td>
</tr>
</tbody>
</table>
7.5. CORR

7.5.1. Block function

The block “CORR” is the block that performs the correlation. It consists of four equally sized arrays which are used to store the data for one calculation i.e. the sonar data and the mean and map data from one correlation area. Two of the arrays are used to receive the data from the blocks “HOST” and “BUFF” and two are used in the calculation. By using four separate arrays the block can calculate and receive data parallel. The calculation is performed by two parallel calculators (described in Figure 9), which can perform the calculation of one correlation area and send the result in 55 clock cycles (according to simulation). Using a clock frequency of 50 MHz the correlation can be performed in about 0.27 seconds. However, at that speed both the communication with the PC and the external memory is slower.

To reduce the problem with overflow in the adders the design stops the calculation every time an overflow occurs. The result sent when the adders are overflowed is the largest result that can be represented i.e. 32 767. This way an overflow is represented by a very poor correlation error, which is the best way to represent it since the correlation error when an overflow occurs actually is even bigger. [21]

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| REQ_HC        | IN        | The signal shows if there is data on ”DATA_HC” for the “CORR” to read. It is set to logic high when there is data on ”DATA_HC”. It remains logic high until ”READY_HC” is switched to logic high, which indicates that data is received by the “CORR”.
|               |           | ”READY_HC” must have switched to logic low before a new request on ”REQ_HC” can be made. |
| READY_HC      | OUT       | The signal show when the data in ”DATA_HC” can be changed. The signal is logic low and set to logic high when the data on ”DATA_HC” can be changed. The signal remains high until “REQ_HC” is switched to low. |
| DATA_HC       | IN        | Receives data parallel from block “HOST”.                                    |
| SONAR_FULL    | OUT       | The signal is set high when the entire sonar is received.                    |
| REQ_CS        | IN        | The signal tells the “SEND” block that there is data on ”DATA_CS”. It switches from logic low to logic high when data can be read on ”DATA_CS”. It remains logic high until ”READY_CS” switches to logic high |
| READY_CS      | OUT       | The signal tells the “CORR” block that the data on ”DATA_CS” has been read. The signal switches from logic low to logic high when ”REQ_CS” is high and the data on ”DATA_CS” is read. It remains logic high until REQ_CS switches to logic low. |
| DATA_CS       | OUT       | Send data parallel to the “SEND” block.                                     |
| MAP_FULL      | OUT       | The signal tells the “SEND” block that the last result is calculated.       |
| FINISH_SEND   | IN        | The signal tells blocks “HOST” and “CORR” when the last result has been sent. |
| RE_MC         | OUT       | The signal is used to enable a read from the “FIFO2” block. A logical one initiates a read. The signal is |
7.5.2. Execution flow

1. The block receives sonar, map and mean data and stores the data in two arrays.

2. When all data for one correlation area is received the calculation begins and the data for the next calculation is received.

3. The result is sent to the block “SEND”.

4. Return to 2 until entire map is correlated, then return to 1.

Design Summary:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of errors</td>
<td>0</td>
</tr>
<tr>
<td>Number of warnings</td>
<td>0</td>
</tr>
<tr>
<td>Number of Slices</td>
<td>4,496</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>6,610</td>
</tr>
</tbody>
</table>

When data for one correlation is received.

When correlation is finished.

When result is sent and new data received.

When calculation ready.

Send results to block “SEND” and then wait for the new data to be received.

Figure 19. Graf showing the execution flow in the block “CORR”.

7.5.3. Area and timing summary

When correlation is finished.

When correlation is finished.
Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA

Kim Andersson and Filip Traugott MDH / FMV / FOI

Total Number 4 input LUTs:       7,122 out of 10,240   69%
Number used as LUTs:                        7,112
Number used as a route-thru:                   10
Number of bonded IOBs:              65 out of     432   15%
Number of GCLKs:                     1 out of      16    6%
Total equivalent gate count for design:  96,741
Additional JTAG gate count for IOBs:  3,120

Timing summary:
-------------------
Timing errors: 0  Score: 0

Constraints cover 517448 paths, 0 nets, and 41186 connections (90.4% coverage)

Design statistics:
Minimum period:  17.800ns (Maximum frequency:  56.180MHz)

7.6. SEND

7.6.1. Block function
The block “SEND” is to communicate with the blocks “HOST” and “CORR”. Its only task is to receive results from the calculations in the block “CORR” and send them to the block “HOST”, this in order to make it possible to calculate while sending results to the PC. The block also contains a small buffer where results are stored if the “HOST” is busy.

Table 5. Table showing entity signals in the block "SEND".

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ_SH</td>
<td>OUT</td>
<td>The signal tells the block “HOST” that there is data on &quot;DATA_SH&quot;. The signal is logic low and switched to logic high when a request is made. It goes logic low when &quot;READY_SH&quot; is high.</td>
</tr>
<tr>
<td>READY_SH</td>
<td>IN</td>
<td>The signal tells the “SEND” block when the data on &quot;DATA_SH&quot; has been read. The signal switches from logic low to logic high when &quot;REQ_SH&quot; is logic high and data has been read on &quot;DATA_SH&quot;. It remains logic high until REQ_SH switches to logic low.</td>
</tr>
<tr>
<td>DATA_SH</td>
<td>OUT</td>
<td>Sends data parallel.</td>
</tr>
<tr>
<td>FINISH_SEND</td>
<td>OUT</td>
<td>The signal tells blocks “HOST” and “CORR” when the last result has been sent.</td>
</tr>
<tr>
<td>REQ_CS</td>
<td>IN</td>
<td>The signal tells the “SEND” block that there is data on &quot;DATA_CS&quot;. It switches from logic low to logic high when data can be read on &quot;DATA_CS&quot;. It remains logic high until &quot;READY_CS&quot; switches to logic high.</td>
</tr>
<tr>
<td>READY_CS</td>
<td>OUT</td>
<td>The signal tells the “CORR” block that the data on &quot;DATA_CS&quot; has been read. The signal switches from logic low to logic high when &quot;REQ_CS&quot; is high and the data on &quot;DATA_CS&quot; is read. It remains logic high until REQ_CS switches to logic low.</td>
</tr>
</tbody>
</table>
7.6.2. Execution flow

1. Wait for block “CORR” to send data.

2. Place data in buffer.

3. Tell block “HOST” that new data can be collected and place the data on DATA_SH.

4. Wait for the “HOST” to collect data, or receive new results from the “CORR” and place them in the buffer (stop receiving when buffer is full).

5. Go back to “1” if buffer is empty, else go to “3”.

![Graf showing the execution flow in the block "SEND".](image)

7.6.3. Area and timing summary

Design Summary:

- Number of errors: 0
- Number of warnings: 0
- Number of Slices: 134 out of 5,120 (2%)
- Number of Slices containing unrelated logic: 0 out of 134 (0%)
- Number of Slice Flip Flops: 196 out of 10,240 (1%)
- Total Number 4 input LUTs: 209 out of 10,240 (2%)
- Number used as LUTs: 206
- Number used as a route-thru: 3
- Number of bonded IOBs: 46 out of 432 (10%)
- Number of GCLKs: 1 out of 16 (6%)
- Total equivalent gate count for design: 2,883
- Additional JTAG gate count for IOBs: 2,208
Timing summary:
--------------
Timing errors: 0  Score: 0

Constraints cover 6622 paths, 0 nets, and 1045 connections (88.6% coverage)

Design statistics:
Minimum period: 11.798ns (Maximum frequency: 84.760MHz)

7.7. SDRAM_CONTROLLER

7.7.1. Block description
The SDRAM_CONTROLLER is the block that controls the sdram memories. The block was provided to us as an open IP-block, therefore only its purpose is described.

The block is the interface between the correlator and the external memory. The correlator can use a few different commands to invoke a read or write cycle. It can write in single bit mode, which is the standard mode where one data is written in every cycle. When reading, the correlator can choose from two different commands single read and burst mode. In the single read mode the read cycle is the same as for single write mode. In burst mode the data is received in packages of eight and the data in the packages are received one on every clock cycle. The burst mode, which is used in read mode in this design, significantly decreases the communication time with the external memories.

7.7.2. Area and timing summary

Design Summary:
Number of errors: 0
Number of warnings: 0
Number of Slices: 114 out of 5,120 2%
Number of Slices containing unrelated logic: 0 out of 114 0%
Number of Slice Flip Flops: 146 out of 10,240 1%
Number of 4 input LUTs: 124 out of 10,240 1%
Number of bonded IOBs: 135 out of 432 31%
Number of GCLKs: 1 out of 16 6%
Total equivalent gate count for design: 2,230
Additional JTAG gate count for IOBs: 6,480

Timing summary:
--------------
Timing errors: 0  Score: 0

Constraints cover 2393 paths, 0 nets, and 577 connections (62.4% coverage)

Design statistics:
Minimum period: 6.829ns (Maximum frequency: 146.434MHz)
8. Protocols

A simple protocol has been developed containing only the most necessary commands to perform the calculation. The protocol can, in the future, be extended to include other commands. In the section the EPP protocol, which is used in this project, and the commands used to control the correlator are described. Interested readers may read in [18] or [19] for more information about the EPP protocol.

8.1. Physical protocol

Parallel communication is used between the PC and the correlator. The interface used is the Enhanced Parallel Port (EPP) standard, which can transfer data, 8 bits at a time, in both directions at up to 2MB/s. The EPP can distinguish between two different types of information, usually defined as data and addresses. The addresses can therefore be used to toggle a command and the data only as data communication (in serial communication the commands has to be sent in the data stream). This function is not used in this design since fast serial communication such as FIRE WIRE might be used as an interface in the future. Therefore the protocol is designed to fit the use of serial communication.

Below follows a description of the EPP interface.

The cable used is the IEEE 1284-A also called D-sub.

Table 6: The signals in the EPP protocol. ( ' Indicates a inverted signal)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Register bit</th>
<th>Pin</th>
<th>EPP signals</th>
<th>Entity in FPGA</th>
<th>IN/OUT seen from the PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data bit 0</td>
<td>D0</td>
<td>2</td>
<td>AD(0)</td>
<td>AD(0)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 1</td>
<td>D1</td>
<td>3</td>
<td>AD(1)</td>
<td>AD(1)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 2</td>
<td>D2</td>
<td>4</td>
<td>AD(2)</td>
<td>AD(2)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 3</td>
<td>D3</td>
<td>5</td>
<td>AD(3)</td>
<td>AD(3)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 4</td>
<td>D4</td>
<td>6</td>
<td>AD(4)</td>
<td>AD(4)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 5</td>
<td>D5</td>
<td>7</td>
<td>AD(5)</td>
<td>AD(5)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 6</td>
<td>D6</td>
<td>8</td>
<td>AD(6)</td>
<td>AD(6)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>Data bit 7</td>
<td>D7</td>
<td>9</td>
<td>AD(7)</td>
<td>AD(7)</td>
<td>IN/OUT</td>
</tr>
<tr>
<td>nError (nFault)</td>
<td>S3</td>
<td>15</td>
<td>Not in use</td>
<td>Not in use</td>
<td>IN</td>
</tr>
<tr>
<td>Select</td>
<td>S4</td>
<td>13</td>
<td>Not in use</td>
<td>Not in use</td>
<td>IN</td>
</tr>
<tr>
<td>PaperEnd</td>
<td>S5</td>
<td>12</td>
<td>Not in use</td>
<td>Not in use</td>
<td>IN</td>
</tr>
<tr>
<td>Nack</td>
<td>S6</td>
<td>10</td>
<td>Interrupt</td>
<td>INTR</td>
<td>IN</td>
</tr>
<tr>
<td>Busy</td>
<td>S7'</td>
<td>11</td>
<td>Wait</td>
<td>nWAIT</td>
<td>IN</td>
</tr>
<tr>
<td>Nstrobe</td>
<td>C0'</td>
<td>1</td>
<td>Write</td>
<td>nWRITE</td>
<td>OUT</td>
</tr>
<tr>
<td>NautoLF</td>
<td>C1'</td>
<td>14</td>
<td>Data Strobe</td>
<td>nDATASTB</td>
<td>OUT</td>
</tr>
<tr>
<td>Ninit</td>
<td>C2</td>
<td>16</td>
<td>Reset</td>
<td>nReset</td>
<td>OUT</td>
</tr>
<tr>
<td>NselectIn</td>
<td>C3'</td>
<td>17</td>
<td>Address Strobe</td>
<td>Not in use</td>
<td>OUT</td>
</tr>
<tr>
<td>Ground</td>
<td>18-25</td>
<td></td>
<td>Ground</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D0 -> D7 is the two-way data buss.

S6 Is a signal to the PC, which is used by the correlator as an indicator for when it is busy. A logical ‘1’ indicates that the correlator is busy and that no data can be sent or received. This is a function not supported in the EPP standard (IEEE 1284), however it is a recommend use of the interrupt pin when the peripheral needs to enable a read or write. Therefore the
programmer has to use the interrupt as an indicator pin from the correlator for enabling a read or write cycle.

**S7**’ Is a signal to the PC, which the correlator uses to inform the PC when data has been written/read. A logical ‘1’ indicates that the correlator has written/read.

**C0**’ Is a signal from the PC, used to chose read or write mode on the parallel port. A logical ‘1’ indicates that the PC wants to read. Since the PC chooses the direction of the communication it always have to enable read mode after sending the sonar data so that the results from the correlator can be received.

**C1**’ Is a signal from the PC, which enables a read or write (read or write is chosen by C0’). A logical ‘0’ enables a read/write on the parallel port.

**C2** Is a signal from the PC used to invoke a reset on the correlator.

### 8.1.1. Data Write

Below follows a description of the timing characteristics for a data write cycle.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>0</td>
<td>10 μsec.</td>
</tr>
<tr>
<td>Tel</td>
<td>0</td>
<td>10 μsec.</td>
</tr>
<tr>
<td>Th</td>
<td>0</td>
<td>1 sec.</td>
</tr>
<tr>
<td>Tes</td>
<td>0</td>
<td>125 nsec.</td>
</tr>
<tr>
<td>Td</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
1. The correlator’s data outputs are disabled and nWait and intr is low. The PC brings nWrite low, writes data to the EPP Data register, which causes the byte to appear on D0 – D7, and brings nDStrobe(C1’) low.
2. The correlator brings nWait high to signal that it is ready to latch the data.
3. The PC brings nDStrobe high to cause the correlator to latch the data.
4. When the correlator is ready for another byte, it brings nWait low.

### 8.1.2. Data read
Below follows a description of the timing characteristics for a data read cycle.
Figure 22. An EPP data read cycle.

1. The correlator’s nWait and intr must be low. The PC brings nWritee high, disables outputs D0 – D7, and brings nDStrobe low.

2. The correlator enables its D0 – D7 outputs, writes data to them, and brings nWait high to signal the PC that the data is available to be read.

3. The PC reads DO – D7 at the EPP Data register and brings nDStrobe high.

4. The correlator disables outputs D0 – D7 and brings nWait low.

8.2. Sending protocol

Three different commands are used in the design: “reset”, “send map and mean data” and “send sonar data”. The first byte in the sending sequence is used to choose the command except for “reset” where the signal “reset” is used to toggle the command.

When a command is received the sequence represented by that command is executed and then a new command can be sent.
8.2.1. Reset

**Command:**
Reset

**Description:**
All operations are stopped and returned to its start position. The memory will not be cleared but all of the information regarding the downloaded data is lost. Therefore the data needs to be downloaded again after a reset.

**Syntax:**
Set C2 in the EPP control register to zero.

**Bit configuration:**
In port nReset set to logical “0” => “reset”

8.2.2. Send map and mean data

**Command:**
Send map and mean data

**Description:**
The map and mean data is downloaded in the order shown in Figure 23. The first byte is the command and the ten following are empty bytes that can be used in the future as data to the correlation, for example map and sonar size. The remaining bytes are the map and mean data downloaded in the sequence described in 9.2. After the data is downloaded, the correlator returns to its start position and waits for a new command.

**Syntax:**
1. Read the status register and check if the interrupt signal is low. If so proceed, if not read again and check.
2. Write 0 to the EPP data register to call upon the command.
3. Read the status register and check the interrupt signal. Write the data to the EPP data register in the order shown in Figure 23 and according to the sequence shown in 9.2. Return to 3 until the last data is sent, then chose new command.

**Bit configuration:**
The command byte is set to “0000 0000” when command “sending map and mean data” is called upon.
Data bits AD(0) – AD(7) are then set according to data written.
Figure 23: Sending order for command “send map and mean data”. The number of bytes is according to the case with a map size of 501x501 pixels and a sonar size of 10x10 pixels.

8.2.3. Send sonar data

**Command:**
Send sonar data

**Description:**
The sonar is downloaded in the order shown in Figure 24. The first byte is the command and the five following are empty bytes which can be used in the future as data to the correlator. The remaining bytes are the sonar data downloaded in the sequence described in 9.1. When the sonar data is downloaded the correlator automatically begins to calculate and starts sending results. *No additional command is therefore needed to start the calculation.*

When the map, mean and sonar data is downloaded the correlator starts to calculate automatically. The results are sent to the PC during the calculation, hence the PC has to start reading the parallel port after sending the sonar. The results are sent in the order shown in Figure 24 and in the sequence described in 9.3. When the results are sent, the correlator returns to its start position and waits for a new command.

**Syntax:**
1. Read the EPP status register and check if signal interrupt is low. If so proceed, if not read again and check.
2. Write 1 to the EPP data register to call upon the command.
3. Read the status register and check the interrupt signal. Write the data to the EPP data register in the order shown in Figure 24 and according to the sequence shown in 9.1. Return to 3 until the last data sent.
4. Read the status register and check the interrupt signal. Read the data on the EPP data register. Return to 4 until last data is received, then chose new command.

**Bit configuration:**
The command byte is set to 0000 0001 when command “sending sonar data” is called upon. Data bits AD(0) – AD(7) are set according to data written or read.
8.2.4. Timing issues

When using sdram memories in a real-time system, timing issues has to be considered. The sdram memory needs to be refreshed every 64 ms and the refresh takes about 160us to complete. The refresh makes the external memory inaccessible and therefore the dataflow is stopped every 64 ms. This is a problem when using the EPP protocol, since the protocol has a timeout time where the peripheral, i.e. the correlator, has to be accessible at all times. This problem can be solved in several ways and in the report two methods are used.

The first and easiest is using buffers in the design so that there always is space for new data. This method can be used when the amount of data that is sent is known. Then the buffers can be dimensioned after the amount of data being transferred. Since the data flow is predetermined in the correlator, this is a way of solving the timing issues for this design.

The second solution used in this design is the use of the interrupt signal in the EPP protocol. The interrupt pin can be used to tell the PC either that the correlator is busy or that data can be sent or received. The communication speed will in this case be reduced since the PC has to read the status register before every time it writes or reads. This use of the interrupt pin has no specified protocol in the EPP standard (IEEE 1284), however it is mentioned as a possible way of reducing timing problems. The protocol therefore has to be specified by the programmer. In this report this is described in the syntax part of the command description in 8.2.
9. Loading/Sending sequence and data structure

A special sequence for downloading the data is used in the correlator. This sequence is used because it greatly simplifies the communication with the memory. By downloading the data in the same sequence it is uploaded to the calculation in the correlator, no additional addressing block is needed. The data is simply uploaded in numerical order starting on the first address in the download (requires that the memory is addressed in numerical order in the download).

The sonar and map data is represented binary by 16 bits where the most significant bit is a sign bit. This makes it possible to represent numbers in the range –32768 to 32767, which is enough for the depths in Swedish waters and the resolution in the maps used in this project. Since the parallel communication only sends 8 bits at a time, the data from the sonar and the map has to be divided in two. A 16 bit data is therefore sent starting with the 8 least significant bits followed by the 8 most significant bits.

9.1. Sending sonar

The sonar data sent from the PC is the data from the sonar subtracted with the mean of the sonar data (as shown in 4.1.1), hence no mean from the sonar data is needed in the correlation. The sonar data is sent column by column i.e. if the sonar image is of the size 10x10 pixels the pixels are numbered from 1 to 100 column vice with the first pixel in the upper left corner and sent in numerical order.

![Figure 26: Numbering of the sonar data. The data is sent in numerical order starting with pixel number 1.](image)

9.2. Sending map and mean values

The map in the example is of the size 501x501 pixels.

9.2.1. Mean data from the map

The mean data is the mean values from every correlation area in the map. The mean values can be represented by a matrix where the first value in the matrix corresponds to the first correlation area in the map and so on. The mean value matrix is therefore of the size:

\[((\text{number of rows in the map}) - (\text{number of rows in the sonar}) + 1) \times (\text{number of columns in the map}) - (\text{number of columns in the sonar}) + 1]\)

E.g. if the map is of the size 501x501 and the sonar of the size 10x10 the mean value matrix will be of the size 492x492. The mean value matrix is numbered in the same way as the sonar data.
9.2.2. Data sequence

A special sequence is used in the download file in order to simplify the upload of data from the external memory to the correlator. This sequence makes it possible to address the memory in numerical order. It also makes it easier to upload the new data for the correlation areas. Every correlation area has 100 data and one mean, which is to be compared with the sonar data (when the sonar is of the size 10x10). When the next correlation area is to be compared there are only 11 data that are new, since the correlation areas overlap. Therefore a lot of upload time would be saved if the old data could be stored. The sequence described below makes it possible to upload only the new data for every correlation area and still have the advantage of addressing the memory in numerical order. The downside to this sequence is that the amount of data downloaded increases to about 10 times the amount of data in the map. The amount of data will increase linear with the size of the sonar data i.e. a sonar size of 5x5 pixels will increase the amount of data downloaded about 5 times. This increases the download time from the PC to the correlator. But since the map and mean data can be downloaded before the sonar data is measured, which is when the critical time for the correlation begins, this is acceptable.

The following section is a detailed description of the download sequence of the map and mean data. The map data is numbered in the same way as the sonar data.

1. First send the first 100 map data column vice (1,2,3,4,5,6,7,8,9,502,503.....4518,4519).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>502</td>
<td>1003</td>
<td>1504</td>
<td>2005</td>
<td>2506</td>
<td>3007</td>
<td>3508</td>
<td>4009</td>
<td>4510</td>
<td>5011</td>
<td>5512</td>
<td>6013</td>
<td>6514</td>
<td>7015</td>
<td>7516</td>
</tr>
<tr>
<td>2</td>
<td>503</td>
<td>1004</td>
<td>1505</td>
<td>2006</td>
<td>2507</td>
<td>3008</td>
<td>3509</td>
<td>4010</td>
<td>4511</td>
<td>5012</td>
<td>5513</td>
<td>6014</td>
<td>6515</td>
<td>7016</td>
<td>7517</td>
</tr>
<tr>
<td>3</td>
<td>504</td>
<td>1005</td>
<td>1506</td>
<td>2007</td>
<td>2508</td>
<td>3009</td>
<td>3510</td>
<td>4011</td>
<td>4512</td>
<td>5013</td>
<td>5514</td>
<td>6015</td>
<td>6516</td>
<td>7017</td>
<td>7518</td>
</tr>
<tr>
<td>4</td>
<td>505</td>
<td>1006</td>
<td>1507</td>
<td>2008</td>
<td>2509</td>
<td>3010</td>
<td>3511</td>
<td>4012</td>
<td>4513</td>
<td>5014</td>
<td>5515</td>
<td>6016</td>
<td>6517</td>
<td>7018</td>
<td>7519</td>
</tr>
<tr>
<td>5</td>
<td>506</td>
<td>1007</td>
<td>1508</td>
<td>2009</td>
<td>2510</td>
<td>3011</td>
<td>3512</td>
<td>4013</td>
<td>4514</td>
<td>5015</td>
<td>5516</td>
<td>6017</td>
<td>6518</td>
<td>7019</td>
<td>7520</td>
</tr>
<tr>
<td>6</td>
<td>507</td>
<td>1008</td>
<td>1509</td>
<td>2010</td>
<td>2511</td>
<td>3012</td>
<td>3513</td>
<td>4014</td>
<td>4515</td>
<td>5016</td>
<td>5517</td>
<td>6018</td>
<td>6519</td>
<td>7020</td>
<td>7521</td>
</tr>
<tr>
<td>7</td>
<td>508</td>
<td>1009</td>
<td>1510</td>
<td>2011</td>
<td>2512</td>
<td>3013</td>
<td>3514</td>
<td>4015</td>
<td>4516</td>
<td>5017</td>
<td>5518</td>
<td>6019</td>
<td>6520</td>
<td>7021</td>
<td>7522</td>
</tr>
<tr>
<td>8</td>
<td>509</td>
<td>1010</td>
<td>1511</td>
<td>2012</td>
<td>2513</td>
<td>3014</td>
<td>3515</td>
<td>4016</td>
<td>4517</td>
<td>5018</td>
<td>5519</td>
<td>6020</td>
<td>6521</td>
<td>7022</td>
<td>7523</td>
</tr>
<tr>
<td>9</td>
<td>510</td>
<td>1011</td>
<td>1512</td>
<td>2013</td>
<td>2514</td>
<td>3015</td>
<td>3516</td>
<td>4017</td>
<td>4518</td>
<td>5019</td>
<td>5520</td>
<td>6021</td>
<td>6522</td>
<td>7023</td>
<td>7524</td>
</tr>
<tr>
<td>10</td>
<td>511</td>
<td>1012</td>
<td>1513</td>
<td>2014</td>
<td>2515</td>
<td>3016</td>
<td>3517</td>
<td>4018</td>
<td>4519</td>
<td>5020</td>
<td>5521</td>
<td>6022</td>
<td>6523</td>
<td>7024</td>
<td>7525</td>
</tr>
<tr>
<td>11</td>
<td>512</td>
<td>1013</td>
<td>1514</td>
<td>2015</td>
<td>2516</td>
<td>3017</td>
<td>3518</td>
<td>4019</td>
<td>4520</td>
<td>5021</td>
<td>5522</td>
<td>6023</td>
<td>6524</td>
<td>7025</td>
<td>7526</td>
</tr>
<tr>
<td>12</td>
<td>513</td>
<td>1014</td>
<td>1515</td>
<td>2016</td>
<td>2517</td>
<td>3018</td>
<td>3519</td>
<td>4020</td>
<td>4521</td>
<td>5022</td>
<td>5523</td>
<td>6024</td>
<td>6525</td>
<td>7026</td>
<td>7527</td>
</tr>
<tr>
<td>13</td>
<td>514</td>
<td>1015</td>
<td>1516</td>
<td>2017</td>
<td>2518</td>
<td>3019</td>
<td>3520</td>
<td>4021</td>
<td>4522</td>
<td>5023</td>
<td>5524</td>
<td>6025</td>
<td>6526</td>
<td>7027</td>
<td>7528</td>
</tr>
<tr>
<td>14</td>
<td>515</td>
<td>1016</td>
<td>1517</td>
<td>2018</td>
<td>2519</td>
<td>3020</td>
<td>3521</td>
<td>4022</td>
<td>4523</td>
<td>5024</td>
<td>5525</td>
<td>6026</td>
<td>6527</td>
<td>7028</td>
<td>7529</td>
</tr>
<tr>
<td>15</td>
<td>516</td>
<td>1017</td>
<td>1518</td>
<td>2019</td>
<td>2520</td>
<td>3021</td>
<td>3522</td>
<td>4023</td>
<td>4524</td>
<td>5025</td>
<td>5526</td>
<td>6027</td>
<td>6528</td>
<td>7029</td>
<td>7530</td>
</tr>
</tbody>
</table>

2. Send the first mean data (1).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>493</td>
<td>985</td>
<td>1477</td>
<td>1969</td>
<td>2461</td>
<td>2953</td>
<td>3445</td>
<td>3937</td>
<td>4429</td>
<td>4921</td>
<td>5413</td>
<td>5905</td>
<td>6397</td>
<td>6889</td>
<td>7381</td>
</tr>
<tr>
<td>2</td>
<td>494</td>
<td>986</td>
<td>1478</td>
<td>1970</td>
<td>2462</td>
<td>2954</td>
<td>3446</td>
<td>3938</td>
<td>4430</td>
<td>4922</td>
<td>5414</td>
<td>5906</td>
<td>6398</td>
<td>6890</td>
<td>7382</td>
</tr>
<tr>
<td>3</td>
<td>495</td>
<td>987</td>
<td>1479</td>
<td>1971</td>
<td>2463</td>
<td>2955</td>
<td>3447</td>
<td>3939</td>
<td>4431</td>
<td>4923</td>
<td>5415</td>
<td>5907</td>
<td>6399</td>
<td>6891</td>
<td>7383</td>
</tr>
<tr>
<td>4</td>
<td>496</td>
<td>988</td>
<td>1480</td>
<td>1972</td>
<td>2464</td>
<td>2956</td>
<td>3448</td>
<td>3940</td>
<td>4432</td>
<td>4924</td>
<td>5416</td>
<td>5908</td>
<td>6400</td>
<td>6892</td>
<td>7384</td>
</tr>
<tr>
<td>5</td>
<td>497</td>
<td>989</td>
<td>1481</td>
<td>1973</td>
<td>2465</td>
<td>2957</td>
<td>3449</td>
<td>3941</td>
<td>4433</td>
<td>4925</td>
<td>5417</td>
<td>5909</td>
<td>6401</td>
<td>6893</td>
<td>7385</td>
</tr>
<tr>
<td>6</td>
<td>498</td>
<td>990</td>
<td>1482</td>
<td>1974</td>
<td>2466</td>
<td>2958</td>
<td>3450</td>
<td>3942</td>
<td>4434</td>
<td>4926</td>
<td>5418</td>
<td>5910</td>
<td>6402</td>
<td>6894</td>
<td>7386</td>
</tr>
<tr>
<td>7</td>
<td>499</td>
<td>991</td>
<td>1483</td>
<td>1975</td>
<td>2467</td>
<td>2959</td>
<td>3451</td>
<td>3943</td>
<td>4435</td>
<td>4927</td>
<td>5419</td>
<td>5911</td>
<td>6403</td>
<td>6895</td>
<td>7387</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>992</td>
<td>1484</td>
<td>1976</td>
<td>2468</td>
<td>2960</td>
<td>3452</td>
<td>3944</td>
<td>4436</td>
<td>4928</td>
<td>5420</td>
<td>5912</td>
<td>6404</td>
<td>6896</td>
<td>7388</td>
</tr>
<tr>
<td>9</td>
<td>501</td>
<td>993</td>
<td>1485</td>
<td>1977</td>
<td>2469</td>
<td>2961</td>
<td>3453</td>
<td>3945</td>
<td>4437</td>
<td>4929</td>
<td>5421</td>
<td>5913</td>
<td>6405</td>
<td>6897</td>
<td>7389</td>
</tr>
<tr>
<td>10</td>
<td>502</td>
<td>994</td>
<td>1486</td>
<td>1978</td>
<td>2470</td>
<td>2962</td>
<td>3454</td>
<td>3946</td>
<td>4438</td>
<td>4930</td>
<td>5422</td>
<td>5914</td>
<td>6406</td>
<td>6898</td>
<td>7390</td>
</tr>
</tbody>
</table>
3. Send the next 10 data in the next column. Column 11 in the example. Start sending from
the top. (5011,5012...5019,5020).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>502</td>
<td>1003</td>
<td>1504</td>
<td>2005</td>
<td>2506</td>
<td>3007</td>
<td>3508</td>
<td>4009</td>
<td>4510</td>
<td>5011</td>
<td>5512</td>
<td>6013</td>
<td>6514</td>
<td>7015</td>
<td>7516</td>
</tr>
<tr>
<td>2</td>
<td>503</td>
<td>1004</td>
<td>1505</td>
<td>2006</td>
<td>2507</td>
<td>3008</td>
<td>3509</td>
<td>4010</td>
<td>4511</td>
<td>5012</td>
<td>5513</td>
<td>6014</td>
<td>6515</td>
<td>7016</td>
<td>7517</td>
</tr>
<tr>
<td>3</td>
<td>504</td>
<td>1005</td>
<td>1506</td>
<td>2007</td>
<td>2508</td>
<td>3009</td>
<td>3510</td>
<td>4011</td>
<td>4512</td>
<td>5013</td>
<td>5514</td>
<td>6015</td>
<td>6516</td>
<td>7017</td>
<td>7518</td>
</tr>
<tr>
<td>4</td>
<td>505</td>
<td>1006</td>
<td>1507</td>
<td>2008</td>
<td>2509</td>
<td>3010</td>
<td>3511</td>
<td>4012</td>
<td>4513</td>
<td>5014</td>
<td>5515</td>
<td>6016</td>
<td>6517</td>
<td>7018</td>
<td>7519</td>
</tr>
<tr>
<td>5</td>
<td>506</td>
<td>1007</td>
<td>1508</td>
<td>2009</td>
<td>2510</td>
<td>3011</td>
<td>3512</td>
<td>4013</td>
<td>4514</td>
<td>5015</td>
<td>5516</td>
<td>6017</td>
<td>6518</td>
<td>7019</td>
<td>7520</td>
</tr>
<tr>
<td>6</td>
<td>507</td>
<td>1008</td>
<td>1509</td>
<td>2010</td>
<td>2511</td>
<td>3012</td>
<td>3513</td>
<td>4014</td>
<td>4515</td>
<td>5016</td>
<td>5517</td>
<td>6018</td>
<td>6519</td>
<td>7020</td>
<td>7521</td>
</tr>
<tr>
<td>7</td>
<td>508</td>
<td>1009</td>
<td>1510</td>
<td>2011</td>
<td>2512</td>
<td>3013</td>
<td>3514</td>
<td>4015</td>
<td>4516</td>
<td>5017</td>
<td>5518</td>
<td>6019</td>
<td>6520</td>
<td>7021</td>
<td>7522</td>
</tr>
<tr>
<td>8</td>
<td>509</td>
<td>1010</td>
<td>1511</td>
<td>2012</td>
<td>2513</td>
<td>3014</td>
<td>3515</td>
<td>4016</td>
<td>4517</td>
<td>5018</td>
<td>5519</td>
<td>6020</td>
<td>6521</td>
<td>7022</td>
<td>7523</td>
</tr>
<tr>
<td>9</td>
<td>510</td>
<td>1011</td>
<td>1512</td>
<td>2013</td>
<td>2514</td>
<td>3015</td>
<td>3516</td>
<td>4017</td>
<td>4518</td>
<td>5019</td>
<td>5520</td>
<td>6021</td>
<td>6522</td>
<td>7023</td>
<td>7524</td>
</tr>
<tr>
<td>10</td>
<td>511</td>
<td>1012</td>
<td>1513</td>
<td>2014</td>
<td>2515</td>
<td>3016</td>
<td>3517</td>
<td>4018</td>
<td>4519</td>
<td>5020</td>
<td>5521</td>
<td>6022</td>
<td>6523</td>
<td>7024</td>
<td>7525</td>
</tr>
<tr>
<td>11</td>
<td>512</td>
<td>1013</td>
<td>1514</td>
<td>2015</td>
<td>2516</td>
<td>3017</td>
<td>3518</td>
<td>4019</td>
<td>4520</td>
<td>5021</td>
<td>5522</td>
<td>6023</td>
<td>6524</td>
<td>7025</td>
<td>7526</td>
</tr>
<tr>
<td>12</td>
<td>513</td>
<td>1014</td>
<td>1515</td>
<td>2016</td>
<td>2517</td>
<td>3018</td>
<td>3519</td>
<td>4020</td>
<td>4521</td>
<td>5022</td>
<td>5523</td>
<td>6024</td>
<td>6525</td>
<td>7026</td>
<td>7527</td>
</tr>
<tr>
<td>13</td>
<td>514</td>
<td>1015</td>
<td>1516</td>
<td>2017</td>
<td>2518</td>
<td>3019</td>
<td>3520</td>
<td>4021</td>
<td>4522</td>
<td>5023</td>
<td>5524</td>
<td>6025</td>
<td>6526</td>
<td>7027</td>
<td>7528</td>
</tr>
<tr>
<td>14</td>
<td>515</td>
<td>1016</td>
<td>1517</td>
<td>2018</td>
<td>2519</td>
<td>3020</td>
<td>3521</td>
<td>4022</td>
<td>4523</td>
<td>5024</td>
<td>5525</td>
<td>6026</td>
<td>6527</td>
<td>7028</td>
<td>7529</td>
</tr>
<tr>
<td>15</td>
<td>516</td>
<td>1017</td>
<td>1518</td>
<td>2019</td>
<td>2520</td>
<td>3021</td>
<td>3522</td>
<td>4023</td>
<td>4524</td>
<td>5025</td>
<td>5526</td>
<td>6027</td>
<td>6528</td>
<td>7029</td>
<td>7530</td>
</tr>
</tbody>
</table>

4. Send the next mean data (493).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>493</td>
<td>985</td>
<td>1477</td>
<td>1969</td>
<td>2461</td>
<td>2953</td>
<td>3445</td>
<td>3937</td>
<td>4429</td>
<td>4921</td>
<td>5413</td>
<td>5905</td>
<td>6397</td>
<td>6889</td>
<td>7381</td>
</tr>
<tr>
<td>2</td>
<td>494</td>
<td>986</td>
<td>1478</td>
<td>1970</td>
<td>2462</td>
<td>2954</td>
<td>3446</td>
<td>3938</td>
<td>4430</td>
<td>4922</td>
<td>5414</td>
<td>5906</td>
<td>6398</td>
<td>6890</td>
<td>7382</td>
</tr>
<tr>
<td>3</td>
<td>495</td>
<td>987</td>
<td>1479</td>
<td>1971</td>
<td>2463</td>
<td>2955</td>
<td>3447</td>
<td>3939</td>
<td>4431</td>
<td>4923</td>
<td>5415</td>
<td>5907</td>
<td>6399</td>
<td>6891</td>
<td>7383</td>
</tr>
<tr>
<td>4</td>
<td>496</td>
<td>988</td>
<td>1480</td>
<td>1972</td>
<td>2464</td>
<td>2956</td>
<td>3448</td>
<td>3940</td>
<td>4432</td>
<td>4924</td>
<td>5416</td>
<td>5908</td>
<td>6400</td>
<td>6892</td>
<td>7384</td>
</tr>
<tr>
<td>5</td>
<td>497</td>
<td>989</td>
<td>1481</td>
<td>1973</td>
<td>2465</td>
<td>2957</td>
<td>3449</td>
<td>3941</td>
<td>4433</td>
<td>4925</td>
<td>5417</td>
<td>5909</td>
<td>6401</td>
<td>6893</td>
<td>7385</td>
</tr>
<tr>
<td>6</td>
<td>498</td>
<td>990</td>
<td>1482</td>
<td>1974</td>
<td>2466</td>
<td>2958</td>
<td>3450</td>
<td>3942</td>
<td>4434</td>
<td>4926</td>
<td>5418</td>
<td>5910</td>
<td>6402</td>
<td>6894</td>
<td>7386</td>
</tr>
<tr>
<td>7</td>
<td>499</td>
<td>991</td>
<td>1483</td>
<td>1975</td>
<td>2467</td>
<td>2959</td>
<td>3451</td>
<td>3943</td>
<td>4435</td>
<td>4927</td>
<td>5419</td>
<td>5911</td>
<td>6403</td>
<td>6895</td>
<td>7387</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>992</td>
<td>1484</td>
<td>1976</td>
<td>2468</td>
<td>2960</td>
<td>3452</td>
<td>3944</td>
<td>4436</td>
<td>4928</td>
<td>5420</td>
<td>5912</td>
<td>6404</td>
<td>6896</td>
<td>7388</td>
</tr>
<tr>
<td>9</td>
<td>501</td>
<td>993</td>
<td>1485</td>
<td>1977</td>
<td>2469</td>
<td>2961</td>
<td>3453</td>
<td>3945</td>
<td>4437</td>
<td>4929</td>
<td>5421</td>
<td>5913</td>
<td>6405</td>
<td>6897</td>
<td>7389</td>
</tr>
<tr>
<td>10</td>
<td>502</td>
<td>994</td>
<td>1486</td>
<td>1978</td>
<td>2470</td>
<td>2962</td>
<td>3454</td>
<td>3946</td>
<td>4438</td>
<td>4930</td>
<td>5422</td>
<td>5914</td>
<td>6406</td>
<td>6898</td>
<td>7390</td>
</tr>
</tbody>
</table>

5. Continue sending according to 3 and 4 until the last column is sent.
6. Send the last 10 data in the row under the rows already sent. Send the values from the left to the right. (246001,246502,...250009,250510).

7. Send the mean data positioned one row under the one already sent. (241574).

8. Send the 10 data from the next column to the left, column number 11 to the left of the last column. Start sending from the bottom. (245500,245499,....245492,245491).
9. Send mean data. (241082).

10. Move one step to the left and continue sending as described in 8 and 9 until column 1 is reached again.

11. Send the 10 first data in the rows below the rows already sent. Send from the left. (12, 513... 4020, 4521).

Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA
Kim Andersson and Filip Traugott MDH / FMV / FOI
12. Send the mean data positioned one row below the one last sent. (3)

<table>
<thead>
<tr>
<th>1</th>
<th>493</th>
<th>985</th>
<th>1477</th>
<th>1969</th>
<th>2461</th>
<th>2953</th>
<th>3445</th>
<th>3937</th>
<th>4429</th>
<th>4921</th>
<th>5413</th>
<th>5905</th>
<th>6397</th>
<th>6889</th>
<th>7381</th>
<th>7873</th>
<th>8365</th>
<th>8857</th>
<th>9349</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>494</td>
<td>986</td>
<td>1478</td>
<td>1970</td>
<td>2462</td>
<td>2954</td>
<td>3446</td>
<td>3938</td>
<td>4430</td>
<td>4922</td>
<td>5414</td>
<td>5906</td>
<td>6398</td>
<td>6890</td>
<td>7382</td>
<td>7874</td>
<td>8366</td>
<td>8858</td>
<td>9350</td>
</tr>
<tr>
<td>3</td>
<td>495</td>
<td>987</td>
<td>1479</td>
<td>1971</td>
<td>2463</td>
<td>2955</td>
<td>3447</td>
<td>3939</td>
<td>4431</td>
<td>4923</td>
<td>5415</td>
<td>5907</td>
<td>6399</td>
<td>6891</td>
<td>7383</td>
<td>7875</td>
<td>8367</td>
<td>8859</td>
<td>9351</td>
</tr>
<tr>
<td>4</td>
<td>496</td>
<td>988</td>
<td>1480</td>
<td>1972</td>
<td>2464</td>
<td>2956</td>
<td>3448</td>
<td>3940</td>
<td>4432</td>
<td>4924</td>
<td>5416</td>
<td>5908</td>
<td>6400</td>
<td>6892</td>
<td>7384</td>
<td>7876</td>
<td>8368</td>
<td>8860</td>
<td>9352</td>
</tr>
<tr>
<td>5</td>
<td>497</td>
<td>989</td>
<td>1481</td>
<td>1973</td>
<td>2465</td>
<td>2957</td>
<td>3449</td>
<td>3941</td>
<td>4433</td>
<td>4925</td>
<td>5417</td>
<td>5909</td>
<td>6401</td>
<td>6893</td>
<td>7385</td>
<td>7877</td>
<td>8369</td>
<td>8861</td>
<td>9353</td>
</tr>
<tr>
<td>6</td>
<td>498</td>
<td>990</td>
<td>1482</td>
<td>1974</td>
<td>2466</td>
<td>2958</td>
<td>3450</td>
<td>3942</td>
<td>4434</td>
<td>4926</td>
<td>5418</td>
<td>5910</td>
<td>6402</td>
<td>6894</td>
<td>7386</td>
<td>7878</td>
<td>8370</td>
<td>8862</td>
<td>9354</td>
</tr>
<tr>
<td>7</td>
<td>499</td>
<td>991</td>
<td>1483</td>
<td>1975</td>
<td>2467</td>
<td>2959</td>
<td>3451</td>
<td>3943</td>
<td>4435</td>
<td>4927</td>
<td>5419</td>
<td>5911</td>
<td>6403</td>
<td>6895</td>
<td>7387</td>
<td>7879</td>
<td>8371</td>
<td>8863</td>
<td>9355</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>992</td>
<td>1484</td>
<td>1976</td>
<td>2468</td>
<td>2960</td>
<td>3452</td>
<td>3944</td>
<td>4436</td>
<td>4928</td>
<td>5420</td>
<td>5912</td>
<td>6404</td>
<td>6896</td>
<td>7388</td>
<td>7880</td>
<td>8373</td>
<td>8865</td>
<td>9356</td>
</tr>
<tr>
<td>9</td>
<td>501</td>
<td>993</td>
<td>1485</td>
<td>1977</td>
<td>2469</td>
<td>2961</td>
<td>3453</td>
<td>3945</td>
<td>4437</td>
<td>4929</td>
<td>5421</td>
<td>5913</td>
<td>6405</td>
<td>6897</td>
<td>7389</td>
<td>7881</td>
<td>8375</td>
<td>8867</td>
<td>9357</td>
</tr>
<tr>
<td>10</td>
<td>502</td>
<td>994</td>
<td>1486</td>
<td>1978</td>
<td>2470</td>
<td>2962</td>
<td>3454</td>
<td>3946</td>
<td>4438</td>
<td>4930</td>
<td>5422</td>
<td>5914</td>
<td>6406</td>
<td>6898</td>
<td>7390</td>
<td>7882</td>
<td>8374</td>
<td>8868</td>
<td>9358</td>
</tr>
</tbody>
</table>

13. Return to 3 until the last row is sent.

9.3. Sending results

The results are sent to the PC during the calculation and according to the sequence described below.

1. Results are sent from the left to the right.

<table>
<thead>
<tr>
<th>1</th>
<th>493</th>
<th>985</th>
<th>1477</th>
<th>1969</th>
<th>2461</th>
<th>2953</th>
<th>3445</th>
<th>3937</th>
<th>4429</th>
<th>4921</th>
<th>5413</th>
<th>5905</th>
<th>6397</th>
<th>6889</th>
<th>7381</th>
<th>7873</th>
<th>8365</th>
<th>8857</th>
<th>9349</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>494</td>
<td>986</td>
<td>1478</td>
<td>1970</td>
<td>2462</td>
<td>2954</td>
<td>3446</td>
<td>3938</td>
<td>4430</td>
<td>4922</td>
<td>5414</td>
<td>5906</td>
<td>6398</td>
<td>6890</td>
<td>7382</td>
<td>7874</td>
<td>8366</td>
<td>8858</td>
<td>9350</td>
</tr>
<tr>
<td>3</td>
<td>495</td>
<td>987</td>
<td>1479</td>
<td>1971</td>
<td>2463</td>
<td>2955</td>
<td>3447</td>
<td>3939</td>
<td>4431</td>
<td>4923</td>
<td>5415</td>
<td>5907</td>
<td>6399</td>
<td>6891</td>
<td>7383</td>
<td>7875</td>
<td>8367</td>
<td>8859</td>
<td>9351</td>
</tr>
<tr>
<td>4</td>
<td>496</td>
<td>988</td>
<td>1480</td>
<td>1972</td>
<td>2464</td>
<td>2956</td>
<td>3448</td>
<td>3940</td>
<td>4432</td>
<td>4924</td>
<td>5416</td>
<td>5908</td>
<td>6400</td>
<td>6892</td>
<td>7384</td>
<td>7876</td>
<td>8368</td>
<td>8860</td>
<td>9352</td>
</tr>
<tr>
<td>5</td>
<td>497</td>
<td>989</td>
<td>1481</td>
<td>1973</td>
<td>2465</td>
<td>2957</td>
<td>3449</td>
<td>3941</td>
<td>4433</td>
<td>4925</td>
<td>5417</td>
<td>5909</td>
<td>6401</td>
<td>6893</td>
<td>7385</td>
<td>7877</td>
<td>8369</td>
<td>8861</td>
<td>9353</td>
</tr>
<tr>
<td>6</td>
<td>498</td>
<td>990</td>
<td>1482</td>
<td>1974</td>
<td>2466</td>
<td>2958</td>
<td>3450</td>
<td>3942</td>
<td>4434</td>
<td>4926</td>
<td>5418</td>
<td>5910</td>
<td>6402</td>
<td>6894</td>
<td>7386</td>
<td>7878</td>
<td>8370</td>
<td>8862</td>
<td>9354</td>
</tr>
<tr>
<td>7</td>
<td>499</td>
<td>991</td>
<td>1483</td>
<td>1975</td>
<td>2467</td>
<td>2959</td>
<td>3451</td>
<td>3943</td>
<td>4435</td>
<td>4927</td>
<td>5419</td>
<td>5911</td>
<td>6403</td>
<td>6895</td>
<td>7387</td>
<td>7879</td>
<td>8371</td>
<td>8863</td>
<td>9355</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>992</td>
<td>1484</td>
<td>1976</td>
<td>2468</td>
<td>2960</td>
<td>3452</td>
<td>3944</td>
<td>4436</td>
<td>4928</td>
<td>5420</td>
<td>5912</td>
<td>6404</td>
<td>6896</td>
<td>7388</td>
<td>7880</td>
<td>8372</td>
<td>8864</td>
<td>9356</td>
</tr>
<tr>
<td>9</td>
<td>501</td>
<td>993</td>
<td>1485</td>
<td>1977</td>
<td>2469</td>
<td>2961</td>
<td>3453</td>
<td>3945</td>
<td>4437</td>
<td>4929</td>
<td>5421</td>
<td>5913</td>
<td>6405</td>
<td>6897</td>
<td>7389</td>
<td>7881</td>
<td>8373</td>
<td>8865</td>
<td>9357</td>
</tr>
<tr>
<td>10</td>
<td>502</td>
<td>994</td>
<td>1486</td>
<td>1978</td>
<td>2470</td>
<td>2962</td>
<td>3454</td>
<td>3946</td>
<td>4438</td>
<td>4930</td>
<td>5422</td>
<td>5914</td>
<td>6406</td>
<td>6898</td>
<td>7390</td>
<td>7882</td>
<td>8374</td>
<td>8866</td>
<td>9358</td>
</tr>
</tbody>
</table>

2. When the last data in the row is sent the next row is sent from the right.

Return to 1 until all rows are sent.
9.4. Using multiple correlators

To make the calculation even faster the map can be divided into different sections and then calculated separately using multiple parallel correlators. This method is not used in the design presented in this report since the calculation is faster then the communication with the memory. However, when the internal memory on the FPGA is big enough to store the data for the correlation, faster calculation methods can be of interest.

Example: The map is divided into four equally sized sections, which are correlated separately by four parallel correlators.

The sections in the case with a map size of 501x501 pixels:
1. row 1->132 all columns
2. row 124->255 all columns
3. row 247->378 all columns
4. row 370->501 all columns

The number of rows in the sections can be calculated using the following formula:

\[
\frac{\text{(number of rows in map)} - \text{(number of rows in sonar)} + 1}{\text{(number of sections)}} + \text{(number of rows in sonar)} - 1.
\]

Note that the result has to be an integer if identical correlators are used. If the result is not an integer, the map should be resized to match the sections. This should not be a problem since only a few rows, which only represent a few meters, need to be added or, more probable, removed. Note also that the sonar image has to be quadratic.

Every section is treated as a separate map and downloaded in the sequence described in 9. The sections are sent “simultaneous” i.e. the first data in the first sequence is followed by the first data in the second sequence and so on.

![Diagram showing the different sections and the send sequence](image-url)

Figure 27. Description of the different sections.
10. Results

The results of the tests were very positive and fortify the hypothesis that a hardware accelerator has the capability of speeding up the calculations in a correlation. A couple of timing counters have been implemented in the design to find out the real time of some operations. The results show that the correlator has a calculation time of more than 100 times faster than the equivalent calculation on a standard PC.

Results:

- Time for calculation of one correlation area: 1.1 µs
- Time for loading new data for one calculation: ~ 450 ns
- Time for calculation of entire correlation: 266 µs (not including time for sending results)
- Parallel EPP communication: 0.06 MB/s
- Memory bandwidth: > 100 MB/s (in burst mode) ~50 MB/s (single bit mode)

The size of the design got bigger than calculated. The reason for this is the assignment of the arrays in the “CORR” block. Since the sequence used in the data flow requires tree different kinds of shifts in the arrays, large multiplexes are created for the assignment of the arrays. These multiplexes require a lot of logic, but since the time is what is important in this project this is a reasonable trade-off for a faster data flow.

The first correlation design tested had total parallel calculation, which means that the design correlates a whole correlation area in one or two clock cycles (as shown in Figure 10). This design got very big and the implementation took too much space on the test card. The reason for this is the large number of lines needed to connect the arrays with the adders combined with the multiplexes used for the assignment. However, the design can be used in later projects since it will fit in larger FPGA’s.

The external sdram memories were much faster than first anticipated, when using the burst mode, a bandwidth over 100 MB/s can be reached. Therefore the conclusion that the memories should be a system delay has to be revised. When using external sdram memories combined with burst mode, very fast correlators can be designed without the use of the internal block ram memory. The on chip memory is still faster though. The problem is the refresh cycles, which are necessary for the memories to store its data. The refresh cycles make the memories inaccessible and therefore the data flow can be stopped if not this problem is addressed. In this design the use of on chip buffers and interrupt signals has reduced this problem to a minimum.

It is clear that faster communication between the PC and the correlator is needed. The parallel communication used in this design has its limitations. Although it has specified speeds of up to 2 MB/s, measurements have shown that speeds of about 0.06 MB/s are more accurate for the design used in this project. These low results probably depend on the software design and since the main objective was the hardware design, no optimising of the software has been made. One might also argue that in the future a much faster communication will be used i.e. PCI for built in systems and FIRE-WIRE for portable systems.
11. Summary

The calculation in this design has reached speeds of over 100 times faster than the same calculations on a regular PC. The results indicate that very fast correlators can be designed when using a hardware calculator instead of software. If the design is optimised and larger FPGA:s are used, speeds of about 1000 times faster should be possible. Then it would be possible to use more data in the correlation, for example using higher resolution in the maps and sonar images or including other parameters in the algorithm such as bottom types. This would increase the accuracy in the correlation i.e. a more accurate position in the navigation system.

The biggest delay in the system is the communication between the PC and the correlator. In this project the EPP parallel communication protocol is used. In spite of the relatively high bandwidth of a maximum of 2 MB/s the communication is the delay in the system, especially when the interrupt pin is used as a busy signal by the correlator since the PC always has to begin with reading the status register before a write or read cycle can begin. The only solution to this is using a faster communication and therefore future designs should be implemented with a faster interface like PCI or FIRE-WIRE.

An interesting part of the design is the external sdram memories. As shown earlier this design uses dynamic memories instead of the more commonly used static memories (more common in hardware designs). The dynamic memories are smaller in size and can store more data than the static memories. The downside is the data refresh cycles needed for the memories to preserve its data. When using sdram memories in real-time designs this issue has to be considered. In the report two different solutions to the problem is presented.

The project has shown that an implementation of a terrain correlation navigation system in a hardware design significantly reduces the calculation time. Tests have shown that the calculation time can be reduced with over 100 times and that future designs can be even faster. Therefore the conclusion is that the correlation algorithm is well suited for implementation in hardware systems.
12. Future work

There are some things that should be further examined, investigated or constructed. Some of them are:

- Looking deeper into which communication is best for this type of construction.
- A deeper investigation on how and in what extent the Xilinx system generator in Simulink can be used to construct a terrain navigation system.
- A benchmark test showing the size, speed power consumption in a system constructed with System Generator compared with manually written VHDL design.
- When an implementation is done it should be tested in real scenarios.
- Out of the results of this project a prototype design should be built and tested in real scenarios.
Implementation of a terrain navigation correlator for an underwater vehicle in a FPGA

Kim Andersson and Filip Traugott MDH / FMV / FOI

13. List of references

[1] Nygren Ingemar
Terrain navigation for underwater vehicles.
Defence Research Establishment. Division of Systems and Underwater Technology. Report, FOA-R--00-01777-313--SE,
December, 2000,
ISSN 1104-9154 (In Swedish)

[2] Lundin Caroline
A study of navigation system for autonomous underwater vehicle. (In Swedish)

Utveckling av terrängnavigeringssystem för autonoma undervattensfarkoster.
Chalmers University of Technology, Div of Underwater Technology.
Göteborg, Sweden 1989
Report 35:1089
ISSN 021-6172 (In Swedish)

Navigation system for an autonomous underwater vehicle.
Master Thesis in Automatic Control, Royal institute of technology.
Stockholm, Sweden 2000
FMV ProjSjö 34434-15782

[5] Nygren Ingemar
Defence Research Establishment. Division of Systems and Underwater Technology. Report, FOA-R--99-01298-313--SE,
December, 2000,
ISSN 1104-9154 (In Swedish)

Multibeam sonar image matching for terrain-based underwater navigation.
UMR CNRS/Université Montpellier, Montpellier, France.

Lost2: A terrain based underwater positioning system result from sea trials.


[10] Son-Cheol Yu, Tamaki Ura, Teruo Fujii and Hayato Kondo  
Institute of Industrial Science, University of Tokyo

*Image analysis for airborne reconnaissance and missile applications.*  
*Corresponding author.

Xilinx, inc.

Xilinx, inc.


*Parallel port complete, programming, interfacing & using the PC’s parallel port.*  

*On-chip monitoring of single- and multiprocessor hardware real-time operating systems.*  
Department of computer engineering, Mälardalen University, Västerås, Sweden.

*VHDL för konstruktion.*  

*Lecture notes in hardware algorithms.*  
Department of electronics, Mälardalen University, January 2001, Västerås, Sweden.

*Analogue-to-digital wireless communication via Bluetooth using FPGA/VHDL system design.*  