Halmstad University Post-Print

Radar signal processing using pipelined optical hypercube interconnects

Håkan Forsberg, Bertil Svensson, Anders Åhlander and Magnus Jonsson

N.B.: When citing this work, cite the original article.

©2001 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Forsberg H, Svensson B, Åhlander A, Jonsson M. Radar signal processing using pipelined optical hypercube interconnects. In: Proceedings of the 15th International Parallel and Distributed Processing Symposium: IPDPS 2001 : abstracts and CD-ROM. Los Alamitos, California: IEEE Computer Society Press; 2001. p. 2043-2052.

DOI: <http://dx.doi.org/10.1109/IPDPS.2001.925201>

Copyright: IEEE

Post-Print available at: Halmstad University DiVA
<http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-2745>

Radar Signal Processing Using Pipelined Optical Hypercube Interconnects

Håkan Forsberg¹⁾, Bertil Svensson^{1,3)}, Anders Åhlander²⁾, and Magnus Jonsson³⁾

¹⁾Department of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden ²⁾Airborne Radar Division, Ericsson Microwave Systems, Mölndal, Sweden ³⁾School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden

Abstract

In this paper, we consider the mapping of two radar algorithms on a new scalable hardware architecture. The architecture consists of several computational modules that work independently and send data simultaneously in order to achieve high throughput. Each computational module is composed of multiple processors connected in a hypercube topology to meet scalability and high bisection bandwidth requirements. Free-space optical interconnects and planar packaging technology make it possible to transform the hypercubes into planes. Optical fan-out reduces the number of optical transmitters and thus the hardware cost. Two example systems are analyzed and mapped onto the architecture. One 64-channel airborne radar system with a sustained computational load of more than 1.6 TFLOPS, and one ground-based 128-channel radar system with extreme inter-processor communication demands.

1 Introduction

Computational and communication complexity are two key issues in embedded signal-processing (ESP)-systems, e.g. radar applications. The sustained computational load in several proposed algorithms for such systems is in the range of 1 GFLOPS to 50 TFLOPS [1]. As a consequence, several processors must work together and thus the inter-processor communication increases. Moreover, the data transfer time grows fast if wrong network topology is chosen, especially in systems with frequent use of all-to-all communication structures. The choice of a high-speed network is therefore essential. Other requirements that must be fulfilled in ESP-systems are real-time processing, low power consumption, small physical size, and multimode operation.

A solution to reduce the time spent in inter-processor data communication is to use optical interconnects. These high-speed links increase the bandwidth over the cross section that divides a network into two halves, i.e. the

bisection bandwidth (BB) [2]. High BB, in turn, reduces the time spent in all-to-all data transfers.

Optical technologies can also reduce the physical size, and increase the scalability to ensure multimode operation. For instance, optical free-space interconnects have been used to connect the processing elements between two electrical planes, and by that form an advanced scalable network architecture for massively parallel computing [3-5]. Several such networks have also been connected through optical starcouplers and wavelength demultiplexing (WDM) [6]. However, computing systems using other optical technologies than free-space do not offer the best promise [7].

In [8], we presented the basic ideas of a new hardware architecture. Firstly, we introduced a computational module as a unit composed of multiple processors connected in a hypercube topology to meet scalability and high bisection bandwidth requirements. Secondly, we showed how it was possible to transform a hypercube topology into a plane by using free-space optical interconnects and planar packaging technology. Thirdly, we showed how optical fan-out either could enhance the communication structure or reduce the number of transmitters and thus the hardware cost and physical size. Finally, we showed how it was possible to massively interconnect multiple planes. The result was a powerful, general, and compact system suitable for embedded signal

Figure 1: An example of the new hardware architecture, a pipelined system of optical planar-packaged hypercubes.

This work is financed by Ericsson Microwave Systems within the Intelligent Antenna Systems Project.

processing applications. The system also proved to be scalable in all three physical dimensions. See Figure 1 for an example of the new hardware architecture, a pipelined system of optical planar-packaged hypercubes.

This work evaluates this new hardware architecture by analyzing the mapping aspects of two different radar applications on the same kind of system. The first application is a 64-channel airborne space-time adaptive processing (STAP) radar with a sustained computational load of more than 1.6 TFLOPS. As a consequence, several processors must be used to reduce the per-processor load. The second application is a ground-based 128-channel radar using both broadcasting and personalized all-to-all communication structures. This, altogether, tests the performance of the inter-processor communication network in our new architecture.

2 Hypercubes

The hypercube is a flexible architecture with many attractive features. Many other well-known topologies like meshes, butterflies and shuffle-exchange networks can be embedded into it [9]. Also rings with even number of nodes and certain balanced trees can be embedded [10]. Another feature that makes the hypercube attractive is that the bisection bandwidth (BB) scales linearly with the number of processors, and thus higher dimensions lead to very high BB [8]. This property is very important in embedded signal processing systems where the time spent in all-to-all communication must be kept low.

A disadvantage of the hypercube is its complexity. It requires many and long wires since not only the nearest neighbors are connected to each other. However, by using optical properties in planar packaging technology, the interconnect complexity can be greatly reduced [8]. As a result, high dimensional hypercubes used in high-performance systems reduce the time spent in inter-processor communication.

2.1 Communication in hypercubes

Hypercubes can be used to implement several algorithms requiring all-to-all communication, e.g. matrix transposition, vector reduction, and sorting [11].

In radar systems two such all-to-all communication structures are of great importance. The first one relies on personalized all-to-all communication, i.e. all nodes send a unique data set to all other nodes. Personalized all-to-all communication is used to efficiently redistribute data between computational units that process information in different dimensions. This redistribution, referred to as corner turning in the radar literature, is, actually, from a mathematical point of view, a matrix transposition [2]. The second communication structure relies on broadcasting, i.e. all nodes need to copy information from

themselves to all other nodes, or a subset thereof. Broadcasting can, for instance, be used in the constant false alarm ratio (CFAR) stage in certain radar algorithms. This will be shown in Section 3.2.1.

In [8], we derived an expression for the time it takes to perform a corner turn in an optically interconnected hypercube using cost-saving beamsplitters. Here, the same expression is given (except for the substitution of Eq. 2 in [8]). A full corner turn takes:

$$\frac{\frac{1}{2} D_{size} \log_2(P)}{PR_{link,eff}} \quad (1)$$

seconds. D_{size} is the total size of the chunk of data to be redistributed, P is the number of processors in the hypercube, and $R_{link,eff}$ is the efficient transfer rate of a single link in one direction when overhead is excluded, e.g. message startup time. The equation above is based on the hypercube transpose algorithm described by Foster [11]. In this algorithm, data is only exchanged in one dimension at a time. Using this one-dimension-at-a-time procedure is a direct result of the cost saving “single-port” behavior, i.e. the beamsplitters used to reduce the number of transmitters. Note, however, that beamsplitters allow each node to transmit the same data to more than one neighbor at the same time. This is an extra feature compared to single-port communication where a node only can send and receive on one of its ports at the same time. In addition, each node in our architecture is also capable of receiving different data from different neighbors at the same time, i.e. similar to a multi-port behavior. Moreover, since the one-port algorithm chosen here is the same as the SBT-routing scheme described by Johnson and Ho [12], we are within a factor of two from the lower bound of one-port all-to-all personalized communication.

In broadcasting, the data transfer time for one-port communication is minimized if one dimension is routed at a time, i.e. the same principle as above, and all nodes use the same scheduling discipline [12]. Using this principle, each node copies its own amount of data M to its first neighbor (along the first dimension), and simultaneously receives M amount of data from the same neighbor. Next time, each node copies its own data and the data just received from the first neighbor, to the second neighbor (along the second dimension), and simultaneously receives $2M$ amount of data. This procedure is repeated over all dimensions in the hypercube. Thus each node has to send and receive:

$$\sum_{i=0}^{\log_2(P)-1} 2^i M = (P-1)M \quad (2)$$

amount of data. M is the data size in each node that has to be copied to all other nodes in the hypercube, and P is the

number of processors (nodes). Since each node has an efficient transfer rate of $R_{link,eff}$, broadcasting will take:

$$\frac{(P-1)M}{R_{link,eff}} \quad (3)$$

seconds. Note, however, that this equation is only valid if we consider the nodes as single-port. In reality, as described above, one copy of data from one node can actually be distributed to all $\log_2(P)$ neighbors at the same time, and each node can actually receive data from all its neighbors at the same time. The equation above should therefore not be considered as the optimal for this architecture, but good enough for its purpose. More investigations need to be carried out to find the optimal broadcasting algorithm for our new architecture. This issue is not discussed further in this paper.

For optimal multi-port personalized communication on hypercubes, see [13,14]. For optimal multi-port broadcasting communication on hypercubes, see [13].

2.2 Hypercubes in planar waveguides

In [8], we showed how a 6D-hypercube was merged into a planar waveguide made of glass or transparent semiconductor based substrate. This substrate serves as a light traveling medium as well as a motherboard for surface mounted optoelectronic and electronic chips [15]. Beamsplitters and other micro-optical devices can also be attached on both sides of the substrate. In Figure 2, it is shown how a 6D-hypercube is created from lower dimensional hypercubes and transformed into a plane. To the left in this figure, the topological view of the respective hypercube is shown. Note that the thick lines in Figure 2c consist of six interconnects each. To the right in Figure 2, it is shown how a 3D-hypercube is created in one row, (Figure 2a), then, how two such rows form a 4D-hypercube (Figure 2b), and, finally, how six rows form a 6D-hypercube, (Figure 2c).

There are many reasons to fold optically interconnected 3D-systems into planes. As we already have mentioned, complicated network topologies can be transformed into these planes. As a result the time spent in inter-processor communication can be reduced. Other reasons are precise alignment, mechanical robustness, and the easiness to cool, test, and repair the optoelectronic and electronic circuits attached to the substrates [15,16].

2.3 Pipelined systems of hypercubes

If the required performance exceeds the computational load in one unit, i.e. a substrate, several units have to work in co-operation. Also, to make these units efficiently work together, massive interconnections are necessary. One way to interconnect several units is to place them in a succession as in Figure 1. The drawback of this placement

Figure 2: a) A 3D-hypercube, b) a 4D-hypercube, and c) a full 6D-hypercube transformed into an optical plane.

is that each plane can only send data forward and backward to the subsequent and the previous unit respectively. However, this arrangement fits the pipelined computational nature in most radar systems, and is therefore a good choice for such applications. Moreover, a pipelined system of hypercubes can, in fact, be partitioned in all three spatial dimensions. For instance, two adjacent 6D-hypercubes form a 7D-hypercube (Figure 3a), a plane

Figure 3: Some configurations of inter-plane hypercubes in the new architecture. a) A 7D-hypercube, b) four 5D-hypercubes.

Figure 4: One big substrate, an alternative implementation of the pipelined system of optical planar-packaged hypercubes. a) and b) represents different incoming data media sources, i.e. fiber and free-space respectively.

divided into four equal squares form four 4D-hypercubes, and, finally, two planes of four 4D-hypercubes each, can form four 5D-hypercubes together (Figure 3b) [8]. As a result, many modes of operation can be executed on the same system, and this plays a central role in radar systems.

An alternative to the arrangement in Figure 1 is shown in Figure 4. In this figure, the pipelined system of planar-packaged hypercubes is merged into one big rectangular unit. As can be seen, the maximum horizontal light bounce interval is the same as the farthest neighbor distance, and not the whole length of the substrate.

The advantages of one big unit are many, for instance, the light beams only travel in one material compared to two (open-air is the other), no temperature dependent displacement problems between different substrates occur, and we do not need to open up the substrates to allow the beams to propagate in and out between computational units, etc. On the other hand, the light beams must travel twice the distance within the substrate, and inflection must be evaluated if the substrate is very long. Furthermore, system expandability is also limited compared to the other implementation shown in Figure 1, where more planes are added if the system performance is inadequate.

A third equivalent system of pipelined hypercubes is shown in Figure 5. Note, however, that this square-shaped system can be regarded as a single plane as in Figure 1, although larger, i.e. as an 8D-hypercube. Yet, slightly more transfer channels must be added if an 8D-hypercube topology should be complete.

3 Radar applications systems

As applications systems, an airborne STAP-radar and a ground-based radar are chosen. The airborne system has extreme demands on the computational load and moderate requirements on the inter-processor communication. The ground-based radar, on the other hand, has extreme demands on the inter-processor communication and moderate requirements on the computational load. As a result, the new architecture must be capable of handling both high system load and high inter-processor data transfers.

A single processing element in both systems is assumed to have a sustained floating-point capacity of

Figure 5: Another equivalent system of pipelined hypercubes.

approximately 3 GFLOPS when all inter-processor data communication is excluded. In addition, no overlap between computation and communication is assumed since this makes the programming more difficult [2].

3.1 The airborne STAP-radar system

Space-time adaptive processing (STAP) is a technique used in radar systems to support clutter and interference cancellation in airborne radars [17]. However, the full STAP-algorithm is of little value for most applications since the computational workload is too high and it suffers from weak convergence [18]. Therefore, some kind of load-reducing and fast convergent algorithm is used. For instance the n^{th} -order doppler-factored STAP. This STAP algorithm is, in addition to the medium (1st-order) and the hard (3rd-order) real-time STAP benchmark described by Cain et al. [17], also used in this airborne case study. Though, the computational load is increased numerous times compared to the 3rd-order STAP benchmark mentioned above. The reason for this increase is manifold, e.g., 64 instead of 22 processing channels, a higher order doppler-factored STAP (5th-order compared to 3rd-order), and a higher sampling rate etc.

The following system parameters are assumed for the airborne radar system:

- 64 processing channels (L)
- 5th-order doppler-factored STAP (Q)
- 32.25 ms integration interval (INTI) (τ)
- 960 samples (range bins) (N_d) per pulse after decimation with a factor of four
- 64 pulses per INTI and channel (C_p)
- 8 Gbit/s efficient data transfer rate of a single link in one direction ($R_{link,eff}$)

Because of the real-time nature of the system, a solution must be sensitive to low latency. Therefore, we put up a latency requirement of 100 ms, i.e. a maximum latency of 3τ to perform all calculations in the STAP-chain from the input stage to the final stage.

In Figure 6, the algorithmic pipeline stages for the chosen STAP-algorithm is shown. The chain consists of six pipeline stages, namely, video-to-I/Q conversion, array calibration, pulse compression, doppler processing,

Figure 6: The algorithmic pipeline stages in the airborne STAP-radar system.

weights computation, and finally weights application. For details concerning each step, see Cain et al. [17].

Table 1 below shows the computational load in each stage. The load is measured in floating-point operations per integration interval (INTI) (and not per second). Note that all floating-point calculations are derived from equations in Cain et al. [17]. Note also that the array calibration and the pulse compression stages are combined in Table 1.

Clearly, the hardest stage to calculate is the weights computation (a factor of 100 times more calculations than the other stages).

<i>Pipeline stage</i>	<i>Flops per INTI</i>
Video-to-I/Q-conversion	$4.56 * 10^8$
Array cal. and pulse comp.	$4.51 * 10^8$
Doppler processing	$1.28 * 10^8$
Weights computation	$5.05 * 10^{10}$
Weights calculation	$1.57 * 10^8$

Table 1: Computational load in each pipeline stage in the airborne radar system (measured in floating-point operations per integration interval).

3.1.1 Hardware architecture analysis

Let us assume, for a short moment, that we only have one processor with its own memory. If we perform all calculations with one processor, we have to perform $5.17 * 10^{10}$ floating-point operations during one INTI. This corresponds to a sustained performance of more than 1.6 TFLOPS (Tera floating-point operations per second) and this is too high for a single processor. As a consequence, we must reduce the per-processor load by using several processors and by using the maximum allowed working time, i.e. the maximum latency (three INTIs). The extended working time is achieved by pipelining some computational parts in the chain. By using many processors, the time spent in inter-processor communication will be noticeable and must be included in the calculations.

Since the weights computation stage is most critical, we will start analyzing that. In this stage, QR-decompositions dominate the computational complexity [17]. (A QR-decomposition is a numerically stable method to triangularize matrices [19].) The total number of QR-decompositions to compute in the entire datacube depends on the chosen algorithm. In this case study, one QR-decomposition is performed on a matrix covering one fourth of all range samples in one pulse, and over all corresponding channels (lobes), see Figure 7. This division requires, however, that the datacube is redistributed from a doppler oriented view to a range oriented view, i.e., we have to perform a corner turn in either the doppler processing stage or in the weights computation stage. Since the computational load is almost two magnitudes higher in the weights computation stage, we avoid doing the corner turn here. Also, to avoid extremely high inter-processor communication, we avoid calculating a single QR-decomposition on more than one processor. This means that 256 are the maximum number of processors to use, to calculate the weights. To reduce the per-processor load even further, we can use the system scalability and divide the computational work on two working chains, see Figure 8. In this Figure, every other datacube (odd numbered) to be processed follows the upper arrow, arrow a), to the dark colored group of processors. Similarly, the even numbered datacubes follow the lower arrow, arrow b), and is processed by the light colored group of processors. Note that each group of

Figure 7: Distribution of QR-decompositions in the datacube. The dark block corresponds to a matrix used in one QR-decomposition.

Figure 8: Two alternating working chains in the weights computation stage extend the working time and reduce the per-processor load.

processors in Figure 8 consists of eight 5D-hypercubes, i.e. 256 processors each. By dividing the computational work on two working chains, we can extend the computational time on a single datacube twice, to two INTI and thus reduce the per-processor work to the half. If we include the load in weights application into the weights computation stage, we need to perform $5.07 * 10^{10}$ Flops on 256 processors during a time of 2τ , i.e. a sustained per-processor floating-point performance of 3.07 GFLOPS, which is fully acceptable.

In the rest of the computational stages, i.e. the video-to-I/Q conversion, array calibration, pulse compression, and the doppler processing stage, we must perform altogether a total of $1.03 * 10^9$ Flops during one INTI (the remaining time of the maximum latency) minus the time it takes to perform two corner turns, see Figure 6, and minus the time it takes to distribute data to all processors in the weights computation stage.

To be able to calculate the corner turn time, we must know the size of the datacube. The total number of samples used in every integration interval in the algorithm is $LN_d C_p$. Since every sample is complex and the real and imaginary part both are 32 bit, the total size (D_{size}) of the datacube is $\approx 252 \text{ Mbit}$. As a result, it will take, $t_{CT} = 1.47 \text{ ms}$ to perform a corner turn on a 6D-hypercube with 64 processors ($P=64$), and 0.86 ms on a 7D-hypercube with 128-processors, according to Equation 1 and the system parameters given above.

Next, we have to calculate the time it takes to distribute data to correct cluster of 5D-hypercubes in the weights calculation stage, i.e. either among path a) or b) in Figure 8. First, we have to fold the datacube, to match the 5D-hypercube size. This time calculation is equivalent to Equation 1, except that we only move data among one direction in one dimension, i.e., we replace $\log(P)$ with 1 and P with $P/2$. If we start from a 6D-hypercube, we only have to fold data once, but if we starts from a 7D-hypercube, we have to add the time it takes to fold the data from a 7D- to a 6D-hypercube first. Next, we have to move all data to the first 5D-hypercube in the CFAR-chain, which in turn must move 7/8 to the next 5D-hypercube etc. This data movement can, however, be pipelined, i.e. as soon as the first hypercube receives its first data, it starts to forward this data to the next cube etc. The total time to distribute data to all 5D-hypercubes from

Figure 9: Final airborne radar system, one 6D-hypercube and sixteen 5D-hypercubes, i.e. 576 processors.

a 6D-hypercube and a 7D-hypercube is, therefore, $t_D = 1.47 \text{ ms}$ and 1.72 ms respectively. The time left to calculate $1.03 * 10^9$ Flops in a 6D-hypercube is thus 27.84 ms ($\tau - 2 t_{CT} - t_D$), i.e., a sustained per-processor floating-point performance of 578 MFLOPS. This is well below the per-processor load needed in the weights computation stage. As a result, using a 7D-hypercube in the rest of the chain is not necessary. (The per-processor load using a 7D-hypercube is 279 MFLOPS.)

The final airborne system consists, therefore, of nine pipelined optical substrates, i.e. 576 processors, see Figure 9. The operation will be as follows:

1. Preprocessing, doppler processing, and two corner turns are performed on the same 6D-hypercube.
2. If the datacube is numbered odd, fold it and distribute it to the upper cluster of eight 5D-hypercubes (arrow a) in Figure 9. If the datacube is numbered even, fold it and distribute it to the other cluster of 5D-hypercubes (arrow b) in Figure 9. This folding and distribution is carried out in the same time interval as step 1.
3. Next, weights computation and application are both performed on the same working cluster of eight 5D-hypercubes and during a time period equal to 2 INTIs.

3.2 The ground-based radar system

As already mentioned, the ground-based 128-channel radar system is less floating-point challenging as the airborne system. However, the inter-processor communication demands are higher. Both personalized and broadcasting all-to-all communication occurs. The following system parameters are assumed for the ground-based radar system:

- 128 processing channels (L)
- 400 kHz max pulse rep. freq. (f_{PRF})
- 40 ms integration interval (INTI) (τ)
- 6.25 Msample per second and channel (N_s)
- 8 Gbit/s efficient data transfer rate of a single link in one direction ($R_{link,eff}$)

Figure 10: The algorithmic pipeline stages in the ground-based radar system.

In Figure 10, the algorithmic pipeline stages for the chosen algorithm is shown. The chain consists of six pipeline stages, namely, digital beamforming, pulse compression, doppler processing, envelope detection, constant false alarm ratio (CFAR), and extraction. The computational load for all but the extraction stage is shown in Table 2. The CFAR-stage reduces data greatly, thus the extractor neither needs much computational power nor much communication time compared to the other stages. Therefore, no specific calculations are presented here, and one can actually assume that the extractor stage can be calculated during the last part of the CFAR stage time period. As in the airborne case, the load is measured in Flops per INTI. Note, however, that the INTI here is 40 ms compared to 32.25 ms as in the airborne case. Moreover, during a sampling period of 40 ms with a sample rate of 6.25 Msample per second and channel, the size of the datacube reaches respectful volumes, which in turn requires a high speed inter-processor communication network. The maximum latency is 3τ , i.e. 120 ms. All floating-point calculations are derived from equations in [19]. Note, however, that the ground-based sample radar system described by Taveniku and Åhländer [19], is not the same as here. The system here has 128 channels compared to 64, four times higher maximum pulse repetition frequency (PRF), and a CFAR-algorithm that is heavier from a communication point of view, but an INTI time that is doubled.

The purpose of the CFAR-process is to reduce the number of possible targets in each INTI, by only allowing

Pipeline stage	Flops per INTI
Digital beamforming	$1.12 * 10^9$
Pulse compression	$4.10 * 10^9$
Doppler processing	$2.20 * 10^9$
Envelope detection	$1.28 * 10^8$
CFAR	$1.10 * 10^{10}$

Table 2: Computational load in each pipeline stage in the ground-based radar system (measured in floating-point operations per integration interval).

a constant number of false items during a given time [20]. This process can be carried out in different ways. Seen from a communication view, the simplest CFAR-method only works in one dimension, usually in the range, and the hardest method works in several dimensions, with the neighborhood defined as a volume [20]. In addition, many different CFAR-techniques can be used in every communication case, and the computational load is usually not a problem. As a consequence, many designers have to choose CFAR-method based on the speed of the inter-processor network and not on the processor performance. Here, however, the choice of CFAR-method is not critical, since our network is designed for fast communication. We choose, therefore, a method based on ordered statistics-CFAR, where the surrounding neighbors in all three dimensions (pulse, range, and channel) are ordered in amplitude. The cell under test (CUT) is considered as a possible target if its value, multiplied with a certain constant, is larger than k neighbor cells [19]. In this case, the neighborhood is a $7 \times 7 \times 7$ volume, i.e. k is 342. This also means that each cell has to be distributed to all other nodes that calculate ordered statistics on a CUT belonging to the cell's neighborhood. For more information concerning the other stages in the ground-based system, see [20].

3.2.1 Hardware architecture analysis

If we do as we did in the airborne system, i.e. calculate the total load of the system if only one processor is used, we end up with $1.85 * 10^{10}$ Flops per INTI. This corresponds to 464 GFLOPS and is too much for a single-processor solution. Therefore, we have to divide the computations on several processors and use the maximum available latency.

As can be seen in Figure 10, two corner turnings must be performed before the CFAR stage. At first, data is sampled per channel, i.e. each node receives data from one or several channels. However, digital beamforming works in the channel dimension. Therefore, we have to redistribute data in such way that each node takes care of all data from all ranges and channels in one or more pulses. In the same way, we have to perform a second corner turn before the doppler stage, since data is

Figure 11: The chosen CFAR-algorithm allows the amount of data in each node, M , to be evenly distributed over many planes, and thus reduce the time it takes to perform broadcasting. Here, the datacube is divided into three fractions.

processed among the pulse dimension in doppler processing.

The size of the datacube to be corner turned is $LN_s\tau$ samples. Every sample is complex and consists of 64 bits. D_{size} is, therefore, 2048 Mbit. As a result, it will take $t_{CT} = 12$ ms to perform a corner turn on a 6D-hypercube with 64 processors ($P=64$), and 7 ms on a 7D-hypercube with 128-processors, according to Equation 1 and the system parameters given above.

If we perform digital beamforming, pulse compression, doppler processing, and envelope detection during the same time period, we have to perform $7.55 * 10^9$ Flops during an interval of $\tau - 2t_{CT}$. This gives a sustained per-processor performance of 7.37 GFLOPS on a 6D-hypercube and 2.27 GFLOPS on a 7D-hypercube. Hence, we choose a 7D-hypercube.

In the CFAR-stage, as mentioned above, each cell has to be distributed to all other nodes that calculate ordered statistics on a CUT within the cell's neighborhood. This is not a trivial problem, and it is not a full broadcasting. However, even if it is not a full all-to-all data transfer that has to be carried out, we can at least guarantee that we are on the right side of the time limit if we calculate with a full broadcasting, i.e. all nodes copy data to all other nodes.

If we disregard a node's memory capacity as the limiting factor, the time it takes to perform a full broadcasting with $M = D_{size}/P$, on a 6D-hypercube, is 126 ms, according to Equation 3. This is way too much (even more than the maximum latency allowed). Note that D_{size} is only 1024 Mbit now, since the envelope detection stage has converted the complex samples to real 32-bit values. We, therefore, need to reduce the per-processor data transfer size, M , by dividing the datacube over more than one computing hypercube. We also extend the operational time by using several working chains in the CFAR-stage (as we did in the weights computation stage in the airborne system, see Figure 8). To distribute data to several planes will, of course, require more time. The overall communication time, however, will be reduced, since the time spent in broadcasting using several planes is greatly reduced.

Figure 12: To reduce the broadcast transfer cost as much as possible in the CFAR-stage, data must be divided either in the pulse dimension or in the range dimension depending on the shape of the datacube in the previous stage.

At first, this inter-plane data distribution seems to be a trivial problem; just divide each nodes data into equally parts, and transfer these plus the overlap needed forward, see Figure 11. But, since the datacube can be shaped into different forms (depending on the pulse repetition frequency), we can divide the datacube into the pulse dimension or in the range dimension, see Figure 12. This division is carried out in that dimension which gives lowest possible size in data overlap. This will also reduce the broadcasting time. Our policy is therefore:

1. If # range bins (B_R) < # pulse bins (B_P), divide among the pulse dimension, i.e. according to Figure 12a.
2. If $B_R \geq B_P$, divide among the range dimension, i.e. according to Figure 12b.

The maximum distribution and broadcasting time will appear when the number of range bins is equal to the number of pulse bins. The number of samples per channel during one INTI is $N_s\tau = 2.5 * 10^5$. This corresponds to $B_R = B_P = 500$. If the neighborhood is $7 \times 7 \times 7$, the overlap section in Figure 11 will be six bins. The overlap that has to be sent forward, δ , is thus three bins. The size for one overlap in the whole datacube is therefore:

$$o_{size} = \delta \min(B_R, B_P)L \quad (4)$$

This give us the maximum $o_{size} = 3 * 500 * 128 = 192,000$ samples or 6.144 Mbit.

The amount of data to be distributed if only two hypercube units are used is $1/2 D_{size} + o_{size}$. If three hypercube units are used, we first have to transmit $2/3 D_{size} + o_{size}$ to the intermediate unit, and then $1/3 D_{size} + o_{size}$ to the last unit. This last transmission will, however, be pipelined with the first. If even more clusters of hypercubes are used, all transmissions will be pipelined. The data distribution time to x clusters is therefore:

$$t_{dist}(x) = \frac{\frac{x-1}{x} D_{size} + o_{size}}{R_{link} P_{cluster}} ; x > 1 \quad (5)$$

where $P_{cluster}$ is the number of processors within one hypercube. Note, however, that the equation above is not valid if the hypercubes have been created from groups of two adjacent planes, e.g. two planes divided into two 5D-hypercubes each, are merged to two inter-plane 6D-hypercubes instead. The reason for this is that the bandwidth between two different inter-plane hypercubes in a chain is limited. In addition, the transmission time also increases if inter-plane hypercubes are used, since broadcasting must be performed over an extra (unnecessary) dimension. The broadcast time within a cluster is then (based on Equation 3):

$$t_{broadcast}(x) = \frac{(P_{cluster} - 1) \left(\frac{D_{size}}{x} + 2o_{size} \right)}{R_{link,eff} P_{cluster}} ; x > 1 \quad (6)$$

Note that an intermediate broadcasting unit must share o_{size} data with both the previous and the next unit, hence the double o_{size} term above. Total time left to calculate the CFAR is then:

$$t_{left}(x) = t_{period} - t_{dist}(x) - t_{broadcast}(x) ; x > 1 \quad (7)$$

where t_{period} is the maximum time period to use in the CFAR stage. As mentioned above, we will use several computational chains to extend the working time. Note, however, that it is undesirable to use more than two working chains here, since t_{period} is always less than 2τ if the maximum latency is 3τ and the other stages work during 1τ , and thus only two working chains can be busy at the same time. If, however, the maximum latency was longer, e.g. 5τ , more working chains could be busy at the same time. Apart from that, a maximum latency of 3τ means that the only suitable configuration in the CFAR-process is to use two working chains of 5D-hypercubes each. t_{period} will then be 2τ minus the time it takes to fold data from a 7D-hypercube to a 5D-hypercube. The folding time for a 1024 Mbit datacube from a 7D- to a 5D-hypercube is 3.00 ms, according to the equation used in the previous discussion concerning folding in Section 3.1.

Number of 5D-hypercubes in the working chain (x)	Per-processor load in GFLOPS
2	14.96
3	3.64
4	2.07
5	1.45

Table 3: Sustained per-processor load in GFLOPS in the CFAR-stage.

Figure 13: Final ground-based radar system, one 7D-hypercube and eight 5D-hypercubes, i.e. 384 processors.

Using all equations above give us the expression for the sustained per-processor load:

$$CPU_{load}(x) = \frac{\left(\frac{CFAR_{Flops}}{x P_{cluster}} \right)}{t_{left}} ; x > 1 \quad (8)$$

where $CFAR_{Flops}$ is the number of floating-point operations per INTI in CFAR found in Table 2. In Table 3, the per-processor load for a chain of two to five 5D-hypercube working units are shown. Since it is undesirable to exceed a per-processor load of 3 GFLOPS, we chose a system with four 5D-hypercubes. The sustained per-processor load is then 2.07 GFLOPS, which is well below the unwanted limit. As a consequence, the extraction stage can hopefully be calculated during the same time period.

The final ground-based system consists, therefore, of six pipelined optical substrates, i.e. 384 processors, see Figure 13. The operation will be as follows:

1. Digital beamforming, pulse compression, doppler processing, envelope detection, and two corner turns are performed on a 7D-hypercube during the first INTI.
2. Fold the datacube twice (from 7D to 5D). Prepare to divide it among the pulse or range dimension depending on the shape of the datacube in the previous stage, and finally, distribute the fractions to the upper cluster of four 5D-hypercubes (arrow a) in Figure 13, if the datacube is numbered odd. If the datacube is numbered even, distribute it to the other cluster of 5D-hypercubes (arrow b) in Figure 13.
3. Compute the CFAR and the extraction stage on the same cluster as described above during the rest of the time available.

4 Conclusions

In this paper, we evaluated the mapping of two different radar applications on a new powerful hardware architecture suitable for embedded signal processing. The architecture consists of several massively interconnected

hypercubes to meet requirements on high throughput, high scalability, high bisection bandwidth, and high versatility.

To challenge the architecture, the choice of applications included both high system load and high inter-processor data-transfer load. An airborne STAP-radar application challenged the architecture in terms of computational load. With a sustained per-processor performance of slightly more than 3 GFLOPS, a total of 576 processors, and a bisection bandwidth of more than 1 Tbit/s, the system was capable of meeting all set requirements. In addition, the total time spent in non-overlapping inter-processor communication was below five percent. Therefore, to challenge the architecture in terms of inter-processor communication, a ground-based radar application was chosen. This 128-channel application spends nearly half of the time in communication between processors. To meet the requirements in this ground-based radar, a total of 384 processors are needed. The maximum sustained per-processor performance is 2.27 GFLOPS.

It can be noted that solutions that are non-optimal, in the sense that there is no overlap between computation and communication, put higher demands on the architecture. However, not putting so much effort into optimizing overlap, makes the software easier to develop, thus increasing engineering efficiency. On the other hand, if more suitable mapping of the algorithms are developed (at the expense of higher complexity), more powerful systems can be built using this new hardware architecture.

References

- [1] W. Liu and V. K. Prasanna, "Utilizing the power of high performance computing", *IEEE Signal Processing Magazine*, vol. 15, no. 5, Sept. 1998, pp. 85-100.
- [2] K. Teitelbaum, "Crossbar tree networks for embedded signal processing applications", *Proceedings of Massively Parallel Processing using Optical Interconnections*, MPPOI'98, Las Vegas, NV, USA, June 15-17, 1998, pp. 200-207.
- [3] A. Louri and H. Sung, "An optical multi-mesh hypercube: A scalable optical interconnection for massively parallel computing", *Journal of Lightwave Technology*, vol. 12, no. 4, April 1994, pp. 704-716.
- [4] A. Louri and H. Sung, "3D optical interconnects for high-speed interchip and interboard communications", *IEEE Computer*, Oct. 1994, pp. 27-37.
- [5] A. Louri and H. Sung, "An efficient 3D optical implementation of binary de Bruijn networks with applications to massively parallel computing", *Proceedings of Massively Parallel Processing using Optical Interconnections*, MPPOI'95, San Antonio, TX, USA, Oct. 23-24, 1995, pp. 152-159.
- [6] A. Louri and C. Neocleous, "Incrementally scalable optical interconnection network with a constant degree and constant diameter for parallel computing", *Applied Optics*, vol. 36, no. 26, 10 Sept. 1997, pp. 6594-6604.
- [7] H. M. Ozaktas, "Towards an optimal foundation architecture for optoelectronic computing", *Proceedings of Massively Parallel Processing using Optical Interconnections*, MPPOI'96, Maui, HI, USA, Oct. 27-29, 1996, pp. 8-15.
- [8] H. Forsberg, M. Jonsson, and B. Svensson, "A scalable and pipelined embedded signal processing system using optical hypercube interconnects", *Twelfth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2000)*, Las Vegas, NV, USA, Nov. 6-9, 2000, pp. 265-272.
- [9] D. E. Culler, J. P. Singh, with A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1999.
- [10] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, inc., Englewood Cliffs, NJ, USA, 1989.
- [11] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison Wesley Publishing Company, Inc., Reading, MA, USA, 1995.
- [12] S. L. Johnsson and C-T. Ho, "Optimum broadcasting and personalized communication in hypercubes", *IEEE Transactions on Computers*, vol. 38, no 9, Sept. 1989, pp. 1249-1268.
- [13] D. P. Bertsekas, C. Özveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes", *Journal of Parallel and Distributed Computing*, no. 11, 1991, pp. 263-275.
- [14] S. L. Johnsson and C-T. Ho, "Optimal communication channel utilization for matrix transposition and related permutations on binary cubes", Technical Report, No. TR-16-92, Parallel Computing Group, Center for Research in Computing Technology, Harvard University, MA, USA, Aug. 1992.
- [15] J. Jahns, "Integrated free-space optical interconnects for chip-to-chip communications", *Proceedings of Massively Parallel Processing using Optical Interconnections*, MPPOI'98, Las Vegas, NV, USA, June 15-17, 1998, pp. 20-23.
- [16] J. Jahns, "Planar packaging of free-space optical interconnections", *Proceedings of the IEEE*, vol. 82, no. 11, Nov. 1994, pp. 1623-1631.
- [17] K. C. Cain, J. A. Torres, and R. T. Williams, "RT-STAP: Real-time space-time adaptive processing benchmark", MITRE Technical Report, The MITRE Corporation, Center for Air Force C3 Systems, Bedford, MA, USA, 1997.
- [18] R. Klemm, "Introduction to space-time adaptive processing", The Institution of Electrical Engineers (IEE), Savoy Place, London WC2R 0BL, UK, 1998.
- [19] M. Taveniku and A. Åhlander, "Instruction statistics in array signal processing", Research Report, Centre for Computer Systems Architecture, Halmstad University, Sweden, 1997.
- [20] A. Åhlander, *Using Multiple SIMD Architectures for Multi-Channel Radar Signal Processing*, Licentiate Thesis, Department of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden, 1996.