Design Space Exploration for Optimal Allocation of Systolic Array for *n*-D Nested Loop Algorithm

Resmi R, B. Bala Tripura Sundari

Abstract— Systolic architectures work like stream architectures and can be exploited to accelerate data and compute bound applications. Such applications are represented as multi-level nested loop algorithms. In other words, nested loop algorithms can be speeded up by exploiting the inherent parallelism, and by mapping the computational tasks of the algorithm using a suitable mapping methodology on to array architecture. In this paper the mapping methodology that identifies a lower dimensional sub-space in the *n*-D problem space wherein lies the axis of up-dating of the computational equation is validated by implementing the allocation algorithm in C programming as applied to higher dimensional - *n*-D nested loop algorithms where n is greater than 2. The algorithms under consideration here are the 3-D matrix-matrix multiplication, 2-D spatial filtering algorithm which is a 4-D nested loop algorithm and 6-D FSBM. The input to the allocation algorithm is the dependence graph (DG) representation of the considered *n*-D nested loop problem. Effectiveness and validity of the mapping methodology for the allocation of the processing element (PE) array to the computational sub-space is obtained by comparing the number of PEs, ports, data reuse registers and memory read with an allocation to a random sub-space of the *n*-D problem space. It is found that the allocation of the computational sub-space of the *n*-D problem space (that lie in the identified computational sub-space) to the PE array of the systolic array architecture results in the optimal allocation as shown by the output of our allocation algorithm.

Index Terms— Design space exploration, Dependence graph, Computational subspace, Random subspace, Systolic array, Processing elements, Spatial filtering, Full search block motion estimation

1 INTRODUCTION

Allocation of the tasks of the computationally intensive algorithms represented by *n*-D algorithms to the processing elements of a parallel architecture is known as mapping. The various possible design solutions and tradeoffs for the problem under consideration is termed as the design space. The design space consists of many possible design solutions for the system and it becomes tough to determine the good designs. A good synthesis system has large design space which allows us to explore different tradeoffs between area, power, cost etc, for different designs which are to be explored and optimized. During the design space exploration (DSE) a conflicting situation always exists for the designer to concurrently maximize the accuracy of the process and minimize the time spent during DSE [3]. An automated synthesis can keep track of the design decisions made and their effects related to multi-parametric optimization has been carried out reduce conflicting situations. So automating the synthesis at higher levels of hierarchy has become important [1].

Different tasks and the priority of computations in the algorithm are represented by data flow graphs and dependence graphs. These representations allow the designer in the high level synthesis flow to represent the tasks in the algorithm and identify the parallel nodes of the algorithms that can be mapped to parallel architectures or folded pipelined architectures with an aim of arriving at optimal semi-parallel architectures [4], [8]. Systolic array is a scalable architecture which can be extended to any number of processors [11], [12]. The VLSI technology offers immense opportunities to develop parallel computations for both special purpose and general purpose devices. Its principles are compatible with VLSI technology characteristics, since systolic arrays are highly regular. Only algorithms with repetitive computations perform well and nested loop algorithms can be efficiently implemented by systolic array architecture.

In this paper the mapping of *n*-D nested loop problems onto systolic array representation is aimed at with the aid of mapping methodology which identify lower dimension sub-space of higher dimension problem using the computational sub-space [6] and followed by this, the results of the said method is compared with an allocation to a random sub-space of the *n*-D problem space [7]. The main objective here is to explore the design space exploration by mapping and allocation along various sub-spaces of the *n*-D problem for selecting proper design of the systolic array architecture with an objective of arriving at an optimal number of ports, data reuse, number of PEs. The possible designs are arrived at by exploring the various ways of allocation namely the computational sub-space and an allocation to a random sub-space of the *n*-D problem space. Application algorithms selected are matrix-matrix multiplication, and 2-D spatial filtering and 6-D FSBM which are higher dimension algorithms. There by effectiveness and validity of the computational sub-space method is obtained for higher dimensional algorithms.

This paper is organized as follows. Section 2 gives a brief overview on literature survey. Section 3 highlights methodologies to map nodes in dependency graph representation of nested loop algorithm to regular systolic array architecture. Section 4 describes the implementation details and results. Section 5 concludes the paper and scope of future work.

2 LITERATURE REVIEW

The techniques of high level synthesis (HLS) are to be required to be used to design an efficient exploration approach with the

ability to determine optimal/near-optimal scheduling solutions and module selection with significant speed and precision. The role of high level synthesis has become very important in the design of application specific processors (ASPs) and application specific integrated circuits (ASICs) [2]. There are many conflicting situations at the time of design space exploration. An efficient designer should maximize the accuracy of design space exploration and minimize the time spent for analysis and selecting best design for implementation. Many techniques are introduced which are capable of drastically reducing the number of variants to be analyzed for proper selection of the optimal design in the minimal time period. i.e., from the given specifications and system requirements the designer has to reduce complex design space into several feasible design solutions which satisfy required multi objectives. Multi-objective optimization is multiple criteria decision making, which is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously [3].

A priority factor method for selecting best design during design space exploration to select the resources for final organization of the design without requiring any graphs or tree arrangements to analyze the candidate variants [1]. Also a design methodology using an efficient multi-objective (area occupied, execution time and power consumption) exploration approach with the architecture synthesis process, that uses min-max analysis for selecting the optimal design during DSE is used [2].

3 METHODOLOGY

An appropriate representation of multi level nested loop algorithms will be essential for building design space exploration methods to arrive at array architectures and to determine the data-path of the arrays after successful mapping and allocation. One of them is the representation of the algorithm by using regular dependence graph (DG) and the mapping is done using linear mapping techniques. The edges in the dependence graph represent the precedence constraints. The DG corresponds to space representation where no time instant is assigned to any computation. Each node in DG represents iteration in nested loop. All the nodes collectively form iteration space. The interdependencies among the inherited and synthesized attributes at the nodes in a dataflow graph can be depicted by a dependence graph. The determination of the allocation and scheduling for HLS is very important to optimize the parameter of the design which are identified as follows for our design.

1. Number of PEs: Processing element is the basic building block of a systolic array. So reducing number of PEs lead to reduced area of architecture. Each processing element performs a sequence of operations on data that flows between them.
2. Number of ports: The data streams are entering and leaving through the ports of the array. Reducing the number of ports also helps to reduce the area of the designed systolic array.
3. Data reuse: Data reuse indicates the number of times that the data entered through every port is reused.
4. Memory read: Total number of reads through the external ports.
5. Number of cycles: Number of cycles required to perform an algorithm is analogous to speed of execution. So number of cycles should be reduced for faster execution.

3.1 Mapping Methodologies for *n*-D Nested Loop Algorithms

Computational Sub-Space Mapping Methodology

The window or sub-space for the allocation is most appropriately chosen so that there is more advantage for implementation in terms of data reuse, amount of memory reads and number of ports. In this methodology window assignment lies along the computational equation [6], [7] as shown in Fig. 1.

Random Sub-Space Method

As opposed to the above allocation, when the tasks in the DG are assigned to the PE array in a random manner not taking into consideration the updating direction of the computational expression as shown in Fig.2.

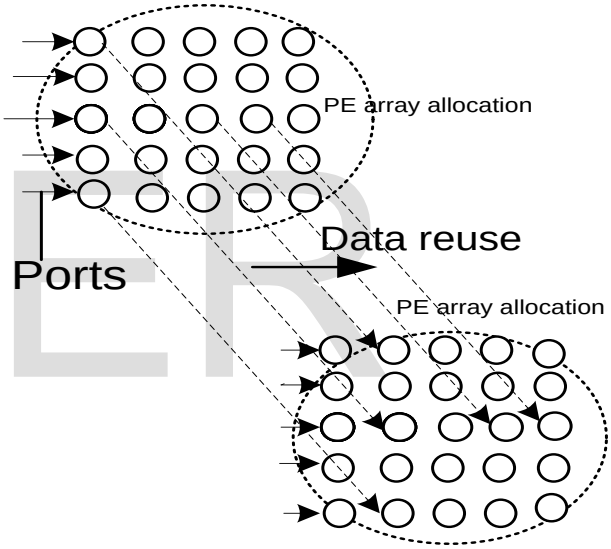

Fig. 1: Computational sub-space mapping methodology

3.2 Matrix-Matrix Multiplication (MMM)

Multiply 2 matrices A and B of order $(m \times n)$ and $(p \times q)$, and resulting C matrix of order $(m \times q)$ [9] and [10]. In the computational sub-space mapping methodology every data element in the first matrix is reused $(q-1)$ times. Similarly every element of thesecond matrix is reused (m-1) times. So total reuse = $(m \times n)(q-1) + (p \times q)(m-1)$ .Number of PEs is $(m \times q)$ and ports is reduced to $(m+n)$. The number of memory read is = $(m \times n) + (p \times q)$. In random *i-k* sub-space mapping methodology reuse is limited to $(m \times n)(q-1)$ and in random *j-k* sub-space mapping methodology reuse is $(p \times q)(m-1)$

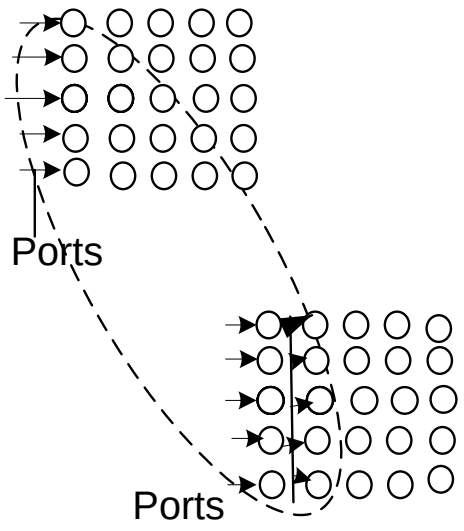

Fig. 2: Random *i-k* sub-space mapping methodology

3.3 2D Spatial Filtering (2-D SF)

To explain the 2-D spatial filtering convolution with 3 x 3 sobel mask and image frame is considered. The sobel operator performs a 2-D spatial gradient measurement on an image so as to emphasize regions of high spatial frequency that correspond to edges. The new image is formed by convolution of sobel mask with the zero padded images. The sobel mask is moved over the image and the convolution sum is computed. The computation of convolution sum is implemented by allocation of the corresponding sub-space to the PE array and compared with an allocation of PE array to an arbitrary subspace termed as the random sub-space method and the result is discussed in the section 4.

3.4 6-D Full Search Block Motion (FSBM) Estimation Algorithm

To explain the 6 level FSBM algorithms, consider two consecutive frames of a video, the current frame and the previously processed frame. A current frame is subdivided in to number of macro blocks. A typical video frame consist of $v \times h$ blocks, where v is the number of block in each row and h is the number of blocks in each column. Each block of a current frame, i.e., sub-frame is usually taken as a $N \times N$ pixel matrix, and this is labeled as $i \times j$ pixels. The pixels in the directions of search are labeled as m × n pixels. The identified sub-space for each pixel is $m \times n$ sub-space, which is allocated to the PEs. In higher dimensional algorithms such as 6-D FSBM the computational sub-space is more than one as shown in the results and the comparison between the two identifies the optimal allocation [6]

Method I: Computation Equation along *m*, *n* Direction

In this method the current frame is compared with reference frame in $(2\times p+1)$ directions, where $p \le N/2$; where N is the block size, here $N = 3$. Hence the comparison is done in 9 directions when $p = 1$. There are various functions to determine the direction of motion in the successive video frames, of which the most popular and less computationally expensive is Mean absolute difference (MAD). The MAD function for m, n direction is given by:

$$MAD(m,n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} c(i,j) - y(i+m, j+n) \quad (1)$$

Where, y is the previously processed frame and c is the current frame.

Method II: Computation Equation along i, j Direction

In this method the 3 x 3 sub-frame of the current frame is compared with reference frame along each direction separately. This comparison can be done by 9 PEs for one direction and similarly for remaining 9 directions. The MAD function for i, j direction is given by

$$MAD(i,j) = \sum_{m=0}^{n-1}\sum_{n=0}^{n-1} c(i,j) - y(i+m, j+n) \quad (2)$$

Reuse of data is greater in *m-n* direction comparing to *i-j* since every element in a block are reused.

4 RESULTS

Dependency graph representation of matrix-matrix multiplication, 2-D-spatial filtering and 6D FSBM in C is input to the allocation algorithm for different mapping methodologies namely computational sub-space and random sub-space and the outputs obtained are the number of PEs allocated to form the systolic array, number of ports, data reuse of registers, memory read and number of cycles.

4.1 Matrix Matrix Multiplication (MMM)

Table 1 gives the result of computational sub-space method for MMM problem. Table 2 gives the result of random *i-k* subspace method and Table 3 gives the result of random *j-k* subspace method.

Graphical Analysis (MMM)

Fig. 3, 4, 5 gives the comparison of data reuse, number of PEs, number of ports respectively. From the graphs it is found that in the computation sub-space mapping methodology data is efficiently reused, and also it is seen that the number of ports allocated for the former method is less than that for the latter method as shown in Fig.5 for a constant matrix size. In the case of number of PEs and cycles, a generalized conclusion can't be evolved since number of PEs and cycles purely depends on sub-space size which is a function of matrix dimension

Table 1: Result of computational sub-space method (matrix multiplication **order [*m*×*n*] with [*p*×*q*])**

Parameter	[2 × 2] × [2 ×2]	[2 × 3] × [3 × 2]	[3 × 3] × [3 × 3]	[4 × 4] × [4× 4]	[4 × 3] × [3 × 5]	[5 × 5] × [5 × 5]	[6× 6] × [6× 6]	[6 ×7] × [7 × 6]
Data reuse	8	12	36	96	93	200	360	420
Memory read	8	12	18	32	27	50	72	84
Number of PEs	4	4	9	16	20	25	36	36
Ports	4	4	6	8	9	10	12	12
Cycle	2	3	3	4	3	5	6	7

Table 2: Result of random *i-k* sub-space method (matrix multiplication of order [*m*×*n*] with [*p*×*q*])

Parameter	[2 × 2] × [2 ×2]	[2 × 3] × [3 × 2]	[3 × 3] × [3 × 3]	[4 × 4] × [4× 4]	[4 × 3] × [3 × 5]	[5 × 5] × [5 × 5]	[6× 6] × [6× 6]	[6 ×7] × [7 × 6]
Data reuse	4	6	18	48	48	100	180	210
Memory read	8	12	18	32	27	50	72	84
Number of PEs	4	6	9	16	15	25	36	42
Ports	6	9	12	20	18	30	42	49
Cycle	2	2	3	4	4	5	6	6

Table 3: Result of random *j-k* sub-space method (matrix multiplication of order [*m*×*n*] with [*p*×*q*])

Parameter	[2 × 2] × [2 ×2]	[2 × 3] × [3 × 2]	[3 × 3] × [3 × 3]	[4 × 4] × [4× 4]	[4 × 3] × [3 × 5]	[5 × 5] × [5 × 5]	[6× 6] × [6× 6]	[6 ×7] × [7 × 6]
Data reuse	4	6	18	48	45	100	180	210
Memory read	8	12	18	32	27	50	72	84
Number of PEs	4	6	9	16	12	25	36	42
Ports	6	9	12	20	15	30	42	49
Cycle	2	2	3	4	5	5	6	6

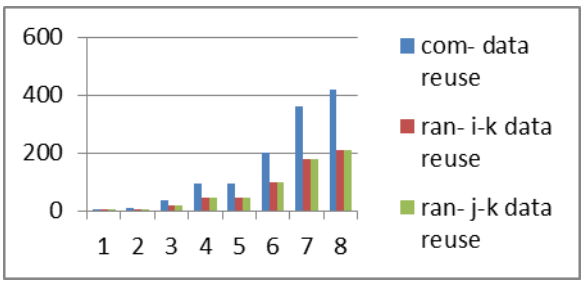

Fig. 3: Data reuse (*x* axis: Matrix dimension *y* axis: Reuse of registers)

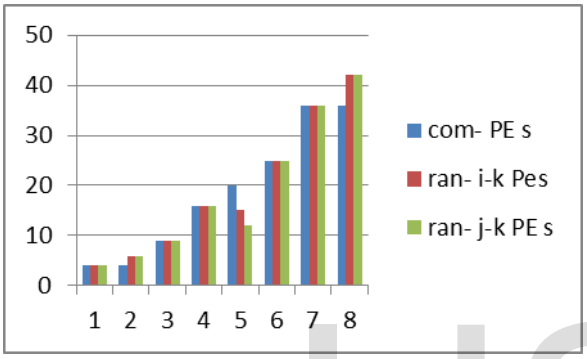

Fig. 4: Number of PEs (*x* axis: Matrix dimension *y* axis: Number of PEs)

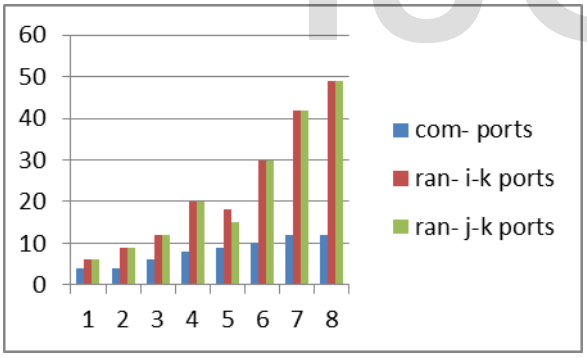

Fig. 5: Number of ports (*x* axis: Matrix dimension *y* axis: Number of ports)

So from the graphical analysis it can be observed that computational sub-space mapping methodology is better than random sub-space methodology in terms of data reuse, memory read and number of ports.

4.2 2-D Spatial Filtering (2-D SF)

Table 4 and 5 represents the result of computation sub-space method and random sub-space method for 2-D spatial filtering.

Table 4: Result of computational sub-space (2-D SF)

Image size	Data reuse	Memory reads	Cycles	PEs	Ports
4×4	60	49	16	9	6
5×5	104	62	25	9	6
6×6	160	105	36	9	6
7×7	228	124	49	9	6
8×8	308	185	64	9	6
9×9	400	210	81	9	6

Table 5: Result of random sub-space (2-D SF)

Image size	Data reuse	Memory reads	Cycles	PEs	Ports
4×4	20	89	16	9	6
5×5	39	115	25	9	6
6×6	64	201	36	9	6
7×7	95	239	49	9	6
8×8	132	361	64	9	6
9×9	175	411	81	9	6

Graphical Analysis (2-D SF)

In the case of spatial filtering also it is found that computational sub-space mapping methodology allows the maximum data reuse lesser access through ports as shown by reduction in the memory read. Fig. 6 shows the graphical analysis of data reuse in 2-D spatial filtering for computational and random sub-space.

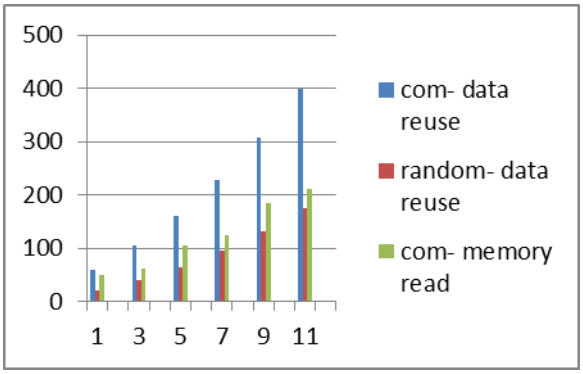

Fig. 6: Data reuse and memory read (*x* axis: Matrix dimension *y* axis: Number of cycles)

4.3 6-D FSBM

Table 6 shows the result of 6-D FSBM

Table 6: Result of FSBM

Parameter	*i-j* direction	*m-n* direction
Data reuse	252	648
Memory read	406	810
Number of PEs	9	9
Ports	18	18
cycles	81	81

Graphical Analysis (6-D FSBM)

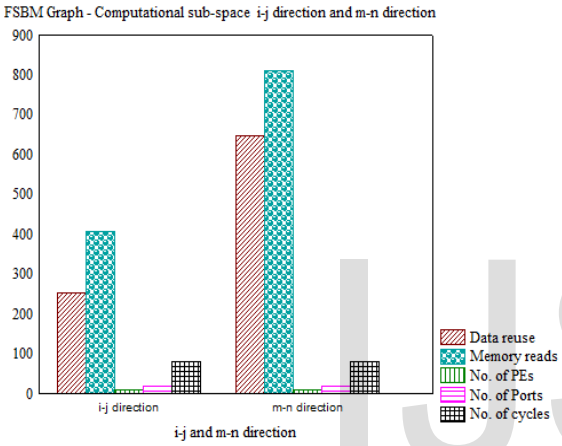

Fig. 7: FSBM Graphical analysis (*x* axis: different parameters *y* axis: values of parameters)

Fig. 7 represents the graphical analysis of different parameters. Among the two computational directions *i-j* direction is more appropriate in terms of optimal data reuse and optimal memory read. This is happening because in the *m-n* direction reuse is mainly for current frame pixels not for previously processed frame. So memory read will be very high because we have to enter all the surrounding pixels in the search frame for every current frame pixel.

5 Conclusion

Nested loop algorithms can be speeded up by exploiting the inherent parallelism, and mapping the computational tasks of the algorithm using a suitable mapping methodology on to array architecture called systolic array. In this project a mapping methodology that identifies a lower dimension sub-space of a higher dimensional problem is implemented using the technique of allocation. The lower dimensional sub-space is chosen to lie along the computational equation. The effectiveness and validity of the computational sub-space method is obtained by comparing the number of PEs, ports, data reuse registers and memory read with a random sub-space method for higher *n*-D algorithms. The graphs shows that the computational sub-space method is found to be better than the random sub-space method in terms of the amount of data reuse, memory read, number PEs, number of ports since they are important parameters for computationally intensive algorithms. The data reuse registers are found to be more in the former showing the advantage of less external memory access. For higher dimension algorithms there may be more than one computational direction. So according to the problem one should have to decide the best solution from the computational design space.

The future work is that a more generalized for different sizes of sub-frames in the FSBM algorithm and to study the PEs allocated and cycle time. Also a HLS synthesis could result in the RTL generation to obtain the exact data-path architecture. Also a reconfigurable systolic array architecture could be evolved using graph merging and multi-objective function, wherein the PE array configuration can be tailored towards different application domains. An array of customizable PEs that can be reconfigured for enabling the mapping of data-paths of both dependent and independent computational tasks in the *n*-D problem space could be obtained.

References

[1] Sengupta, Anirban, Reza Sedaghat, and ZhipengZeng. "A high level synthesis design flow with a novel approach for efficient design space exploration in case of multi-parametric optimization objective." Microelectronics Reliability 50, no. 3 (2010): 424-437.

[2] Sengupta, Anirban, Reza Sedaghat, and ZhipengZeng. "Multi-objective efficient design space exploration and architectural synthesis of an application specific processor (ASP)."Microprocessors and Microsystems 35, no. 4 (2011): 392-404.

[3] Sengupta, Anirban, Reza Sedaghat, and PallabiSarkar."Rapid exploration of integrated scheduling and module selection in high level synthesis for application specific processor design." Microprocessors and Microsystems 36, no. 4 (2012): 303-314.

[4] Sharma, Hrishikesh, and SachinPatkar."A design methodology for optimally folded, pipelined architectures in VLSI applications using projective space lattices."Microprocessors and Microsystems 37, no. 6 (2013): 674-683.

[5] Andriamisaina, Caaliph, Philippe Coussy, Emmanuel Casseau, and CyrilleChavet. "High-level synthesis for designing multimode architectures."Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 29, no. 11 (2010): 1736-1749.

[6] B. Bala Tripura Sundari, T.R. Padmanabhan, "A direct method for optimal VLSI realization of deeply nested *n*-D loop problems",Journal of Microprocessors and Microsystems,Embedded Hardware Design, Affiliated with Euromicro, Copyright © Elsevier B.V; DOI: 10.1016/ j.micpro.2013.04.003. Volume 37, Issues 6–7, pp.610-628, 2013.

[7] Sundari, B. Bala Tripura, and Varsha Krishnan. "Comparison of Configurations of Data Path Architecture Developed Using Template." In Proceedings of International Conference on Advances in Computing, pp. 539-548.Springer India, 2012.

[8] Jóźwiak, Lech, and Yahya Jan. "Design of massively parallel hardware multiprocessors for highly-demanding embedded applications."Microprocessors and Microsystems 37, no. 8 (2013): 1155-1172.

[9] Alias, Christophe, BogdanPasca, and AlexandruPlesco. "FPGA-specific synthesis of loop-nests with pipelined computational cores." Microprocessors and Microsystems 36, no. 8 (2012): 606-619.

[10] Kittitornkun, Surin, and Yu Hen Hu. "Mapping deep nested do-loop DSP algorithms to large scale FPGA array structures." Very Large Scale Integration

(VLSI) Systems, IEEE Transactions on 11, no. 2 (2003): 208-217.

[11] Kung, Sun Yuan. "VLSI array processors."Englewood Cliffs, NJ, Prentice Hall, 1988, 685 p. Research supported by the Semiconductor Research Corp., SDIO, NSF, and US Navy.1 (1988).

[12] Parhi,K.K."VLSI Digital Signal Processing Systems: Design and Implementation", John Wiley & Sons, 1999 J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," *Neurocomputing – Algorithms, Architectures and Applications,* F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)