VOL. 11, NO. 2, JANUARY 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

HIGH LEVEL SYNTHESIS FOR DESIGN SPACE EXPLORATION

Ranjini Krishnanunni K. and Bala Tripura Sundari B.
Electronics & Communication Engineering Department of Amrita Viswa Vidyapeetham, Coimbatore, Tamil Nadu, India
E-Mail: ranjoos1990@yahoo.co.in

ABSTRACT

In VLSI, design space exploration considering various constraints complex using conventional RTL design flow. The techniques of high level synthesis are useful in abstracting the design to a higher level than in the regular RTL design flow. The various hardware architectures possible need to be explored to bring out the design trade-offs in terms of parameters namely latency, critical path delay and resource utilization. The focus of the work presented here is to explore systolic array mapping methods with and without HLS transformations. Unfolding and pipelining are the HLS transformations applied on the DSP benchmark –FIR filter. Unfolding enhances the possibilities of concurrency in loops and pipeline architecture makes concurrency possible. Vivado HLS tool is used to explore the design space for random subspace mapping and computational subspace mapping and analyze their merits and demerits in terms of the design trade-off performance parameters when the design is mapped to Zynq architecture.

Keywords: design space exploration, RTL, high level synthesis, systolic array mapping, FIR filter, latency.

INTRODUCTION

Applications requiring VLSI design flow may be either be control intensive or data intensive. VLSI designs in digital signal processing applications are mostly data-intensive due to the presence of regular and iterative number of looping and repeated computations. The performance of such systems depends on the user specified constraints and performance requirement and hence there is a need to explore the different possible design considerations. The constraints needed to be taken into consideration are the data path delay, area utilization and latency. Reduced latency leads to better execution speed and less resource utilization leading to reduced area obviously leading to reduced power. A design is termed as better if it has better speed and reduced area [1]. Design space exploration (DSE) helps to adopt the best design to meet the specifications. High level synthesis techniques, which model a hardware implementable form from the obtained algorithmic model of the system [2], are used to abstract the best possible design space exploration.

Concurrent execution of computations is a solution to improve the execution speed of a system because more than one operation is executed at the same time. High level synthesis (HLS) transformations simplify the complexity of computations and exposes possibilities of concurrency among different iterations of the loop while pipeline architecture enables concurrency in design [3]. But, they may increase the number of resources used.
Another approach adopted to improve the performance of the system is by choosing a different data path for computation. The manner in which the computations are scheduled in the data path can alter the complexity of the designs and thereby can improve the execution speed of the system. Systolic array helps to schedule the data path computations in a defined manner so that data flow in a rhythm [3]. Mapping of the application onto the computational hardware units of the systolic array can be defined in terms of computational subspace mapping and random subspace mapping. Any mapping along the normal computation flow is called computational subspace and anything different from the normal flow is called random subspace [1].

The proposed work aims at combining systolic array mapping and HLS transformations to achieve a better solution to the development of an effective design of data-intensive DSP circuit using Vivado HLS tool. It also compares computational subspace mapping and random subspace before and after transformation to determine the advantages and disadvantages in terms of the parameters of delay, area and latency. Vivado HLS tool allows the designer to apply the transformations and target the application onto the specific FPGA to determine the hardware architectural parameters. Section II describes the HLS design flow and discusses the advantages of HLS transformations. Section III brings out the systolic array mapping methods. The overall methodology adopted in this work is brought out in section IV followed by the details of implementation of the methodology in section V and the results are presented in section VI.

ROLE OF HIGH LEVEL SYNTHESIS IN VLSI DESIGN

Digital design using high level synthesis mostly starts with the graphical representation of the design and it goes up to the generation of RTL diagram to determine the hardware complexity of the design [4]. High level synthesis effectively transforms the behavior of a model to its RTL form [5].

Design Representations

Digital design in DSP applications can be represented in the form of either dataflow graphs or dependency graphs. Dataflow graphs are highly helpful in obtaining an overall perspective of the process flow

occurring in the system. The processes and their dependencies are clearly depicted in dataflow graphs in terms of vertices and edges [3].

Complex computations may be modeled in a technically feasible manner using systolic array representations. Systolic array is formed by the interconnection of processing cells. The manner by which the algorithm is mapped to systolic array determines the performance and the possibilities of parallel computation [6]. We model systolic array using space diagrams.

Loop Transformations Techniques

Loops provide more possibilities of parallelism in many applications [7]. Retiming, unfolding are the major loop transformation techniques adopted in digital signal processing applications [3].

High complexity due to control hazards can be resolved using unfolding loop transformation/loop unrolling [3]. When different iterations of a loop are implemented in parallel, it is called iteration level parallelism [7]. Unfolding exposes more parallelism among different iterations. The unfolding factor f defines the number of replicas of the loop code in an unfolded manner. This reduces control hazards greatly at the expense of greater code length and greater number of flip-flops [3][8].

Pipeline architecture enables concurrency. Many inputs can be processed at the same time using this kind of architecture and hence this increases the throughput of the design [3]. This maximizes the efficiency in utilization of resources. This increases the use of storage elements.

RTL Representation

The HLS efficiently transforms the design to an architectural form called RTL structure which contains data-path (registers, multiplexers, functional units and buses) and a controller along with memory banks [9].

SYSTOLIC ARRAY MAPPING

Data flow in a determined fashion in systolic array which is helpful to design a system more efficiently. Space-time representations are required for systolic arrays to represent the design as an interconnection of processing units [3]. Figure 1 depicts the space representation of FIR filter. We know that in systolic array, the data flow in a rhythm and this rhythm is being defined via mapping. In space-time representations, each node represents a processing unit. The interconnections are same irrespective of the mapping. Mapping schedules when each computation must be carried out in the process [3]. Mapping may be done either using computational sub-space or random sub-space [1].

Computational Subspace Mapping

Computational sub-space is considered as the most advantageous and simple in terms of number of memory reads. In this sub-space, the computation goes along the normal flow. Each output sample may be calculated one after the other and next output sample computation will start only after the computation of the sample before is over. This is a direct mapping method and hence easy. Figure-2 shows the computational mapping on FIR filter.

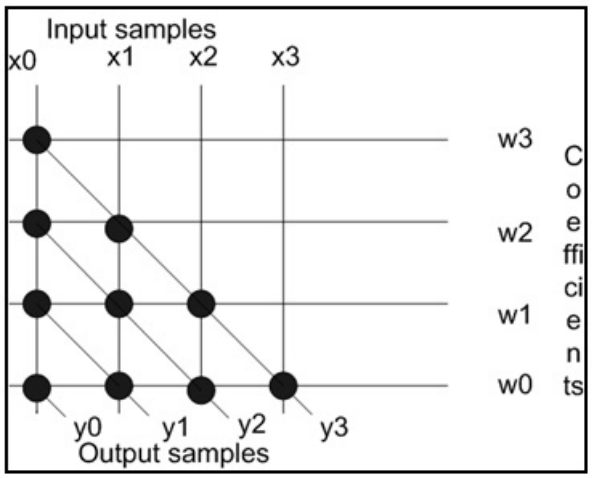

Figure 1. Space representation of FIR filter.

Random Subspace Mapping

The subspace of computation which is not along the line of computation is called as random subspace mapping. This is considered slightly complicated than the computational subspace mapping. Figure-3 shows one of many random subspace mappings done on the FIR filter. Figure-3 shows one of the random subspace mappings on FIR filter, which is used for the experiment.

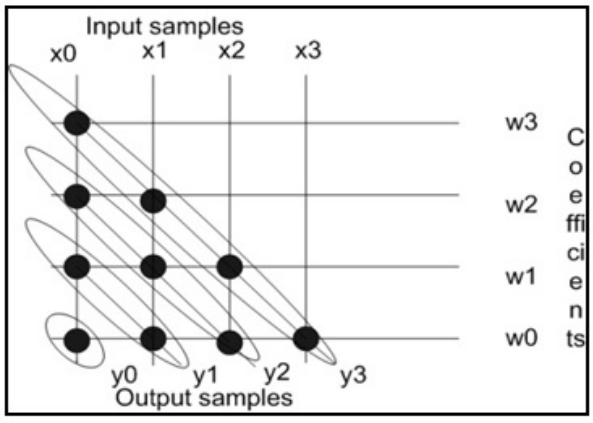

Figure-2. Computational subspace mapping on FIR filter.

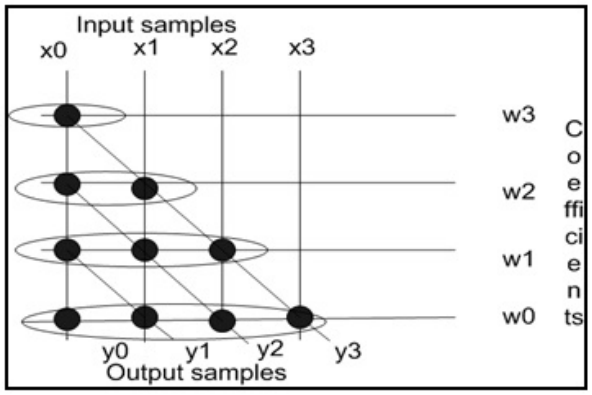

Figure 3. Random subspace mapping on FIR filter.

METHODOLOGY

Hence, first, dependence graph of the design is converted to its space-time representation. The mapping also defines the way in which computations must be carried out.

The steps to be carried out for the experiment:

- Represent the design application in space-time diagram
- Code in C/C++ based on the mapping methodology used for a) computational sub-space and b) random mapping
- Input the C/C++ file to Vivado HLS tool.
- Apply the following transformations (one by one) using the tool.
- Pipelining; 2. Unrolling
- Execute the design and determine the performance parameters namely resources utilized and latency in mapping and transformation applied followed by the RTL generation by targeting the Xilinx ISE.

EXPERIMENT

Finite impulse response (FIR) filter is used as the benchmark for the experiment. The design is coded in C language. Vivado HLS tool converts C modules to hardware modules [12]. First, the design with computational subspace mapping is coded and followed the steps as stated in methodology. Then, design with random subspace mapping as per Figure-3 is coded and implemented. The hardware chosen is Zynq architecture. The results of the Vivado HLS tool is input to obtain the architecture results by targeting the design on Zynq zynq_fpv6 xc7z015clg485-2 which has 160 DSP cores, 93800 flipflops and 46800 LUTs using Xilinx ISE.

RESULTS AND ANALYSIS

Results of the experiment are analyzed based on the transformation applied.

Mapping without any Transformation

Table-1 shows the results of mapping methods when no transformations are applied on 4-tap FIR filter and 15-tap FIR filter.

Table-1. Computational subspace mapping of 4-TAP and 15 TAP FIR filter using VIVADO HLS tools. FPGA used ZYNQ ZYNQ_FPV6 XC7Z015CLG485-8).

	Maximum latency (in clock cycles)	No of look-up tables(LUTs)	No of flip-flops	Critical Path Delay (in ns)
4-tap FIR filter				
Computational subspace	172	197	166	0.749
Random subspace	149	256	213	0.749
15-tap FIR filter				
Computational subspace	2266	199	235	1.154
Random subspace	1264	286	286	0.755

Computational subspace is along the line of computation where random subspace is not. From table-1, for high order 15-tap FIR filter. The following are to be noted:

- It is found that the critical path delay and latency less for random subspace.
- Random subspace requires more storage elements as tabulated in the results (no. of flip-flops are much more in random subspace) than computational subspace mapping , and hence it is inferred and this is due to the allocation of more intermediate registers/flip-flops that are a necessity in the random sub-space mapping methodology than in the computational subspace.

Mapping with Pipelining

Table-2 shows the results obtained when the 4-tap FIR filter and 15-tap FIR filter are pipelined and it is observed that

- Comparing Table-1 and Table-2, the number of flip-flops and LUTs used drastically increases when pipelining is applied on the design and latency has improved. Critical path delay is determined to be reduced for pipelined random subspace mapping.
- From Table-2, we can see that number of flip-flops and LUTs required is less in the case of pipelined random subspace than pipelined computational subspace.
- The reduction in the resources used due to random subspace mapping is more pronounced as the size of the design increase (15-tap filter design)
- The inference is that the random sub-space allocation methodology utilizes normal registers used for computation to form pipelining registers and hence the pipelining transformation shows lesser resources allocated.

Table-2. Result when pipelining is done to computational subspace and random subspace mappings of 4-TAP FIR filter using Vivado HLS tool. FPGA Used: ZYNQ ZYNQ_FPV6 XC7Z015CLG485-2).

	Maximum latency (in clock cycles)	No of Look-up tables	No of flip-flops	Critical path delay (in ns)
4-tap FIR filter				
Computational subspace	14	543	1024	0.927
Random subspace	11	456	906	0.923
15-tap FIR filter				
Computational subspace	18	5084	6797	1.154
Random subspace	17	520	733	1.15

Mapping with Unrolling

Table-3 shows the results of unrolling applied to the loops present in the 4-tap FIR filter and 15-tap FIR filter design.

- Comparing Table-1 and Table-3, shows that unrolling increase the resource allocation of a system design for computational sub-space more than for random sub-space.
- Table-3 shows that number of flip-flops used is less for unrolled random subspace mapping than unrolled computational subspace mapping and is reflected pronouncedly for higher order filter.
- The critical path delay is more for unrolled random subspace mapping when compared to computational subspace.

Table-3. Result of unrolled computational subspace and random subspace of 4-TAP FIR filter used Vivado HLS tool. FPGA used: zynq zynq_fpv6 xc7z015clg485-2).

	Maximum latency (in clock cycles)	No of look-up tables	No of flip-flops	Critical path delay (in ns)
4-tap FIR filter				
Computational subspace	10	543	862	0.9
Random subspace	11	433	658	0.927
15-tap FIR filter				
Computational subspace	17	6181	9963	1.154
Random subspace	17	328	669	1.197

Analysis based on RTL Schematic

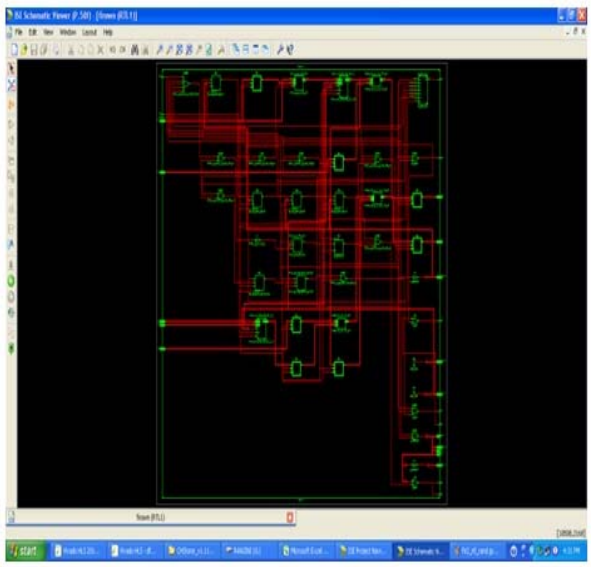

Figure-4. RTL view of 2-tap FIR filter when unrolling is applied on computational subspace mapping.

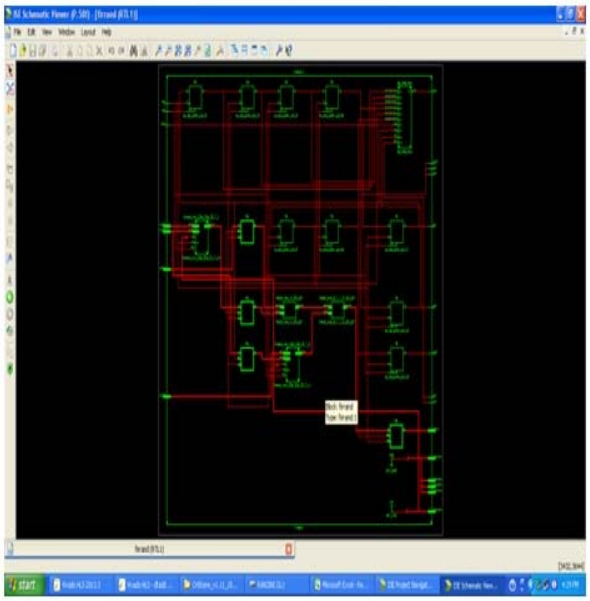

Figure-5. RTL view of 2-tap FIR filter when unrolling is applied in random subspace mapping.

Figure-4 and Figure-5 show the RTL view of 2-tap FIR filter when unrolling is applied. Here, it is observed that the resources used in random subspace is less than the computational subspace followed by unrolling of loops.

Graph-Based Analysis

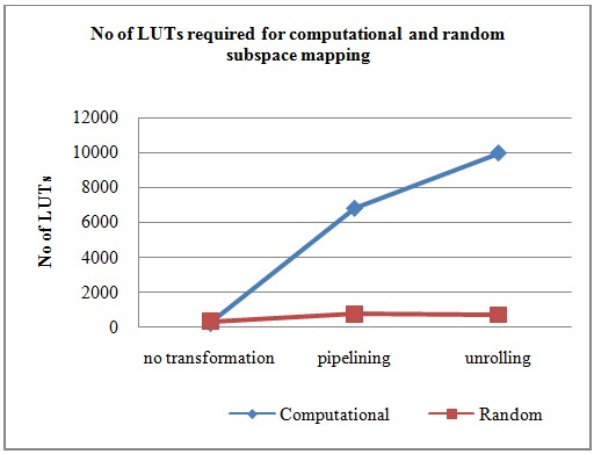

Figure-6. Graph showing no of LUTs required for computational and random subspace mapping done on 15-tap FIR filter.

The reduction of resources is due to reuse of resources that leads to more latency and critical path for unrolling transformation applied on random sub-space mapping as shown in the graphs in Figure-6 and Figure-7.

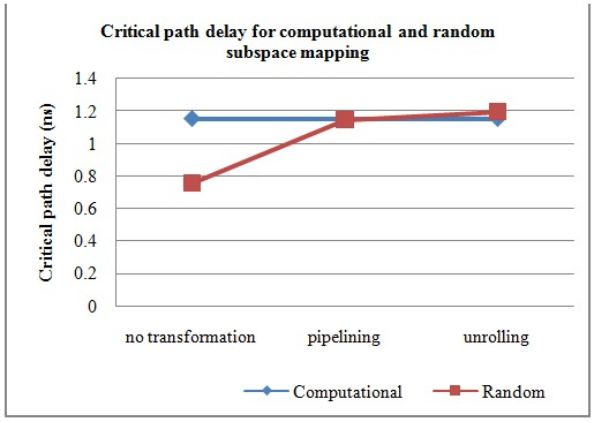

Figure 7. Graph showing critical path delay for computational and random subspace mapping done on 15-tap FIR filter

CONCLUSIONS

The paper explores design space exploration using HLS techniques to determine the trade-off of the performance parameters of the architecture obtained. Systolic array is considered as the basic hardware type and the mapping methods with and without transformations are applied to implement a compute intensive DSP application. Pipelining and unrolling are the HLS transformations considered here and the target hardware is Xilinx Zynq- zynq_fpv6 xc7z015clg485-2.

When unrolling is applied on the circuit, designer must decide based upon the requirements because even though unrolling on random subspace mapping reduces the allocation of number of resources, but it increases critical path delay. Reduced resource allocation which indicates more resource reuse reduces the area utilization of the FPGA device. But the increased critical path delay/ latency affects the performance of the system. Hence, there is a trade-off between allocation of number of resources, their re-use and speed of the processing of the compute intensive application by the architecture implemented. When the goal is parallel processing, we go in for unfolding on the computational sub-space mapping. Whereas random sub-space mapping is akin to pipelining that gives a reduced critical path and improves the latency. The improvement with further pipelining and unfolding on random sub-space mapping is to be investigated thoroughly based on the type of iterative compute intensive applications.

REFERENCES

[1] B. Bala Tripura Sundari and Varsha Krishnan, "Comparison of Configurations of Datapath architecture developed using template", in Proceedings of ICAdc, AISC 174, pp. 539-548, 2012.

[2] Selvaraj Ravi,M. Joseph, "High level test synthesis: A survey from synthesis process flow perspective", in ACM Trans. on design automation of electronic systems, Vol 19, No.4, Article 38, p.p 38-64,2014.

[3] Keshab. K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation",Wiley Student Edition, 2007.

[4] Alastair C. Murray, Richard V Bennett, Bjorn Franke and Nigel Topham, "Code Transformation and Instruction Set Extension", in ACM Transactions on embedded computing systems, Vol 8, No. 4, pp. 26-31,July 2009.

[5] Sharad Sinha and Thambipillai Srikanthan, "High level synthesis: Boosting designer productivity and reducing time to market", in IEEE Potentials, pp. 31-35, 2015.

[6] H.T.Kung, " Why Systolic Array?", in IEEE Trans. on Computer, Vol. 15,Issue 1, pp.37-46, 1982.

[7] Ville Eerola, JariNurmi,"High-level parameterizable area estimation modeling for ASIC designs", in Integration, the VLSI Journal, Vol 47, pp 461-475, 2014.

[8] Jae-Jin Lee,Gi-Yong Song, "Implementation of the Super-Sytolic Array for Convolution", in IEEE Design Automation Conference.

[9] Duo Liu, Yi Wang, Zili Shao,Minyi Guo,Jingling Xue,"Optimally Maximizing Iteration-Level Loop

Parallelism", in IEEE Transactions on parallel and distributed systems, Vol 23, No.3, pp 564-572, March 2012.

[10] Keshab K. Parhi,"Hierarchical Folding and Synthesis of Iterative Data Flow Graphs", in IEEE Transactions on circuits and systems-II:Express Briefs, Vol 60, No.9, p.p 597-601,September 2013.

[11] Philippe Coussy, Daniel D Gajski, Micheal Meredith, Andres Takach, "An introduction to high level synthesis",in IEEE Design and Test of computers vol 26, issue 4, p.p 8-17, Jul-Aug 2009.

[12] Erdal Oruklu, Richard Hanley, Semih Aslan, Cristophe Desmouliers, Fernando M Vallina, Jafer Saniie, "System-on-chip design using High level synthesis tools", in Journal on Circuits and systems, Scientific Research, Vol.3, p.p 1-9, 2012