Bit density-based pre-characterization of RAM cells for area critical SOC design

Dilip Kumar Ajay, QT Technologies Ireland Limited (*dajay@qti.qualcomm.com*)

Abstract*— Memories constitutes a significant percentage of almost all SOC's die area. For a given logical depth, logical width and cycle time requirement, we get many memory configurations with different values for access time, physical width, physical depth and area. Today the decision to pick the right RAM cell for a given design is iterative and manual. Existing techniques for this mostly involve designer feedback which is a slow, reactive and incremental approach. In multi-million gate design, prediction and mitigation of design issues as early as possible in the design cycle helps avoid many iterative fixes which subsequently have a huge impact on design convergence and design turnaround time. In this paper, Pre-characterization of RAMs (**random access memory**) for automatic area efficient design is modeled and we use pre-characterization scores during synthesis. Using this approach, one can be sure about the tool picking the most area efficient RAM cell for the design, while meeting other design constraints. This allows designers to resolve the area problems caused by RAM cells before going to the layout design phase. We haven't applied this flow to a large-scale high-level synthesis production design because it is mostly a concept at this stage.

Keywords: *High-Level Synthesis, Memory Optimization, System-on-Chip.*

I. Introduction

Memory cells constitutes a significant percentage of the SOC area. Today, for a given cycle time requirement, the memory cell which best fits the design is typically chosen manually by the designer. If the designer's choice goes wrong, this can lead to an increase in die area or not meeting the timing requirements. There is a comprehensive need to explore early predictions, estimations and mitigations early in the design to aid synthesis tools to do better.

There are difficulties related to design turnaround time (back and forth design churns between logic / synthesis / place and route each time) and design closure (balance timing / power / area specification simultaneously with congestion) in existing approaches, especially in large design block implementations.

Physical aware synthesis only includes the aspect ratio of the block for optimization. However, there is no known technique which includes the intelligence to automatically decide the best RAM cell for a given aspect ratio during synthesis for better area optimization for designs, while taking other design parameters like timing, congestion, power etc. into account.

In the worst-case scenario, design failure can be caused if incorrect RAM cells are mapped during synthesis, due to density overflow. In traditional RTL-based (Register Transfer Level) design flows, area problems caused by RAM cells cannot be found until the floorplan step. This problem is exacerbated during logic synthesis.

In this proposal we pre-characterize RAM cell library for area. We score each library cells based on area characteristics. This is a one-time processing effort before the synthesis flow. Multiple scores are possible per library cell for the range of expected values for different parameters available during memory compilation. The library cell score for area optimization is then used in the synthesis flow. The pre-characterization scores are used to estimate/optimize the area profile.

Other cost metrics used in synthesis like congestion, timing, power etc. will not be affected at all when applying the formulae developed in this paper because the whole concept is to only pre-characterize RAM cells based on the factors discussed. The aim of the technique is to decide the best RAM cell for a design from the library in terms of area, while taking frequency and access times into account. Once the decision is made, we can map it during the technology mapping step during logic synthesis.

Below flow-chart explains how the proposed idea can be integrated in the existing synthesis technique.

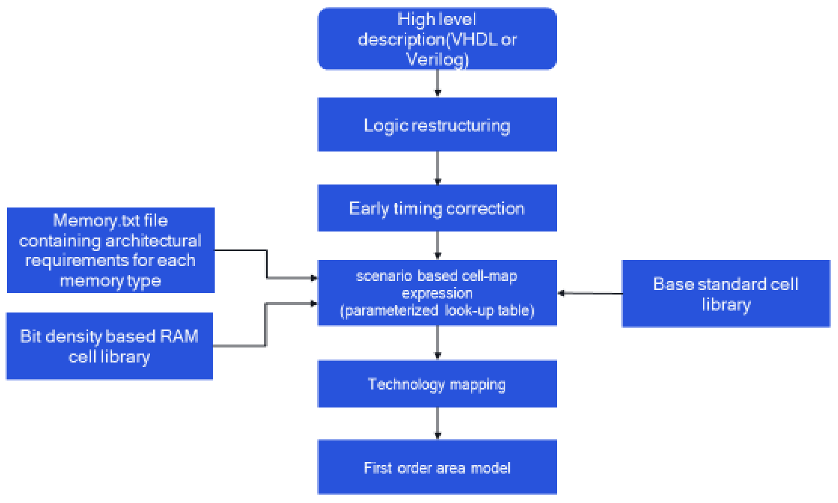

High level description(VHDL or Verilog)
Logic restructuring
Early timing correction
Memory.txt file containing architectural requirements for each memory type
Bit density based RAM cell library
scenario based cell-map expression (parameterized look-up table)
Base standard cell library
Technology mapping
First order area model

Fig.1

The proposed technique ensures minimum impact in the sign-off process while significantly improving design turnaround time (TAT).

II. RELATED WORK

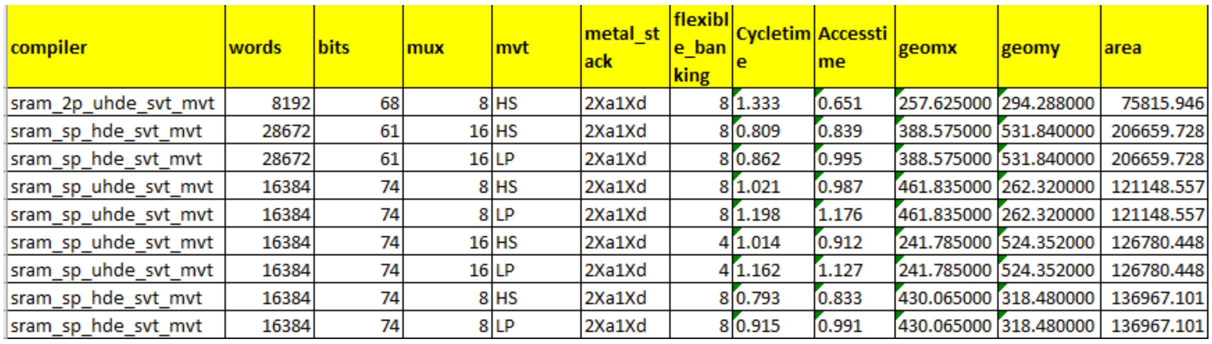
The specialization of the memory architecture has been widely studied in embedded systems due to its impact on area, power, and performance [1].
Panda et al. have proposed various solutions to create custom architectures and improve the system's performance, both in terms of memory organization and data layout [1].
Similarly, Benini et al. proposed a technique to customize the memory sub-system with respect to an application, using profiling, while accounting for layout information to minimize power consumption [2].
The possibility of sharing and reusing memory elements has been explored by Desnos et al. [3].
Wang et al. [6] propose an automated methodology for data reuse, memory partitioning, and memory merging for FPGAs. This approach has also been extended to consider the concurrent optimization of multiple processes [4], but only to optimize the fine-grained communication, not the storage elements.
On the other hand, Wang et al. presented a complete theory for data partitioning which includes the support for more complex data access patterns [5]. This method can be added to our approach as a technique to determine how the logical addresses need to be translated into the corresponding physical addresses. All these solutions are applied to the behavioral specification, before high-level synthesis.
This is efficient and elegant, but it imposes several limits to the reusability of the components as they need to be resynthesized each time the memory subsystem is modified.
Recently, different approaches have been proposed to effectively enable the reuse of pre-characterized components in SoCs. Liu et al. compose pre-characterized components to create a Pareto set of the entire system [7]. However, memory aspects are not considered.

Li et al. adopt a similar approach to create systems by composing pre-characterized IPs through a pre-defined architectural template [8]. The approach has been further extended to determine the organization of the memory subsystem [9]. However, it is not clear how the authors can deal with memories having different access times, especially considering the explicit synchronization of the entire system through a finite state machine. Moreover, it seems that parameters like the logical width and logical depth are not explored as this constraint is not considered when determining the memory elements.
Tatsuoka Masato et al. propose a novel design flow by integrating HLS tool with physically aware logic synthesis technology [10]. Here congestion prediction and mitigation techniques are incorporated during logic synthesis

however there is nothing about scoring or pre-characterization of memory cells and area aware mapping technique are not considered.

III. KEY WORDS AND FORMULAE

A typical memory configuration today looks like



compiler	words	bits	mux	mvt	metal_stack	flexible_banking	Cycletime	Accesstime	geomx	geomy	area
sram_2p_uhde_svt_mvt	8192	68	8	HS	2Xa1Xd	8	1.333	0.651	257.625000	294.288000	75815.946
sram_sp_hde_svt_mvt	28672	61	16	HS	2Xa1Xd	8	0.809	0.839	388.575000	531.840000	206659.728
sram_sp_hde_svt_mvt	28672	61	16	LP	2Xa1Xd	8	0.862	0.995	388.575000	531.840000	206659.728
sram_sp_uhde_svt_mvt	16384	74	8	HS	2Xa1Xd	8	1.021	0.987	461.835000	262.320000	121148.557
sram_sp_uhde_svt_mvt	16384	74	8	LP	2Xa1Xd	8	1.198	1.176	461.835000	262.320000	121148.557
sram_sp_uhde_svt_mvt	16384	74	16	HS	2Xa1Xd	4	1.014	0.912	241.785000	524.352000	126780.448
sram_sp_uhde_svt_mvt	16384	74	16	LP	2Xa1Xd	4	1.162	1.127	241.785000	524.352000	126780.448
sram_sp_hde_svt_mvt	16384	74	8	HS	2Xa1Xd	8	0.793	0.833	430.065000	318.480000	136967.101
sram_sp_hde_svt_mvt	16384	74	8	LP	2Xa1Xd	8	0.915	0.991	430.065000	318.480000	136967.101

Table.1

Area Difficulty Score (ADS): is a parameter or a weight given to each RAM cell of the library based on how much each RAM cell is likely to contribute towards the area. Area difficulty score of each RAM cell is a function of number of words, number of bits, physical width and depth.

$$\text{ADS of a RAM cell} = F\ (\text{words, bits, geom\_x, geom\_y}) \qquad (1)$$

ADS of a RAM cell is directly proportional to a new parameter we are introducing called "Megabits per mm square", which is nothing but bit density which will determine the area efficiency of each RAM cell.

Calculating Megabits per mm square (Mbits/mm^2)

$$\text{Area of RAM cell in mm}^2 = [(\text{geom\_x}) * (\text{geom\_y})]/1000000 \qquad (2)$$

$$\text{Megabits} = [(\text{words}) * (\text{bits})]/1048576 \qquad (3)$$

$$\text{Mbits/mm}^2 = [(\text{Eq.3}) / (\text{Eq.2})] \qquad (4)$$

$$\text{ADS } \alpha\ [1/A_t],\ \text{where } A_{t,} \text{ is the access time of RAM cell.} \qquad (5)$$

Cycle time C_t of the RAM cell is treated as constant.

Therefore, we can say area difficulty score of each RAM cell can be derived from

$$\text{ADS} = [(C_t * \text{Mbits/mm}^2) / A_t] \qquad (6)$$

Higher the Mbits/mm2 value, better it is for area optimization. Hence, we can now pre-characterize or score RAM cells based on Mbits/mm2 value.

IV. BIT DENSITY AWARE SYNTHESIS FLOW

To address area problem caused by mapping inappropriate RAM cell, we propose a bit-density aware synthesis design flow as shown in Fig. 1. There are 2 additional steps which are introduced to the traditional design flow: provide bit-density aware pre-characterized RAM cell library and provide a memory.txt file with logic requirement for the memory used in the module.

Example

Let’s take a case where the logical requirement for the memory used in the module is logic depth of 21504 and logic width of 266. In each write cycle we need to write 160B of data into 3 banks of memory. The aspect ratio of the block is x=5000 micron and y=1650 micron.

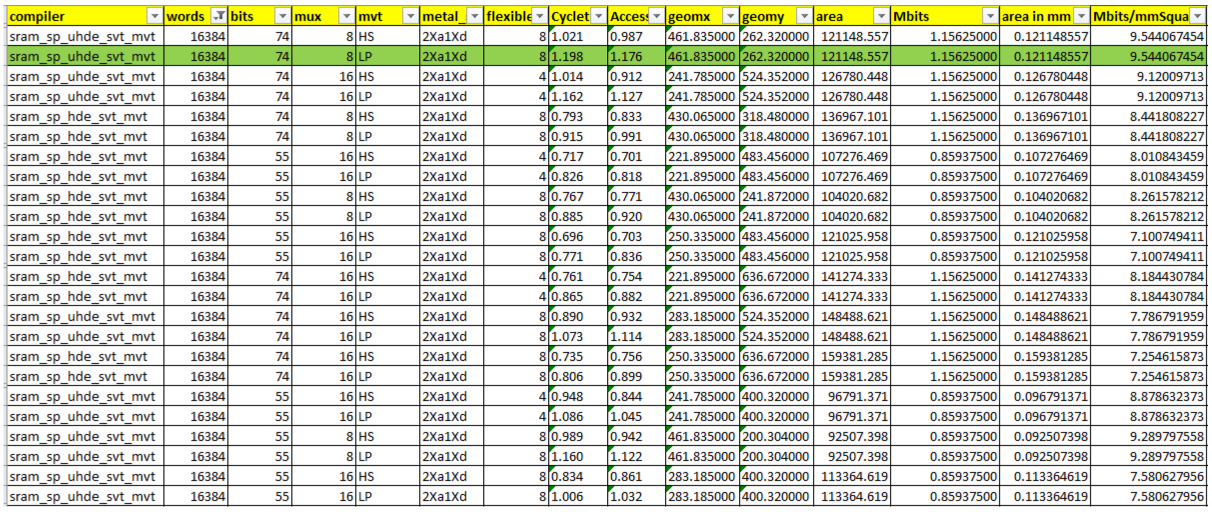
The memory.txt file content will be as below

```
#-------------------------------------------------------------------#
# Description:
#    List of memories used in this module
#
# Line format:
# vendor=VENDOR type=TYPE vt=VT depth=DEPTH width=WIDTH mux_factor=MF write_mask=WM
repaircols=RCOLS pipeline=PL \
# [banks=BANKS] wrapper_name=WNAME logical_width=LWIDTH width_inst=WINST
#

# Memory related options:
#
# VENDOR=[arm]
# TYPE=[sr1p_hd|sr2p_hd|sr2p_uhd|rf1p_hd|rf2p_hs]
# VT=[svt|hvt]       default: hvt
# MF= [2|4|8|16|32]
# WM=[yes|no] default: no
# PL=[yes|no] default: no
# BANKS= [1|2|4|8] - only applicable to sr2p_uhd memory
#
# Memory and wrapper related options:
#
# RCOLS= [0|1|2|3]
#
# Wrapper related options:
#
# WNAME=verilog module name
# LWIDTH=[integer]
# WINST=[integer]

############### PD XYZ mems ##############
vendor=arm type=sr1p_uhd vt=hs banks=8 mux_factor=8  write_mask=no repaircols=2 pipeline=ext
logical_width=266 logical_depth=21504  wrapper_name=XYZ
```

The pre-characterized RAM cell library for this given memory configuration specified in memory.txt file above will typically look like below table using Eq.4.



compiler	words	bits	mux	mvt	metal_	flexible	Cyclet	Access	geomx	geomy	area	Mbits	area in mm	Mbits/mmSqua
sram_sp_uhde_svt_mvt	16384	74	8	HS	2Xa1Xd	8	1.021	0.987	461.835000	262.320000	121148.557	1.15625000	0.121148557	9.544067454
sram_sp_uhde_svt_mvt	16384	74	8	LP	2Xa1Xd	8	1.198	1.176	461.835000	262.320000	121148.557	1.15625000	0.121148557	9.544067454
sram_sp_uhde_svt_mvt	16384	74	16	HS	2Xa1Xd	4	1.014	0.912	241.785000	524.352000	126780.448	1.15625000	0.126780448	9.12009713
sram_sp_uhde_svt_mvt	16384	74	16	LP	2Xa1Xd	4	1.162	1.127	241.785000	524.352000	126780.448	1.15625000	0.126780448	9.12009713
sram_sp_hde_svt_mvt	16384	74	8	HS	2Xa1Xd	8	0.793	0.833	430.065000	318.480000	136967.101	1.15625000	0.136967101	8.441808227
sram_sp_hde_svt_mvt	16384	74	8	LP	2Xa1Xd	8	0.915	0.991	430.065000	318.480000	136967.101	1.15625000	0.136967101	8.441808227
sram_sp_hde_svt_mvt	16384	55	16	HS	2Xa1Xd	4	0.717	0.701	221.895000	483.456000	107276.469	0.85937500	0.107276469	8.010843459
sram_sp_hde_svt_mvt	16384	55	16	LP	2Xa1Xd	4	0.826	0.818	221.895000	483.456000	107276.469	0.85937500	0.107276469	8.010843459
sram_sp_hde_svt_mvt	16384	55	8	HS	2Xa1Xd	8	0.767	0.771	430.065000	241.872000	104020.682	0.85937500	0.104020682	8.261578212
sram_sp_hde_svt_mvt	16384	55	8	LP	2Xa1Xd	8	0.885	0.920	430.065000	241.872000	104020.682	0.85937500	0.104020682	8.261578212
sram_sp_hde_svt_mvt	16384	55	16	HS	2Xa1Xd	8	0.696	0.703	250.335000	483.456000	121025.958	0.85937500	0.121025958	7.100749411
sram_sp_hde_svt_mvt	16384	55	16	LP	2Xa1Xd	8	0.771	0.836	250.335000	483.456000	121025.958	0.85937500	0.121025958	7.100749411
sram_sp_hde_svt_mvt	16384	74	16	HS	2Xa1Xd	4	0.761	0.754	221.895000	636.672000	141274.333	1.15625000	0.141274333	8.184430784
sram_sp_hde_svt_mvt	16384	74	16	LP	2Xa1Xd	4	0.865	0.882	221.895000	636.672000	141274.333	1.15625000	0.141274333	8.184430784
sram_sp_uhde_svt_mvt	16384	74	16	HS	2Xa1Xd	8	0.890	0.932	283.185000	524.352000	148488.621	1.15625000	0.148488621	7.786791959
sram_sp_uhde_svt_mvt	16384	74	16	LP	2Xa1Xd	8	1.073	1.114	283.185000	524.352000	148488.621	1.15625000	0.148488621	7.786791959
sram_sp_hde_svt_mvt	16384	74	16	HS	2Xa1Xd	8	0.735	0.756	250.335000	636.672000	159381.285	1.15625000	0.159381285	7.254615873
sram_sp_hde_svt_mvt	16384	74	16	LP	2Xa1Xd	8	0.806	0.899	250.335000	636.672000	159381.285	1.15625000	0.159381285	7.254615873
sram_sp_uhde_svt_mvt	16384	55	16	HS	2Xa1Xd	4	0.948	0.844	241.785000	400.320000	96791.371	0.85937500	0.096791371	8.878632373
sram_sp_uhde_svt_mvt	16384	55	16	LP	2Xa1Xd	4	1.086	1.045	241.785000	400.320000	96791.371	0.85937500	0.096791371	8.878632373
sram_sp_uhde_svt_mvt	16384	55	8	HS	2Xa1Xd	8	0.989	0.942	461.835000	200.304000	92507.398	0.85937500	0.092507398	9.289797558
sram_sp_uhde_svt_mvt	16384	55	8	LP	2Xa1Xd	8	1.160	1.122	461.835000	200.304000	92507.398	0.85937500	0.092507398	9.289797558
sram_sp_uhde_svt_mvt	16384	55	16	HS	2Xa1Xd	8	0.834	0.861	283.185000	400.320000	113364.619	0.85937500	0.113364619	7.580627956
sram_sp_uhde_svt_mvt	16384	55	16	LP	2Xa1Xd	8	1.006	1.032	283.185000	400.320000	113364.619	0.85937500	0.113364619	7.580627956

Table.2

We can see from the above table the number of options available for synthesis tool for the given memory specification. Since the synthesis tool is physically aware, it can read the size of the module, which is 5000x1650 microns.

It is evident from the above table, that the synthesis tool can easily pick the memory configuration with highest Mbits/mm^2, which would otherwise be user input based on floorplanning results and can be iterative.

Since we need to write 160B (1280 bits) of data in each cycle into 3 banks. From Table.2, we can see that we would need 4 instances of each RAM cell selected to realize logic depth of 21k and logic width of 266 (256 bits of data + 10 bits of ECC logic) plus 2 repair columns. Therefor we would need 5*4 instances to realize a logic width of 1280 bits or 160 Bytes. Since we need to write 160 Bytes of data in 3 banks, the total instances needed for this module to meet the aspect ratio and logical requirement would be 60 instances. A typical floorplan for this module will look like this as shown in Fig.2

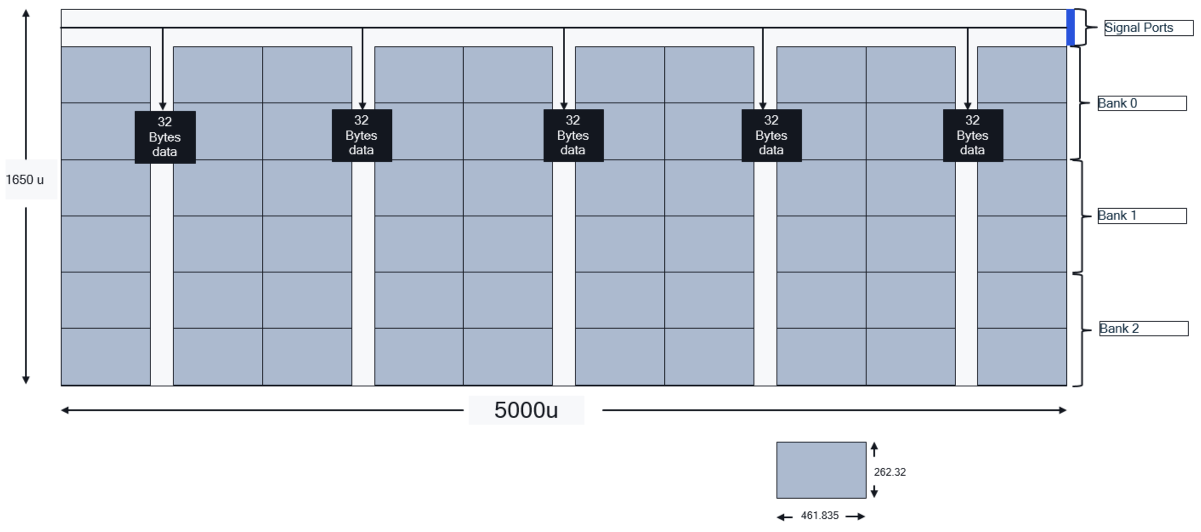


Fig.2

V. CONCLUSIONS

A new approach is proposed which can be easily be integrated into existing physical aware synthesis tool. The pre-characterized RAM cell based on bit-density aids synthesis tool during technology mapping step to intelligently select the best RAM cell for the given module for the aspect ratio and logical requirements, which would otherwise be iterative and error prone saving a lot of turnaround time.

VI. REFERENCES

[1]. P. R. Panda, N. D. Dutt, A. Nicolau, F. Catthoor, A. Vandecappelle, E. Brockmeyer, C. Kulkarni, and E. de Greef. Data memory organization and optimizations in application-specific systems. IEEE Design & Test of Computers, 18(3):56--68, 2001. Google ScholarDigital Library.

[2]. L. Benini, L. Macchiarulo, A. Macii, and M. Poncino. Layout-driven memory synthesis for embedded systems-on-chip. IEEE Trans. on Very Large-Scale Integration Systems, 10(2):96--105, Apr. 2002. Google ScholarDigital Library.

[3]. K. Desnos, M. Pelcat, J.-F. Nezan, and S. Aridhi. Pre- and post-scheduling memory allocation strategies on MPSoCs. In Proc. of the Electronic System Level Synthesis Conf. (ESLsyn), pages 1--6, June 2013.Google Scholar

[4]. W. Zuo, Y. Liang, P. Li, K. Rupnow, D. Chen, and J. Cong. Improving high-level synthesis optimization opportunity through polyhedral transformations. In Proc. of FPGA, pages 9--18, Jan. 2013.

[5]. Y. Wang, P. Li, and J. Cong. Theory and algorithm for generalized memory partitioning in high-level synthesis. In Proc. of FPGA, pages 199--208, Feb. 2014. Google ScholarDigital Library

[6]. Y. Wang, P. Zhang, X. Cheng, and J. Cong. An integrated and automated memory optimization flow for FPGA behavioral synthesis. In Proc. of ASPDAC, pages 257--262, Jan. 2012.Google Scholar

[7]. H.-Y. Liu, M. Petracca, and L. P. Carloni. Compositional system-level design exploration with planning of high-level synthesis. In Proc. of DATE, pages 641--646, Mar. 2012. Google ScholarDigital Library

[8]. S. Li, N. Farahini, A. Hemani, K. Rosvall, and I. Sander. System level synthesis of hardware for DSP applications using pre-characterized function implementations. In Proc. of the Int.l Conf. on Hardware/Software Codesign and System Synthesis, pages 1--10, Oct. 2013. Google ScholarDigital Library

[9]. S. Li and A. Hemani. Memory allocation and optimization in system-level architectural synthesis. In Proc. of ReCoSoC, pages 1--7, July 2013.Google ScholarCross Ref

[10]. tatsuoka.masato, watanabe.ryosuke, otsuka.tatsushi, hasegawa.takashiPhysically Aware High Level Synthesis Design FlowDAC '15, June 07 - 11, 2015, San Francisco, CA, USA