AREA-OPTIMAL CACHE COHERENT PROTOCOL FOR MANY-CORE NETWORK-ON-CHIP

NG WAI KIN

UNIVERSITI TEKNOLOGI MALAYSIA

AREA-OPTIMAL CACHE COHERENT PROTOCOL FOR MANY-CORE NETWORK-ON-CHIP

NG WAI KIN

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2022

DEDICATION

This thesis is dedicated to my father, who taught me that the best kind of knowledge to have been that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGEMENT

This project would not be possible without the aid and support from a lot of people. I would like to take this opportunity to express my gratitude to the effort and time from those who have aided me in making this work possible.

First and foremost, I would like to express my deepest sense of gratitude towards my supervisor, Professor Madya Ir. Dr. Muhammad Nadzir Bin Marsono for his advice and guidance throughout the progression of the project. I have solved numerous difficulties and problems which I have encountered during this work with his guidance. In addition, I would also like to thank him for the encouragement he had provided to me to improve on the project.

Lastly, I would like to express my gratitude to my family members for their continuous and unconditional love and support and express my appreciation to my fellow friends who have supported me throughout the project.

ABSTRACT

Cache coherence support is a major component in network-on-chip (NoC) systems which consist of multiple processing cores or elements as it is essential to ensure that the changes in shared memory are well communicated between all cores. Due to the nature and architecture of NoC, cache coherence protocols can have different characteristics in terms of various design consideration factors such as performance, area and power. Since the number of cores are expected to increase more in computing systems in the future, these factors need to be appropriately considered for scalability during design process so that the implementation will be feasible and be able to maintain an effectiveness of the system design. Cache coherence protocols proposed for NoC systems such as the directory protocol, Hammer and token protocol each has different impact on execution performance and design cost associated, due to the different mechanism used to maintain the cache coherency. In this project, these protocols are implemented and simulated using the GEM5 simulator and the area overhead is estimated using the Multicore Power, Area, and Timing (McPAT) framework. The simulation using blackscholes, fluidanimate and bodytrack application from the Princeton Application Repository for Shared-Memory Computers (PARSEC) benchmark shows that the Hammer protocol outperforms all evaluated protocols in execution performance, but the area overhead required for the protocol is also the largest. Token protocol, on the other hand, provide a significant lower performance, which is 2% lower compared to the Hammer protocol, but its 7% area overhead incurred is the lowest among all protocols. This shows that token protocol exhibits the best scalability for area overhead with increasing number of processing cores while providing moderate performance in terms of execution time.

ABSTRAK

Sokongan kesepaduan cache ialah komponen utama dalam sistem rangkaian-pada-chip (NoC) yang terdiri daripada berbilang teras pemprosesan atau elemen kerana ia adalah penting untuk memastikan bahawa perubahan dalam memori dikongsi dikomunikasi dengan baik antara semua teras. Disebabkan oleh sifat dan seni bina NoC, protokol kesepaduan cache mempunyai ciri yang berbeza dari segi pelbagai faktor pertimbangan reka bentuk seperti prestasi, kawasan dan kuasa. Memandangkan nombor teras dijangka akan meningkat dalam sistem komputer pada masa hadapan, faktor-faktor ini perlu dipertimbangkan dengan sewajarnya untuk kebolehskalaan semasa proses reka bentuk supaya implementasi boleh dilaksanakan dan dapat mengekalkan keberkesanan reka bentuk sistem. Protokol kesepaduan cache yang dicadangkan untuk sistem NoC seperti protokol direktori, Hammer dan token mempunyai kesan yang berbeza terhadap prestasi pelaksanaan dan kos reka bentuk, disebabkan oleh perbezaan mekanisme yang digunakan untuk mengekalkan kesepaduan cache. Dalam projek ini, protokol-protokol tersebut telah dilaksanakan dan disimulasikan menggunakan simulator GEM5 dan overhed kawasan telah dianggarkan menggunakan Multicore Power, Area, and Timing (McPAT). Simulasi menggunakan aplikasi blackscholes, fluidanimate dan bodytrack daripada Princeton Application Repository for Shared-Memory Computers (PARSEC) menunjukkan bahawa protokol Hammer mengatasi semua protokol dari segi prestasi perlaksanaan, tetapi overhead kawasan tambahan yang diperlukan juga adalah yang paling besar. Protokol token, sebaliknya, memberikan prestasi yang lebih rendah, iaitu 2% lebih rendah berbanding dengan protokol Hammer, tetapi overhed kawasannya sebanyak 7% adalah yang paling rendah antara semua protokol. Ini menunjukkan bahawa protokol token mempamerkan kebolehskalaan yang terbaik untuk overhed kawasan dengan peningkatan bilangan teras pemprosesan sambil memberikan prestasi sederhana dari segi masa pelaksanaan.

TABLE OF CONTENTS

	TITLE	PAGE
	DECLARATION	**iii**
	DEDICATION	**iv**
	ACKNOWLEDGEMENT	**v**
	ABSTRACT	**vi**
	ABSTRAK	**vii**
	TABLE OF CONTENTS	**viii**
	LIST OF TABLES	**x**
	LIST OF FIGURES	**xi**
	LIST OF ABBREVIATIONS	**xii**
	LIST OF APPENDICES	**xiii**
CHAPTER 1	**INTRODUCTION**	**1**
1.1	Problem Background	1
1.2	Problem Statement	3
1.3	Research Objectives	4
1.4	Scope of research	4
1.5	Report Organization	5
CHAPTER 2	**LITERATURE REVIEW**	**7**
2.1	Overview	7
2.2	Shared Memory on Many-core Systems	7
2.3	Cache coherence protocols	9
	2.3.1 MOESI Protocol	9
	2.3.2 Directory protocol	10
	2.3.3 Hammer protocol	11
	2.3.4 Token protocol	12
	2.3.5 Scalability of Cache Coherence Protocols	12
2.4	Related works	13

2.5	Summary	21
CHAPTER 3	**RESEARCH METHODOLOGY**	**22**
3.1	Overview	22
3.2	Cache Coherence Protocol	22
3.3	Overall Project Flow	22
3.4	Simulation Environment	24
	3.4.1 M5	24
	3.4.2 GEMS	24
	3.4.3 GEM5	25
3.5	Memory system and interconnect network	27
3.6	McPAT	29
3.7	Benchmark	31
3.8	Simulation Environment and Setup	33
3.9	Summary	34
CHAPTER 4	**RESULTS AND DISCUSSION**	**35**
4.1	Overview	35
4.2	GEM5 Simulator Setup	35
4.3	Validating System Setup	37
4.4	PARSEC Benchmark	38
4.5	Cache Access	40
4.6	Network Access	41
4.7	Area Estimation	42
4.8	Summary	43
CHAPTER 5	**CONCLUSION**	**44**
5.1	Achievement of Project Objectives	44
5.2	Future Work	45
REFERENCES		**46**
Appendices A - C		**50 - 63**

LIST OF TABLES

TABLE NO.	TITLE	PAGE
Table 2.1	Description of MOESI protocol's cache state.	10
Table 2.2	Summary of related works.	20
Table 3.1	Comparison of PARSEC, SPLASH-2 and SPEC CPU 2006.	31
Table 3.2	Comparison of PARSEC workloads used.	32
Table 3.3	Simulation parameters.	34
Table 4.1	Switches for the GEM5 simulation run command.	37
Table 4.2	Blackscholes.	38
Table 4.3	Fluidanimate.	38
Table 4.4	Bodytrack.	39
Table 4.5	Cache access statistics for Bodytrack application.	40
Table 4.6	NoC statistics.	41
Table 4.7	Area estimation with 65nm technology process.	43

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 1.1	Typical tiled chip multi-processor (CMP) structure [6].	2
Figure 2.1	KNL's architecture [14].	8
Figure 2.2	MOESI protocol's cache state transition [6].	9
Figure 2.3	Handling of cache miss in directory, Hammer and token protocols [6].	11
Figure 2.4	SelectDirectory's proposed structure [19].	14
Figure 2.5	Average memory delay (ns) against area:	15
Figure 2.6	Network latency for each protocol [24].	16
Figure 2.7	Area overhead for different protocols [6].	17
Figure 2.8	3D mesh NoC architecture [29].	18
Figure 2.9	The structure of DWP-D [32].	19
Figure 3.1	Overall project flow.	23
Figure 3.2	Mapping of system sketch to simulation output in GEM5. [33]	26
Figure 3.3	System call emulation mode (a) and full system simulation mode (b) of GEM5 simulator. [33]	27
Figure 3.4	High level view of Ruby memory simulator.	28
Figure 3.5	GARNET's interconnection modelling.	28
Figure 3.6	McPAT framework overview.	29
Figure 3.7	McPAT's estimation model.	30
Figure 3.8	McPAT's manycore system model.	30
Figure 3.9	Experiment setup.	33
Figure 3.10	Core interconnection in simulation.	33
Figure 4.1	Building GEM5 simulation model.	36
Figure 4.2	Running test application using the GEM5 simulator in SE mode.	37
Figure 4.3	Execution time overhead for blackscholes, fluidanimate and bodytrack application.	39

LIST OF ABBREVIATIONS

AMD	-	Advanced Micro Devices
CMP	-	Chip Multi-processor
DWP-D	-	Dynamic Way Partitioning Directory
FFT	-	Fast Fourier Transform
GEMS	-	General Execution-driven Multiprocessor Simulator
IRDS	-	International Roadmap for Devices and Systems
KNL	-	Knights Landing
McPAT	-	Multicore Power, Area, and Timing
NoC	-	Network-on-Chip
PARSEC	-	Princeton Application Repository for Shared-Memory Computers
PDE	-	Partial Differential Equation
ROI	-	Region-of-Interest
SLICC	-	Specification Language for Implementing Cache Coherence
SPH	-	Smoothed Paticke Hydrodynamics
XML	-	Extensible Markup Language

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	GEM5 Simulator Build Output Log	50
Appendix B	GEM5 Simulator Run Output Log	52
Appendix C	Linux Full System Output Log	56

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Most of the electronic devices today feature more than single processor core [1], as a methodology to better harness the thread-level parallelism in order to improve the computing performance. It is evident that the approach of adding more and more processing cores which can be individually turned on or off is apparently an appropriate architectural decision made in most modern architectures. It can maintain the balance between both the maximum peak performance when necessary and efficient power consumption during idle period.

The 2015 International Roadmap for Devices and Systems (IRDS) report predicts a 30-fold increase in the number of processing cores by 2030 due to the increasing demand for information processing [2]. With the number of processing cores increases and multicore computer architecture will be the norm in future, the architectural paradigm is trending towards communication intensive from computationally intensive.

Network-on-chip (NoC) possess a huge amount of application potential, as relatively, conventional bus-based systems can no longer effectively handle the communication between the large number of cores and leads to performance bottleneck [3]. At the time being, NoC architectures are widely researched [4] as an on-chip interconnect in improving inter-core communication to maximize performance of multi-core systems. Crossbar switches are not scalable when the number of processing elements increase to a certain extent. Since the shared interconnect becomes impractical as the cores increases [5], unordered point-to-point networks will be the mainstream interconnect technology [6].

With the introduction of NoC, associated challenges arise as the architecture per se when dealing with multi core system design considerations such as cache coherency [7]. Shared memory model provided to software programmers requires an efficient cache coherence support [6], to ensure all the changes made by processors in shared memory are communicated with all concerned processors in the system to maintain the overall order of instruction execution.

As in multi-core SoC designs, on-chip memory will be the mainstream paradigm used in NoC systems. Therefore, an efficient cache coherence protocol is a vital component in ensuring the optimum functioning and performance of a multi-core NoC system. Snooping based cache coherence protocol, which is common on conventional bus-based systems are not practical for NoC systems [6]. It is because its implementation does not scale well on systems with large number of cores, since it requires the usage of an impractically large interconnect bus while the snooping broadcast traffic incurred is also not feasible.

Therefore, cache coherence protocols such as directory [8], token [9] and Hammer [10] protocols, have been proposed as alternatives for unordered interconnect to address the shortcoming of the existing cache coherence methodologies. Since the number of processing elements are expected to exhibit an increasing trend, chip resources such as area utilization and power consumption is also expected to be increasing in the future computers.

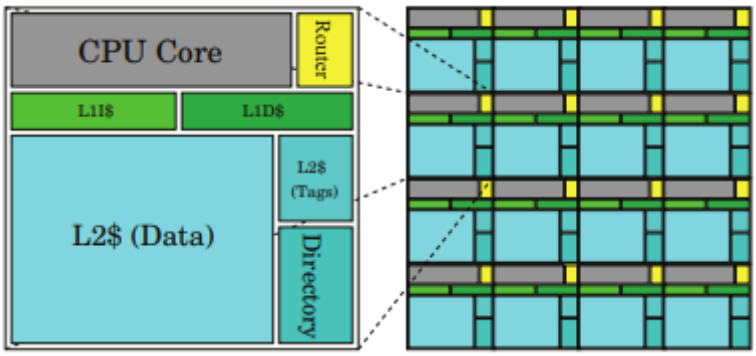

Figure 1.1 Typical tiled chip multi-processor (CMP) structure [6].

1.2 Problem Statement

With the current technology, it is possible that number of cores of multi core systems to be doubled every 18 months [11]. Scalability of various design parameters such as area and traffic will be a major challenge in chip design as the number of cores increases with the advancement in fabrication technology to boost both the computing performance and efficiency. When the number of cores in a system increases beyond some extend, not all protocols targeting implementation on unordered NoC will be well-adapted in terms of area and power overhead [4].

Chip area is among one of the most critical design constraints in today's chip design as cost of chip die is determined by chip area and chip power [12]. Furthermore, area-efficient designs will also result in a smaller size in the final end product, which is a desirable trait for designs which are targeting mobile platform market segment.

The directory protocol requires large chip area overhead as it utilises on-chip directory to store cache coherence information and the directory area grows with the number of processors. Hammer protocol on the other hand uses broadcast mechanism instead and requires less are overhead, giving slightly lower execution performance. Token protocol is based on token counting for cache coherence, hence also requires low area overhead, but can achieve comparatively better performance.

Therefore, it will be beneficial for multi core NoC systems to be optimally designed with the appropriate selection of the cache coherence protocol and architecture which exhibit good scalability for area overhead requirements.

1.3 Research Objectives

The primary goal of this proposal is to build a cache coherent multi core NoC system optimized for area scalability. Specifically, the objectives of this project are:

I. To characterize the impact of various cache coherence protocols in terms of the execution performance.
II. To validate and characterize using benchmark the cache coherence protocol with good scalability for area overhead.

1.4 Scope of research

When the number of processors to be simulated increases, the simulation time will also increase significantly. Therefore, the simulated architecture will be targeting a NoC system with 4 processing cores. The interconnect network topology is 2-dimensional (2-D) mesh network. Due to lack of hardware for evaluation, the performance evaluation is done through simulation approach, utilizing the GEM5 NoC simulator.

The evaluation of the cache coherence protocols is carried out using the Princeton Application Repository for Shared-Memory Computers (PARSEC) [13], limited to a subset of the benchmark suite, focusing on workload applications with moderate or high intensity of memory access. The chip area for proposed models is obtained based on the simulator is subsequently used for analysis and comparison.

1.5 Report Organization

This report is organized into five chapters. In Chapter 2, the following chapter, presents a literature review of related works on cache coherent NoC design and the analysis on different cache coherence protocol proposals. Next, Chapter 3 illustrates the proposed research methodology in this work, which includes the overall experiment flow and validation procedures. The corresponding project results and discussion will be contained in Chapter 4 of this thesis. Last but not least, the execution performance and area overhead for various cache coherence protocols will be concluded in Chapter 5.

REFERENCES

[1] M. Gschwind, "Chip multiprocessing and the cell broadband engine," *Proc. 3rd Conf. Comput. Front. 2006, CF '06*, vol. 2006, pp. 1–7, 2006, doi: 10.1145/1128022.1128023.

[2] "THE INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS 2.0: 2015 LINK TO ITRS 2.0, 2015 FULL EDITION DETAILS 2.0 INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS 2.0 2015 EDITION EXECUTIVE REPORT THE ITRS 2.0 IS DEVISED AND INTENDED FOR TECHNOLOGY ASSESSMENT ONLY AND IS WITHOUT REGARD TO ANY COMMERCIAL CONSIDERATIONS PERTAINING TO INDIVIDUAL PRODUCTS OR EQUIPMENT."

[3] A. Ben Ahmed, A. Ben Abdallah, and K. Kuroda, "Architecture and design of efficient 3D network-on-chip (3D NoC) for custom multicore SoC," *Proc. - 2010 Int. Conf. Broadband, Wirel. Comput. Commun. Appl. BWCCA 2010*, pp. 67–73, 2010, doi: 10.1109/BWCCA.2010.50.

[4] Y. Li, J. Han, S. Wang, J. Liu, and X. Zeng, "A NoC-based multi-core architecture for IEEE 802.11i CCMP," *undefined*, pp. 196–199, 2011, doi: 10.1109/ASICON.2011.6157155.

[5] M. Azimi, "Integration Challenges and Tradeoffs for Terascale Architectures," *Intel Technol. J.*, vol. 11, no. 03, Aug. 2007, doi: 10.1535/ITJ.1103.01.

[6] A. Ros, M. E., and J. M., "Cache Coherence Protocols for Many-Core CMPs," *Parallel Distrib. Comput.*, Jan. 2010, doi: 10.5772/9454.

[7] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. B. Carter, "Interconnect-Aware Coherence Protocols for Chip Multiprocessors," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 339–351, May 2006, doi: 10.1145/1150019.1136515.

[8] L. A. Barroso *et al.*, "Piranha," pp. 282–293, 2000, doi: 10.1145/339647.339696.

[9] M. M. K. Martin, M. D. Hill, and D. A. Wood, "Token coherence," p. 182, 2003, doi: 10.1145/859639.859640.

[10] A. Ahmed, P. Conway, B. Hughes, and F. Weber, “AMD Opteron™ Shared Memory MP Systems.”

[11] S. Jafar, P. Kumar, R. Rajnish, and M. Jafar, “Architectural scheme for future embedded systems involving large number of processing cores,” *Proc. 7th Int. Conf. Conflu. 2017 Cloud Comput. Data Sci. Eng.*, pp. 392–396, Jun. 2017, doi: 10.1109/CONFLUENCE.2017.7943181.

[12] J. Kim, “Low-cost router microarchitecture for on-chip networks,” *Proc. Annu. Int. Symp. Microarchitecture, MICRO*, pp. 255–266, 2009, doi: 10.1145/1669112.1669145.

[13] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC benchmark suite: Characterization and architectural implications,” *undefined*, pp. 72–81, 2008, doi: 10.1145/1454115.1454128.

[14] A. Sodani *et al.*, “Knights Landing: Second-Generation Intel Xeon Phi Product,” *IEEE Micro*, vol. 36, no. 2, pp. 34–46, Mar. 2016, doi: 10.1109/MM.2016.25.

[15] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, “Interconnect-power dissipation in a microprocessor,” p. 7, 2004, doi: 10.1145/966747.966750.

[16] F. Baskett, T. Jermoluk, and D. Solomon, “The 4D-MP graphics superworkstation: computing+graphics=40 MIPS+MFLOPS and 100000 lighted polygons per second,” pp. 468–471, Jan. 2003, doi: 10.1109/CMPCON.1988.4913.

[17] L. M. Censier and P. Feautrier, “A New Solution to Coherence Problems in Multicache Systems,” *IEEE Trans. Comput.*, vol. C–27, no. 12, pp. 1112–1118, 1978, doi: 10.1109/TC.1978.1675013.

[18] M. Shah *et al.*, “UltraSPARC T2: A highly-threaded, power-efficient, SPARC SOC,” *2007 IEEE Asian Solid-State Circuits Conf. A-SSCC*, pp. 22–25, 2007, doi: 10.1109/ASSCC.2007.4425786.

[19] Y. Yao, G. Wang, Z. Ge, T. Mitra, W. Chen, and N. Zhang, “SelectDirectory: A selective directory for cache coherence in many-core architectures,” *Proc. - Design, Autom. Test Eur. DATE*, vol. 2015-April, pp. 175–180, Apr. 2015, doi: 10.7873/DATE.2015.0438.

[20] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, “The SPLASH-2 programs: characterization and methodological considerations,” *undefined*, pp. 24–36, 1995, doi: 10.1145/223982.223990.

[21] Naveen Muralimanohar and Rajeev Balasubramonian. Cacti 6.0: A tool to model large caches.

[22] L. Yavits, A. Morad, and R. Ginosar, “Cache Hierarchy Optimization.”

[23] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, “Toward dark silicon in servers,” *IEEE Micro*, vol. 31, no. 4, pp. 6–15, Jul. 2011, doi: 10.1109/MM.2011.77.

[24] B. Aghaei and N. Zaman-Zadeh, “Evaluation of Cache Coherence Protocols in terms of Power and Latency in Multiprocessors,” 2016.

[25] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha, “GARNET: A detailed on-chip network model inside a full-system simulator,” *ISPASS 2009 - Int. Symp. Perform. Anal. Syst. Softw.*, pp. 33–42, 2009, doi: 10.1109/ISPASS.2009.4919636.

[26] P. S. Magnusson *et al.*, “Simics: A full system simulation platform,” *Computer (Long. Beach. Calif).*, vol. 35, no. 2, pp. 50–58, 2002, doi: 10.1109/2.982916.

[27] M. M. K. Martin *et al.*, “Multifacet’s general execution-driven multiprocessor simulator (GEMS) toolset,” *ACM SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 92–99, Nov. 2005, doi: 10.1145/1105734.1105747.

[28] V. Puente, J. A. Gregorio, and R. Beivide, “SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems,” pp. 15–22, Jun. 2003, doi: 10.1109/EMPDP.2002.994207.

[29] H. Suresh, “OPTIMIZATION OF COMMUNICATION TRAFFIC IN HAMMER PROTOCOL USING 3D ELECTRONIC MESH NETWORK ON CHIP.”

[30] C. H. Chao, K. Y. Jheng, H. Y. Wang, J. C. Wu, and A. Y. Wu, “Traffic- and Thermal-Aware Run-Time Thermal Management Scheme for 3D NoC Systems,” *undefined*, pp. 223–230, 2010, doi: 10.1109/NOCS.2010.32.

[31] M. Xie, D. Zhang, and Y. Li, “Meshim: A high-level performance simulation platform for three-dimensional network-on-chip,” *undefined*, pp. 349–352, 2011, doi: 10.1109/ASICON.2011.6157193.

[32] J. J. Valls, M. E. Gómez, A. Ros, and J. Sahuquillo, “A Directory Cache with Dynamic Private-Shared Partitioning,” *undefined*, pp. 382–391, Feb. 2016, doi: 10.1109/HIPC.2016.051.

[33] J. Lowe-Power *et al.*, “The gem5 Simulator: Version 20.0+,” Jul. 2020, doi: 10.48550/arxiv.2007.03152.

[34] Y. Kodama, T. Odajima, A. Asato, and M. Sato, “Evaluation of the RIKEN Post-K Processor Simulator,” Apr. 2019, doi: 10.48550/arxiv.1904.06451.

[35] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha, “GARNET: A detailed on-chip network model inside a full-system simulator,” *ISPASS 2009 - Int. Symp. Perform. Anal. Syst. Softw.*, pp. 33–42, 2009, doi: 10.1109/ISPASS.2009.4919636.

[36] J. Lacord, G. Ghibaudo, and F. Boeuf, “MASTAR VA: A predictive and flexible compact model for digital performances evaluation of CMOS technology with conventional CAD tools,” *Solid. State. Electron.*, vol. 91, pp. 137–146, Jan. 2014, doi: 10.1016/J.SSE.2013.10.007.

[37] J. Deutscher and I. Reid, “Articulated Body Motion Capture by Stochastic Search,” *Int. J. Comput. Vis. 2005 612*, vol. 61, no. 2, pp. 185–205, Feb. 2005, doi: 10.1023/B:VISI.0000043757.18370.9C.

[38] H.-S. Lim and M. Muller, “Particle-Based Fluid Simulation for Interactive Applications,” 2018.