Design and Implementation of an Adaptive Learning System: An MSc Project Experience

Daniel V. Martinez[†], Rishad A. Shafik[‡], Amit Acharyya[$], Geoff Merrett[†]
† School of ECS, University of Southampton, SO17 1BJ, UK, *e-mail: {ras1n09,gvm}@ecs.soton.ac.uk*
‡ School of EEE, Newcastle University, NE1 7RU, UK, *e-mail: {rishad.shafik}@ncl.ac.uk*
$ Indian Institute of Technology, Hyderabad, India, *e-mail: {amit_acharyya}@iith.ac.in*

***Abstract*—Individual project is a major component of MSc degrees in the UK in terms of allocated marks and overall weight. An MSc project typically lasts for up to 14-16 weeks, which makes it challenging to carry out elaborate research works. In this paper we highlight our experience in setting up one such project, which addresses the design and implementation of a bio-inspired adaptive learning system using cerebellar model articulation controller (CMAC) principles. To systematically execute the project, initially a guided and comprehensive literature survey was carried out to analyze in detail the CMAC design tradeoffs. Underpinning this analysis an adaptive design methodology was developed, leading to the implementation on a field programmble gate array (FPGA). The implementation was then validated using a real-time audio compression application. The validations demonstrated the advantages of adaptive learning systems in terms of reduced energy consumption with improved throughput and convergence. Moreover, it underlined the importance of a good plan and systematic delivery for a successful short-term research project.**

I. Introduction and Motivation

Microelectronics design projects are typically associated with two major challenges: firstly, finding an optimized design methodology for energy efficiency and secondly, dealing with the heterogeneity of the underlined platform for implementation. Carrying out a research-based design projects within a short time frame (typical for MSc projects) adds to these challenges the problem of finding a suitable problem and the applications that will be relevant to the research interests. This particular case has benefited from the overlap between the student's personal interest in automated learning systems and the supervising teams interest in research on such learning systems, such as cerebellar model articulation controller (CMAC) based adaptive learning system.

Developing automated learning-based system has recently gained significant attention from different sectors. Although various systems have proved to be reliable in practical applications, such as Real-time Control Systems (RCS) used in unmanned ground vechicles [7] and the system architecture for the Automated Manufacturing Research Facility (AMRF) [8]. These two being based on the CMAC which is a cerebellum-based neural network originally proposed in 1975. It is considered superior to traditional neural networks due to its fast learning abilities of non-linear or unpredictable systems [1]–[5]. It is typically used to model the behavior of unknown systems with reasonable accuracy. The learning is performed by a set of cumulative hyper-rectangular functions, each associated with a quantized weight variable. When learnt, the system behavior can be reproduced via these functions and weights. Besides generating the model, the CMAC-based learning systems also benefit from a high level of compression, which depends on the number of and weights.

CMAC learning systems can be configured as control systems in two major ways: with or without classical PID controllers. When configured without PID controllers, the system needs to be trained against a reference model. When configured with PID controllers, the system benefits from integral feedback from the target system, and CMAC trains the PID controller to model the target transfer function.

The following control applications stand out among the existing literature reviewed as being relevant to the current work: Learning CMAC control systems with a PID controller used to model the walking behavior of a bipedal robot [9] by means of a reinforced learning algorithm. A CMAC PD torque controller to deal with non-linearities such as friction and mechanical clearance [10].

Additional literature reviewed has dealt with mathematical and analytical aspects of the CMAC: Convergence has been proved for a certain range of learning rate values [21]. Over-learning, which is the system instability after succesful learning in systems controlling a plant, has been cancelled after analyzing the iterative learning model of a traditional CMAC [10]. Futhermore, hardware realizations of CMAC-based systems have also been reviewed, such as the color calibration module for image reproduction between scanner & printer using a CMAC without a PID controller [11], [12] and a photovoltaic non-linear system [1].

To successfully set up this challenging MSc project, three phases have been followed:

1) Comprehensive analysis of the CMAC & investigation into the tradeoffs of design parameters.
2) Development of a novel methodology design for the CMAC & validation through proper simulations.
3) Implementation of the system on an FPGA board & validation through a real-life application.

These three phases have implied the acquisition and further development of the following knowledge, abilities and skills:

- Programming skills for high level codes such as Matlab and C, to initially replicate the original CMAC model, simulate it and validate it against reported work.

- RTL modelling abilites to propagate the high level CMAC into a synthesizable hardware model; including extra processes in the testbench for data extraction for validation.
- Knowledge on FPGA boards through the manufacturer's documentation, manuals and tutorials to implement the hardware model and extract relevant data.
- Creativity in devising a series of applications after reviewing additional manufacturer's documentation on the peripherals available on the board.

Through the assessment and guidance provided on a weekly basis by the project's supervisors, the concurrent project flow outlining its milestones shown in Fig. 1 emerged.

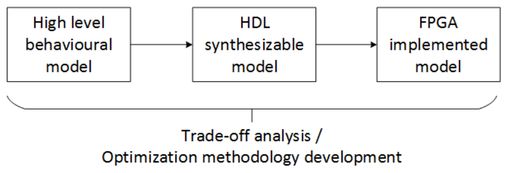

Fig. 1: Concurrent flow for the MSc project.

II. Phase I: Literature Review & Tradeoff Analysis

To carry out the project, a given set of previously published articles were reviewed in detail with an aim of understanding CMAC principles and the associated design tradeoffs, Fig. 2 shows a one-dimensional CMAC block diagram based on the originally proposed by Albus [19]. It is comprised by a *mapping* block containing the hyper-rectangular function, the *weight* memory block and arithmetic blocks to compute the sum for the output and the error times the *learning rate* to update the weight memory. The main parameters have been renamed from Albus' original work, to better fit McCann's [20] and Yang's [10] proposal in which the total length of the association vector is: $|\mathbf{A}| = N + C$:

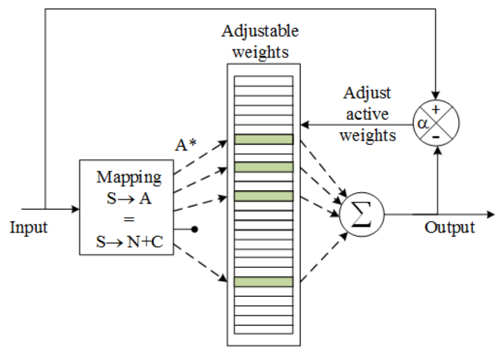

Fig. 2: CMAC block diagram based on Albus' original work.

Active association cells C: equivalent to Albus' A*
Quantization length N: equivalent to Albus' $|\mathbf{A}|$ - A*

The range R of the input S needs to be determined to set up the CMAC for N to quantize the input into a valid number of *states*. Depending on the immediate magnitude of S, a continuous group of C active association cells will be placed in the mapping vector. These active association cells will then map to certain weights in the weight vector. The minimum input means the active association cells remain on the bottom portion of the weight vector pictured in Fig. 2, or the LSB side of a binary vector if represented by one.

Such mapping process has been originally proposed by [19] and is shown in TABLE V for the case where $N = 3$ & $C = 2$.

TABLE I: Effect of $N = 3$ & $C = 2$ on an input S

		$\|\mathbf{A}\|$ = N + C				
S	S/N	A_0	A_1	A_2	A_3	A_4
0	0	1	1	0	0	0
1	0	1	1	0	0	0
2	0	1	1	0	0	0
3	1	0	1	1	0	0
4	1	0	1	1	0	0
5	1	0	1	1	0	0
6	2	0	0	1	1	0
7	2	0	0	1	1	0
8	2	0	0	1	1	0
9	3	0	0	0	1	1
10	3	0	0	0	1	1

N rules on the number of discrete states $(N + 1)$ regardless of the input range. The larger the input range is, the larger the discrete states or the size of the steps become.

The mapped weights in the weight vector are selected and added up to compute the output of the system. Learning is achieved in a *Gauss-Seidel* iterative scheme [21], learning is performed by evenly distributing the product of the *error* and the learning rate to the mapped weights in each iteration.

The main parameters typically produce tradeoffs between energy consumption, performance and convergence rate, which has been reported as follows:

1) Lane *et.al.* comment on how increasing the quantization length results on higher CMAC accuracy [15].
2) Lin and Chiang [21] determined that to guarantee stability and convergence for a CMAC with no plant, the learning rate should be kept: $0 \leq \alpha \leq 2$.
   For the case in which the CMAC has a plant, Yang *et.al.* developed another criteria [10].
3) Yang *et.al.* identified that a large amount of active association cells causes the CMAC to be less accurate [10].
4) Chew *et.al.* have clarified that their CMAC's parameters for the robot were adjusted by *trial-and-error* [9].

TABLE II shows there's a gap for completely adaptive CMAC systems for multiple applications. Some systems are pure simulations or HDL syntheses (rows 1, 3, 4, 7), while others have been implemented in FPGA and real systems (rows 2, 6) These limitations were studied and it was felt necessary to develop an *adaptive* and *application-tailored* CMAC system.

As shown in Fig. 1 the first milestone of the project was to build a high level behavioural model. During the first two weeks of the project plan, the high level behavioural model has been adapted from [20] in MATLAB by stripping it off of any PID functionality to emulate a

TABLE II: Summary of the existing CMAC systems

Approach	Application adaptive?	Implementation?	Features
[9]	No	Simulation	Q-learning
[10]	No	Real System	Solved instability in practical applications
[11]	No	Simulation	Variable set-up
[18]	No	Simulation	Reduced memory requirements
[15]	No	Theoretical	Differentiable
[1], [5]	No	Real System	Differentiable Improved learning & accuracy
[17]	No	Simulation	Differentiable CMAC
Proposed	Yes	FPGA	Adaptive for multiple applications

classical CMAC. The heuristics of the code are listed in TABLE III.

Two important figures in the CMAC: *convergence rate* and *accuracy* were measured for three types of periodic signals.

The measurement takes place for the ranges of $C = \{1, 2 \ldots 39, 40\}$ and $N = \{10, 20 \ldots 390, 400\}$ and a unitary learning rate $\alpha = 1$, the latter has been decided to prevent the necessity of a floating point operator hence simplifying future hardware implementation. To increase the relevance and accuracy of the measured data a phase sweep is done for each periodic signal from 0° to 360° and the individual values recorded averaged at the end.

1) Convergence Rate: A neural network converges when its memory contents remain unaltered for the same sample in two consecutive learning cycles [21]. The high level behavioural model compares the difference in the weight vectors between two consecutive cycles to determine convergence, it records the cycle for each phase shift to average it at the end.

2) Accuracy: By deasserting the learning function after convergence, the CMAC responds only with the information already learnt, this depends heavily on N & C. The recorded value is the absolute average of the individual errors for each sample in the whole signal cycle.

Heuristics in TABLE IV describe the loops added to perform such measurements. Two days were devoted to the analysis and interpretation of the results, the outcomes are in Fig. 3 for the convergence rate and Fig. 4 for accuracy.

From Fig. 1, the second milestone was to develop an HDL block for the CMAC in System Verilog, the standard in every MSc lecture. Three weeks were devoted to the creation and validation of the HDL model. CMAC was split into two separate blocks: the mapping block and the weight memory block. Parting from [20] in which the

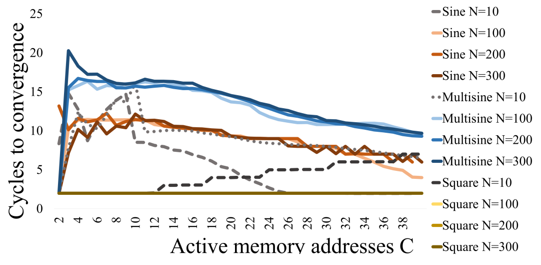

Fig. 3: Convergence rate and its relationship to N & C

TABLE III: Heuristics for the high level behavioural model

```
1  Set learning rate
2  Create and initialize mapping and weight vectors sized $N + C$
3  Determine signal maximum amplitude and resolution
4      FOR each time sample:
5          Compute the input of a periodical function
6          Compute the position of C in mapping vector
7          Compute output as the product of the weight
8          vector and the transpose of the mapping vector
9          Compute the error as the difference
10         between output and input
11           IF the CMAC is in learning mode:
12               Distribute error evenly among C active weights
13     Repeat for next iteration
```

verilog code made no distinction between each, the split was intended to reduce the gate count by reducing the amount and complexity of combinational FOR loops.

The mapping block has been designed as a purely combinational block which computes the range as the maximum possible values due to the selected word length. A combinational FOR is used to initialize the mapping vector, with a shift left operation << to place the active association cells C according to the quantization of the input. The output is the whole association vector, the only input is the input to the system S. The main portion of the SystemVerilog code can be seen in TABLE V.

The weight memory block is an active low reset, active high enable, synchronous block which computes the output through a combinational FOR loop, adding

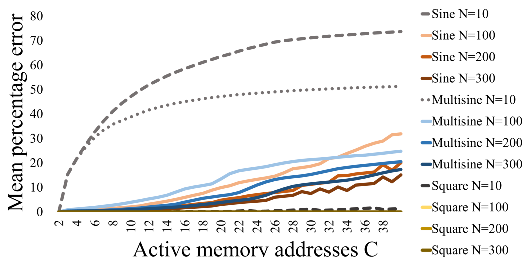

Fig. 4: Accuracy and its relationship to N & C

TABLE IV: Heuristics for computing convergence & accuracy

```
1  Set learning rate
2  Create and initialize mapping and weight vectors sized $N + C$
3  Determine signal maximum amplitude and resolution
4  Initialize $N$
5  FOR every value of $N$
6      Initialize $C$
7      FOR every value of $C$
8          Initialize phase shift $ph$
9          FOR every value of $ph$
10             Activate learning mode
11             FOR each time sample:
12                 Compute the input of periodical function
13                 Compute the position of C in mapping vector
14                 Compute output as the product of the weight
15                 vector and the transpose of the mapping vector
16                 Compute the error as the difference
17                 between output and input
18                 IF the CMAC is in learning mode:
19                     Distribute error evenly among C active weights
20             Repeat for next iteration
21         Increase $ph$ value by 1 until 360
22         IF previous weight vector equal to current
23             Record cycle
24             Exit learning mode
25         IF learning mode off
26             Record absolute average error
27     Increase $C$ value by 1 until 40
28 Increase $N$ value by 10 until 400
```

TABLE V: Combinatorial mapping

```
always_comb
  begin
    Smax = (1<<(ADC-1))-1;
    //Range is determined by ADC & bit count
    Smin = -(1<<(ADC-1));
    //ADC's work with 2's complement
    //a negative semicycle is taken into account
    range = Smax+((~Smin)+1);
    //Total range calculation
     minstep = range / (N-1);
     //Minimum shift step depends on N

     for(int i=0;i<((N+C)-1);i++)
     //FOR to assign active association cells
       if(i<C)
         A[i] = 1;
       else
         A[i] = 0;

     A = A << ((S-Smin)/minstep);
     //Final shift to compute output vector
  end
```

up each mapped weight. Whenever the learn signal is asserted it synchronously updates the weight memory values with another combinational FOR loop with the error adjust. The inputs are S and the association vector, the outputs are the computed total, and the error. The main part of the HDL code can be seen in TABLE VI.

To validate the HDL model it was necessary to add the following output commands: `$fopen` selects a file to read from or write to. `$fscanf` reads from a file, a CSV file in this case, storing the read data in a variable. `$fwrite` works similarly to `$monitor` but saving the information to a CSV file instead of displaying it on screen.

Given the availability of Synopsys Design Compiler at the University of Southampton, this tool has been chosen to perform the synthesis of the block, mapping it onto the 350nm library also available. Another three weeks were devoted, this time to research the available commands in Synopsys DC, discovering that it features the possibility of command line working and automated scripting, utilized to perform multiple batch syntheses and report power consumption and area usage. The main part of the script used to generate these syntheses is presented in TABLE VII.

Regarding the range of N values in TABLE VII, the quantization in the hardware application is accomplished by an analogue-to-digital converter (ADC), which is limited to work on powers of two. Four days were allocated to analyze the information reported by Design Compiler, which has been interpreted and presented in Fig. 5 and Fig. 6.

TABLE VI: Weight block, reset has been omitted

```
always_ff @ (posedge clock, negedge n_reset)
  for(int i=0;i<(N+C);i++)
  //Loop to update the weight vector
  if(enable & learn & A[i])
    w[i] <= w[i] + d_w;
    y_out <= y_out_1;
    //Computed result and error output
    error <= error_1;
  end

always_comb begin    y_out_1='0;
   error_1='0;
   for(int i=0;i<(N+C);i++)
   //Loop to calculate the output
     if(enable & A[i])
       y_out_1=y_out_1+w[i];

     error_1=S+~y_out_1+1;
     //Error and weight update calculation
     d_w=error_1/C;
end
```

As per Fig. 1 concurrently to the modelling, a tradeoff analysis has been devised by analyzing the graphs presented. The characteristic we are interested the most is a fixed trend in the main figures, meaning optimization can be achieved. The three periodic signals are taken into account in this analysis. The most relevant conclusions are outlined below:

3) Convergence rate: From Fig. 3 convergence in sine and multisine signals show no clear trend for values of C less than 10, therefore C has to be larger than 10. Also, whenever C is larger than N, convergence in square signals show a rising trend, therefeore C must be lower than N.

4) Accuracy: From Fig. 4 accuracy shows to be directly proportional to N and inversely proportional to C. However, N values below 100 show a significant degradation.

5) Power: From Fig. 5 power shows to be directly and linearly proportional to N as powers of two. However it is inversely proportional to C as powers of two: the larger the power of two C is the less power CMAC consumes.

6) Area: From Fig. 6, area shows no relevant influence from C while showing to be linearly proportional to N as powers of two.

III. Phase II: Design Optimization Methodology

Following the literature survey, a systematic optimization flow was devised. The main objective has been to generate a learning system with minimized energy consumption, while meeting the application-specific performance and accuracy requirements. Extensive analyses have been made to develop the optimization methodology outlined in Fig. 7. The four tradeoffs analyzed, their relationship to N & C and the suggestions to achieve an optimizable trend are shown.

TABLE VII: Script for parametric synthesis

```
for {set N 256} {$N < 1025} {set N [expr {$N * 2}] } {
for {set C 1} {$C < 40} {set C [expr {$C + 1}] } {
#Analyzing the SystemVerilog format of
every module of the CMAC
analyze -format sverilog $cmac_source
analyze -format sverilog {./cmac8_Ferr.sv
./w_mem_Ferr.sv ./map_clk_F.sv}

#Elaborating the top level
elaborate $cmac_top -architecture verilog
-library DEFAULT

#Creating a clock
create_clock clock -name clock -period 10

#This command allows for division
set synthetic_library {dw_foundation.sldb}

#FOR loops to a value less than the clock
compile

#Exporting each of the reports
report_power > power_${N}_${C}.rpt
report_area > area_${N}_${C}.rpt
remove_design -all
} }
```

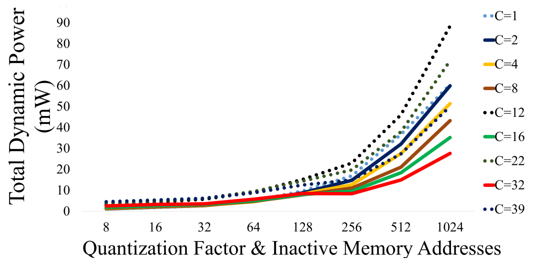

Fig. 5: Power Consumption for a 350nm CMAC vs N & C

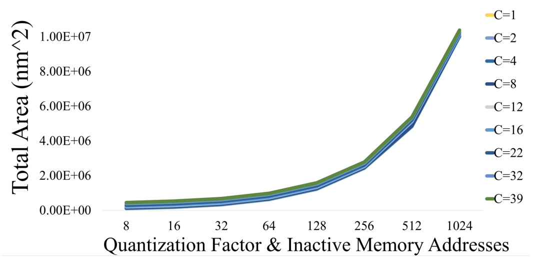

Fig. 6: Area required for a 350nm CMAC vs N & C

Depending on the particular application for the CMAC and its specific requirements, the steps below are suggested as guidelines for a successful implementation:

1) Understand the application: The most important thing is for the student to understand the application and its priorities, whether it is a wearable technology in need of optimized power or a critical control loop that requires high accuracy or even a fast learning system for limited training data.

2) Develop a high level behavioral model: A high level behavioural model allows the student to take advantage of the data extraction and display utilities for analyzing accuracy and convergence rate. Following Fig. 7 accuracy is improved by increasing N and decreasing C. Convergence rate is improved by decreasing N and increasing C.

3) Develop an HDL model: An HDL model can be created to extract approximate of power consumption and area usage.

According to Fig. 7 if power needs to be reduced N can be stepped down a power of two. As for C it can be stepped up a power of two. For area, a step down of N to the power of two below, directly reduces the total area usage.

4) Implement into FPGA: The implementation into an FPGA depends mostly on the size of the CMAC and the timing requirements, these data can be properly estimated from synthesis in the previous step.

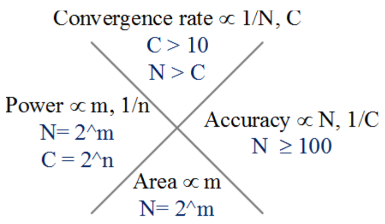

Fig. 7: Relationship of main figures with main parameters

IV. Phase III: Implementation & Validation

The design optimization methodology in Phase II was implemented on an Altera Ciclone IV FPGA available in the DE2 evaluation board from Terasic. After the SystemVerilog HDL files were successfully synthesized by Quartus II, the student had to create the appropriate file system to emulate SystemVerilog's output commands for CSV files. This was done by thoroughly revieweing the available documentation from Altera. The whole system was built up using the graphical system creation tool Qsys. A softcore processor, the Nios II was instanced to manage communication with the PC via JTAG-UART. The final C code was implemented in Eclipse, using the Nios II SBT. [22]–[24]

Finally, the whole system was programmed into the FPGA and data was written in and read out. Such data, presented in Fig. 8 validates the FPGA implemented CMAC response in black, implicitly validating the HDL model as well, by comparing it to the high level model response in magenta against a determined dashed input. This is done for $N = 64$ and $C = 32$, expecting low accuracy but fast learning or convergence rate.

A. CMAC as an Audio Compressor

One of the applications devised by the student was that of an audio compressor, thought of to exploit accuracy tradeoffs and to observe the effects in real signals. Due to time constraints, the development of such application was aided by the Altera University Program. The Laboratory Practice 12: *Basic Digital Signal Processing* [25] deals with real time audio input, processing and output by using the included drivers to configure and interface the Wolfson WM8731 audio CODEC directly from Quartus II [26].

CMAC can be understood as a compressor in the sense that it takes an analogue input and produces two digital outputs.

1) A *weight vector* sized [$(N+C)$,1] which at convergence remains constant.
2) An *association vector* sized [$(N+C)$,1] which varies in time in accordance to the instantaneous input.

This raises the question of how well could the sound be reinterpreted once it has been compressed. To answer this and further extend on the virtues of the CMAC, it was trained with a sound wave to regenerate it as per the Heuristics in Fig. VIII. The experiment has been

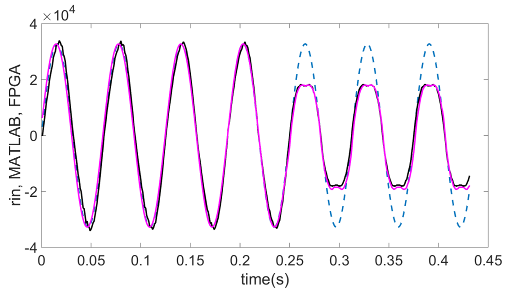

Fig. 8: Validation of FPGA CMAC against high level CMAC

TABLE VIII: Heuristics for the compression application

1 A WAV file is recorded with standard PC utilities
2 It is read in with the `wavread` available in MATLAB 2012b
3 The WAV file is processed into a single column vector
4 The WAV vector is fed into a CMAC until convergence is attained
5 After convergence, learning behaviour is cancelled
6 The WAV vector is fend into the CMAC again
7 The error is accounted as *noise* in MATLAB's `snr` function
8 `snr` function returns the signal to noise ratio in dB

performed exhaustively for the hardware friendly range of interest of N & C. From Fig. 9 the SNR of a CMAC regenerated audio signal is directly proportional to N and inversely proportional to C.

For the given ranges, the highest signal-to-noise ratio happens at $N = 1024$ & $C = 1$ with a magnitude of **40.8dB**. The lowest signal-to-noise ratio happens at $N = 8$ & $C = 39$ with a magnitude of **0.7647dB**. Fig. 9 shows that the signal to noise ratio is proportional to N in a *square root* scheme and inversely proportional to C.

This validates the assumption that quality, is directly proportional to N and inversely to C. This application was further validated by the student by recording his own voice through the Audio Codec and a microphone into the FPGA implemented CMAC. Depending on the choice for N and C the sound quality was perceived ranging from grainy (low N, high C) to clear (high N, low C), as expected.

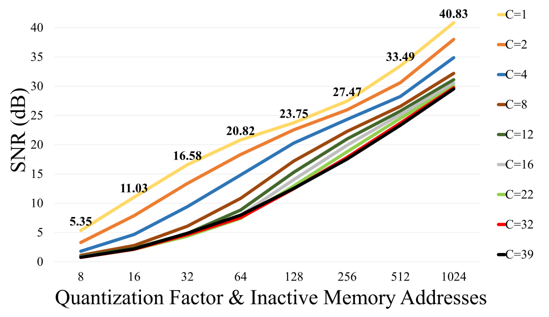

Fig. 9: N & fixed C vs SNR in dB for an audio signal

V. Experiences and Conclusions

Carrying out projects with novel insights in limited times is well known to be challenging. This, together with the broad range of papers, technical manuals and other materials that had to be reviewed from scratch, have represented a personal milestone in the student's personal career. Despite the stated challenges, it has been seen that such projects can be guaranteed to succeed if the following can be ensured:

a. Students' aptitude to carry out challenging project(s).
b. Good time and contingency plans.
c. Supervisors' awareness of the various design and implementation challenges.
d. Systematic multi-tasking in the project work.

The paper has presented the major outcomes from the work, highlighting the design tradeoffs studied initially, developing a systematic optimization methodology, and prototyping it on an FPGA platform. The outcomes demonstrate that the envisioned goals were achieved with the required adaptability and energy efficiency of CMAC based learning systems.

References

[1] C.-M. Chung *et al.*, "Hardware Implementation of CMAC Neural Network using FPGA Approach," *2007 International Conference on Machine Learning and Cybernetics*, vol. 4, no. August, pp. 19–22, 2007.

[2] F. Lewis *et al.*, "Multilayer neural-net robot controller with guaranteed tracking performance," *Neural Networks, IEEE Transactions on*, vol. 7, no. 2, pp. 388–399, March 1996.

[3] C. Gan and K. Danai, "Model-based recurrent neural network for modeling nonlinear dynamic systems," in *Control Applications, 1999. Proceedings of the 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1749–1754 vol. 2.

[4] C.-M. Lin and C.-F. Hsu, "Neural-network hybrid control for antilock braking systems," *Neural Networks, IEEE Transactions on*, vol. 14, no. 2, pp. 351–359, Mar 2003.

[5] C. Ching-Tsan and L. Chun-Shin, "{CMAC} with general basis functions," *Neural Networks*, vol. 9, no. 7, pp. 1199 – 1211, 1996.

[6] D.Lide, "A century of excellence in measurements, standards, and technology." *Measurement Science and Technology*, vol. 13, no. 10, p. 1653, 2002.

[7] J. S. Albus, "The NIST Real-time Control System (RCS): An Application Survey", *Proceedings of the AAAI 1995 Spring Symposium Series*, Stanford University, pp.27-29, 1995.

[8] J. Simpson *et al.*, *Automated manufacturing research facility of the national bureau of standards.* Natl. Engineering Lab, Washington,, DC, USA: SME, 1984.

[9] C.-M. Chew and G. a. Pratt, "Dynamic bipedal walking assisted by learning," *Robotica*, vol. 20, no. 05, 2002.

[10] B. Yang, H. Han, and R. Bao, "An intelligent cmac-pd torque controller with anti-over-learning schemke for electric load simulator." *Transactions of the Institute of Measurement and Control*, vol. 38, no. 2, pp. 192 – 200, 2016.

[11] R.-C. Wen *et al.*, "A cmac neural network chip for color correction," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 3, Jun 1994, pp. 1943–1948 vol.3.

[12] J.-S. Kerl *et al.*, "Hardware realization of higher-order CMAC model for color calibration," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1656–1661, 1995.

[13] Bowsher, David, "Introduction to the anatomy and physiology of the nervous system.," 1988.

[14] Albus, James S., "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, pp. 25–61, 1971.

[15] S. H. Lane *et al.*, "Theory and development of higher-order cmac neural networks," *Control Systems, IEEE*, vol. 12, no. 2, pp. 23–30, April 1992.

[16] C.-t. Chiang and C.-m. Chong, "Hardware Implementation of A Simple Structure of Addressing Technique for CMAC _ GBF," *Electrical Engineering*, pp. 139–144, 2005.

[17] M.-F. Yeh and C.-H. Tsai, "Standalone CMAC control system with online learning ability." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 40, no. 1, pp. 43–53, 2010.

[18] J.-s. Ker *et al.*, "Brief Papers," vol. 8, no. 6, pp. 1545–1556, 1997.

[19] J. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control*, pp. 220–227, 1975.

[20] D. Mccann, "Cerebellar Model Articulation Controller," Dissertation, University of Bristol, 2013.

[21] C. S. Lin and C. T. Chiang, "Learning convergence of CMAC technique." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 8, no. 6, pp. 1281–1292, 1997.

[22] A. Corporation, "Introduction to the Altera Qsys System," pp. 1–33, 2015.

[23] QSYS, "Making qsys components," pp. 1–36, 2015.

[24] Altera Inc., "Introduction to the Altera Nios II Soft Processor," pp. 1–27, 2011.

[25] Altera Inc., "Laboratory Exercise 12 - Basic Digital Signal Processing," pp. 1–4, 2015.

[26] Wolfson Electronics, "WM8731 Specifications," 2004.