Designing Efficient Computing Systems: The Approximate Computing Breakthrough

XXXIV Cycle – III year presentation

Salvatore Barone
salvatore.barone@unina.it

Tutor: Antonino Mazzeo
mazzeo@unina.it

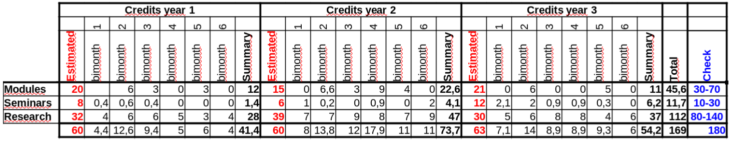
November 5, 2021

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Background I

- Master Degree in Information Technology at University of Naples – Federico II
- Thesis: Un prototipo di middleware configurabile – nel dominio ferroviario – per fault detection e comunicazione affidabile
  - This thesis work is part of a joint project involving the DIETI and Rete Ferroviaria Italiana – RFI – Gruppo delle ferrovie dello Stato S.p.a
- I'm currently involved in three research topics:
  - Approximate computing
  - Design of signaling systems for the railway domain
  - Hardware security

Background II



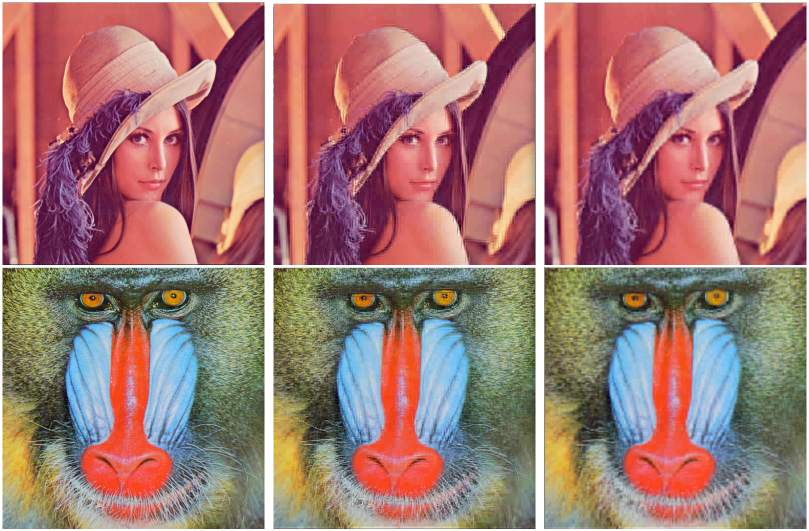
	Credits year 1								Credits year 2								Credits year 3									
		1	2	3	4	5	6			1	2	3	4	5	6			1	2	3	4	5	6			
	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Estimated	bimonth	bimonth	bimonth	bimonth	bimonth	bimonth	Summary	Total	Check
Modules	20		6	3	0	3	0	**12**	15	0	6,6	3	9	4	0	**22,6**	21	0	6	0	0	5	0	**11**	**45,6**	**30-70**
Seminars	8	0,4	0,6	0,4	0	0	0	**1,4**	6	1	0,2	0	0,9	0	2	**4,1**	12	2,1	2	0,9	0,9	0,3	0	**6,2**	**11,7**	**10-30**
Research	32	4	6	6	5	3	4	**28**	39	7	7	9	8	7	9	**47**	30	5	6	8	8	4	6	**37**	**112**	**80-140**
	60	4,4	12,6	9,4	5	6	4	**41,4**	60	8	13,8	12	17,9	11	11	**73,7**	63	7,1	14	8,9	8,9	9,3	6	**54,2**	**169**	**180**

Credit summary

Introduction I

- We are rapidly approaching the physical limit of the manufacturing process for integrated circuits (ICs).
  - We cannot rely on die-shrinking to improve performances anymore.
- Shrinking the size also increase manufacturing costs.
- Modern computing systems are experimenting an unprecedented growth of data to be processed.
- The increasing costs of energy severely impacts operating costs of computing systems.
  - It is estimated energy consumption will exceed the amount of energy produced before 2040 [3]

What can we do?



Outline

The Approximate-Computing design paradigm (AxC)

Introduction

- Inherent error-resilience:
  - the property of an application to produce acceptable outputs despite some of its underlying computations being incorrect or approximate.
    - significant redundancy is present in large, real-world, data sets that applications process,
    - computation patterns (such as statistical aggregation and iterative refinement) that intrinsically attenuate or correct errors due to approximations,
    - outputs are equivalent (i.e., no unique golden output exists), or
    - small deviations in the output cannot be perceived by users.

Challenges I

- The wide-spread use of AxC is hindered by several challenges:
  - Identifying approximable data/portion, and suitable approximation techniques
  - Error metrics and error-assessment
  - Savings estimation
  - Optimization
  - Lack of general methodology

Approximable data/portion and techniques I

- Applying AxC requires the designer to have deep knowledge of the target application.
  - Which data should I approximate?
  - Which portion should I approximate?
  - How should I introduce approximation?
- A naive approach is unlikely to be effective.
- A vast plethora of approximation techniques have been proposed, either for SW or HW applications
  - SW: loop-perforation, memoization, load-value approximation, precision-scaling, etc.
  - HW: timing: voltage/frequency scaling; functional: Boolean algebra, AIG-rewriting, precision-scalig, Cartesian Genetic Programming (CGP), etc.
- How to avoid manually-introducing approximation?

Approximable data/portion and techniques II

- Mutators! [2]
  - They leverage the Abstract Syntax Tree (AST) representation.
  - $\{match, mutate\}$:
    - *match*: identify which part of the algorithm has to be approximate
    - *mutate*: defines how to apply approximation
  - The definition is application-independent.
  - Require almost no knowledge on the target application.
  - Mutators hide the adopted approximation technique.
  - Applying multiple mutators to the same AST allows simultaneously applying multiple approximation techniques!

Approximable data/portion and techniques

- Applying mutators define an approximate variant of the original algorithm.
- Approximate variants allow configuring the degree of introduced approximation.
  - n approximable operations, each allowing k different degrees of approximation
  - simultaneously approximating j operations results in $\binom{n}{j}$ different approximate variants
  - each variant allows k^j different approximate configurations
  - the total number of approximate configurations is $\sum_{i=1}^{n} k^i \times \binom{n}{i}$.
  - **The design space may be awkwardly large!**

Design-space exploration I

- The goal is to save as much as possible while introducing as little error as possible.
- Unfortunately, little error and significant savings are conflicting design objectives!
- AxC requires addressing Multi-objective Optimization Problems (MOPs)
  - There is no single solution!
  - Solving MOPs may be computation-intensive.
  - A suitable heuristic must be selected, e.g., NSGA-II [4] or AMOSA [6].
  - Decision-variables and fitness-functions must be properly identified.

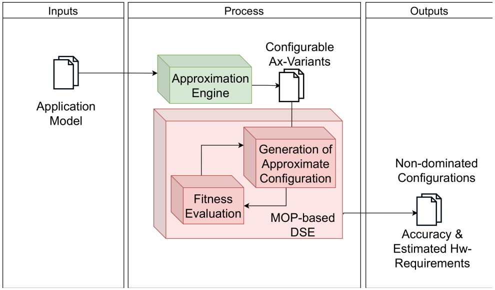
Error-assessment

- Error-assessment is mandatory to guarantee the output-quality satisfies application-defined constraints.
  - A suitable error metric, or a set of metrics, have to be selected.
    - Some metrics may be too sensitive, hampering the AxC.
    - The PSNR is too sensitive to noise, the error-frequency is not suitable for arithmetic operators
  - A suitable error-assessment process have to be selected:
    - It strictly depends on the application.
    - Simulating the whole application may be utterly time-consuming.
    - Formal-methods may be inapplicable, due to complexity.
    - The lack of input-output pairs may hamper machine-learning-based techniques.

Estimating hardware-requirements

- Assessing requirements of an application may require the simulation over a significant dataset
  - The dataset may be not available.
  - The whole procedure may be too time-consuming.
  - Concerning hardware, assessing requirements may require the synthesis and simulation of large, complex designs.
- Model-based estimation
  - The model is application-dependent
  - Must consider how the approximation technique impacts on requirements

Summary

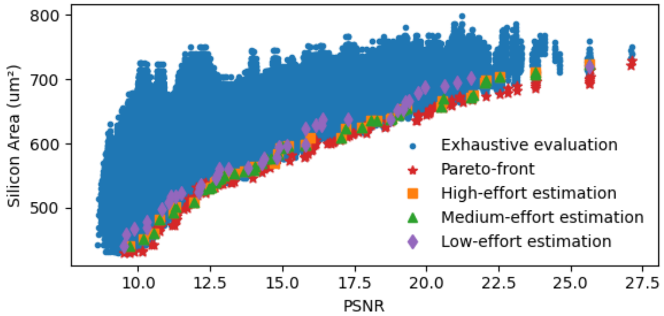

Inputs
Process
Outputs
Application Model
Approximation Engine
Configurable Ax-Variants
Generation of Approximate Configuration
Fitness Evaluation
MOP-based DSE
Non-dominated Configurations
Accuracy & Estimated Hw-Requirements

- Case studies:
  - Generic combinational-logic and building-blocks
  - Image-processing applications
    - The Sobel edge-detector
    - The JPEG compression
  - Artificial intelligence applications
    - Decision-tree based multiple classifier systems
    - Deep-neural networks

The Sobel edge-detector I

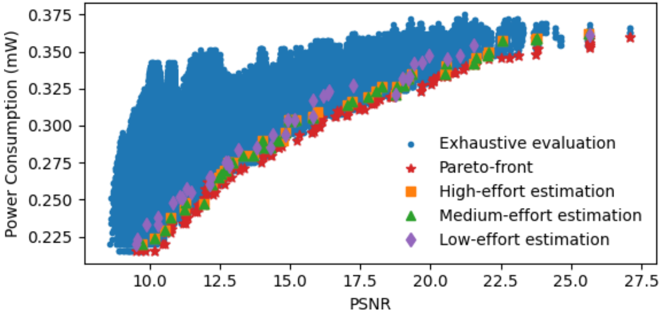
- Aim: design a hardware accelerator for the Sobel edge-detector
  - Approximate mathematical operations from the EvoApprox-Lite [9] library of components
  - Decision variables allow selecting which component from the library should be used
  - Fitness-functions:
    - Error: PSNR computed on the SIPI Image -database [1]
    - Silicon area: as the sum of the contributions of each single approximate circuit.
    - Power consumption: as the sum of the contributions of each single approximate circuit.
  - NSGA-II heuristic.
  - Design space: $\approx 4.9 \times 10^7$ configurations.
    - This allows comparing the methodology against exhaustive exploration.

The Sobel edge-detector II



PSNR v.s. Required silicon area

The Sobel edge-detector III



PSNR v.s. Power consumption

The Sobel edge-detector IV

				Absolute Distance			Normalized Distance		
Effort	**Pop.**	**Iter.**	**Time**	**Min.**	**Avg**	**Max**	**Min**	**Avg**	**Max**
Exh.	-	-	≈170h	-	-	-	-	-	-
Low	500	3	≈5min	0.013	1.58	7.6	5.9e-6	2.4e-4	1.7e-3
Med.	2000	11	≈4h	0.002	1.57	5.8	3.7e-6	6.9e-6	5.6e-4
Hig.	20000	100	≈22h	0.001	1.5	4.4	3.6e-6	6.8e-6	5.4e-4

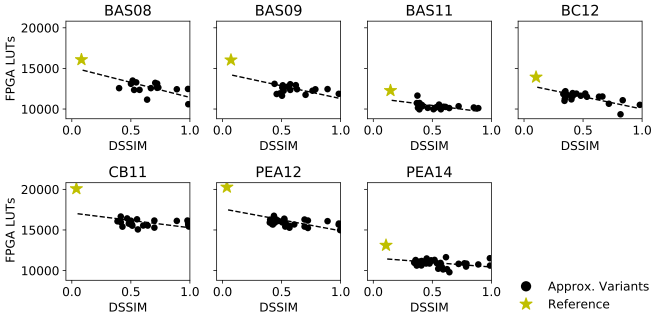
The JPEG case study I

- Aim: design a hardware accelerator for the JPEG compression
  - The Discrete Cosine Transform (DCT) is the most demanding step of the algorithm
  - Three levels of approximation:
    - At user-level: we adapt the high-frequency filter threshold
    - At algorithmic-level: we adopted multiplier-less fast algorithms (seven different algorithms are taken into consideration)
    - At hardware-level: we use approximate circuits to compute the least significant part of additions
  - Decision variables allow:
    - selecting the high-frequency threshold for the high-frequency filter
    - how many bits have to be approximate in each of the sum
    - which approximate sum implementation to be adopted

The JPEG case study II

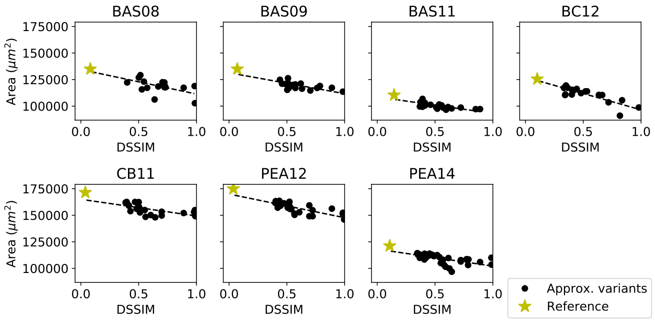
- Fitness functions:
  - Error: MDSSIM [10] computer over the SIPI Image Database [1]
  - Silicon area: estimated from the number of transistors required to implement single adder cells.
- Design space: $\approx 2.66 \times 10^{49} \approx 2^{164}$ and $\approx 1.081 \times 10^{80} \approx 2^{265}$ configurations

The JPEG case study III



Required LUTs for Xilinx Zynq-7020 FPGA

The JPEG case study IV

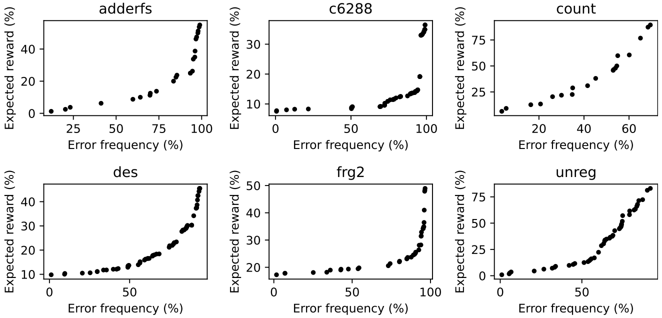


Silicon Area for 65nm FinFet

Generic logic I

- Aim: design generic combinational logic circuits
  - cut-based AIG rewriting
  - Decision variables allow selecting which cut to replace
  - Fitness-functions:
    - Error: depends on the domain (error-frequency / AWCE)
    - Silicon area: estimated from the number of AIG nodes
  - AMOSA heuristic.

Generic logic II

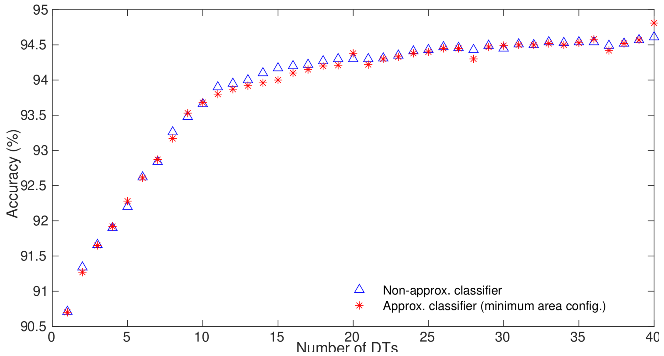


Pareto-front for combinational circuits

Decision-tree based classifiers (DTMCS) I

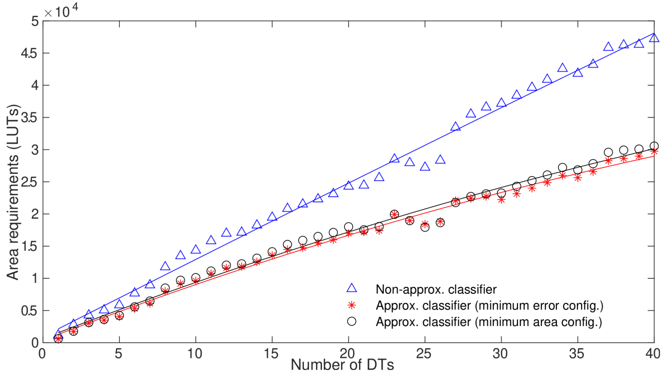
- Aim: design a hardware accelerator for DTMCSs
  - Approximation: introduced through the use of approximate comparators, designed using the precision scaling technique
  - Decision variables: allow selecting the amount on neglected bits for each of the features
  - Fitness functions:
    - Error: classification-accuracy loss (to be minimized), measured through simulations on the SPAM-base data set [5]
    - Hardware requirements: silicon area (to be minimized), estimated from the number of neglected bits
  - Design space: $52^{|F|}$ different configurations, F is the size of the feature set.
  - Number of trees ranging from 1 to 40

Decision-tree based classifiers (DTMCS) II



Resiliency varying the number of trees

Decision-tree based classifiers (DTMCS) III

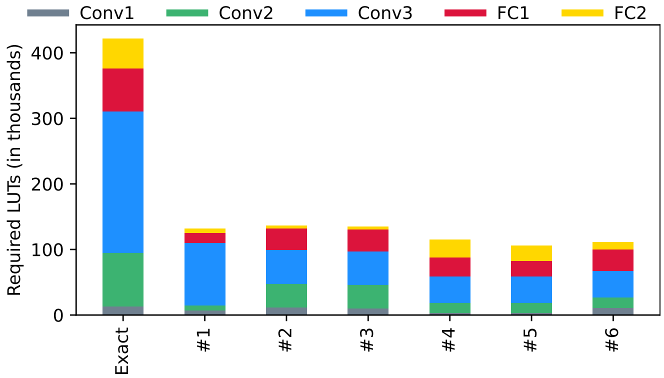


Required LUTs for Xilinx Zynq-7020 FPGA

Neural Networks I

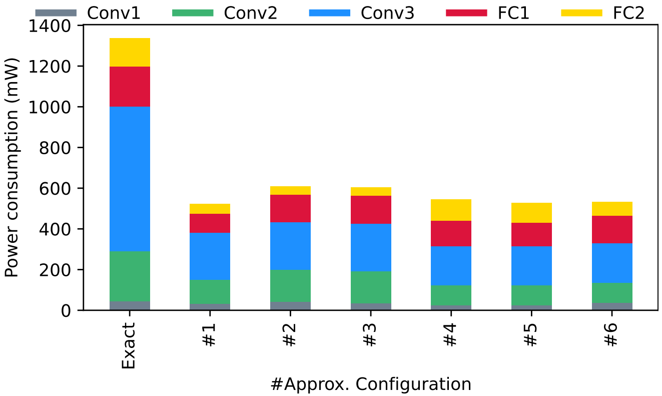
- Aim: design a hardware accelerator for NNs
  - Approximation: introduced through the use of approximate multipliers and adders, designed using the precision scaling technique
  - Decision variables: allow selecting the amount on neglected bits for each of the multiplications/additions
  - Fitness functions:
    - Error: classification-accuracy loss (to be minimized), measured through simulations on the MNIST handwritten digits dataset [8], using LeNet5 [7]
    - Hardware requirements: silicon area (to be minimized), estimated from the number of neglected bits

Neural Networks II



LUTs requirements while targeting Xilinx Virtex Ultrascale+

Neural Networks III



Power consumption while targeting Xilinx Virtex Ultrascale+ FPGA

Pubblications

- M. Barbareschi, S. Barone, N. Mazzocca. Advancing synthesis of decision tree-based multiple classifier systems: an approximate computing case study. Knowledge and Information Systems 63 (6), 1577-1596, 2021.
- S. Barone, M. Traiola, M. Barbareschi, A. Bosio. Multi-Objective Application-driven Approximate Design Method. IEEE Access, 2021.
- M. Barbareschi, S. Barone, A. Bosio, M. Traiola, J. Han. A Genetic-Algorithm-Based Approach to the Design of DCT Hardware Accelerators. ACM Journal on Emerging Technologies in Computing Systems. (Currently under review, round two)
- M. Barbareschi, S. Barone, N. Mazzocca, and A. Moriconi. A Catalog-based AIG-Rewriting Approach to the Design of Approximate Components. IEEE Transaction on Emerging Topics in Computing. (Currently under review, round two)
- M. Barbareschi, S. Barone, N. Mazzocca, and A. Moriconi. Design Space Exploration Tools (Book Chapter). (Currently under review)

Questions?

References I

[1] SIPI Image Database.
https://sipi.usc.edu/database/.

[2] Mario Barbareschi, Federico Iannucci, and Antonino Mazzeo.
Automatic Design Space Exploration of Approximate Algorithms for Big Data Applications.
In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 40–45, March 2016.

[3] dcadmin.
Rebooting the IT Revolution: A Call to Action.
https://www.semiconductors.org/resources/rebooting-the-it-revolution-a-call-to-action-2/.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan.
A fast and elitist multiobjective genetic algorithm: NSGA-II.
IEEE Transactions on Evolutionary Computation, 6(2):182–197, April 2002.

[5] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt.
Spambase Data Set.
https://archive.ics.uci.edu/ml/datasets/spambase, 1999.

[6] Scott Kirkpatrick.
Optimization by simulated annealing: Quantitative studies.
Journal of Statistical Physics, 34(5-6):975–986, March 1984.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[8] Yann LeCun, Corinna Cortes, and Chris Burges.
MNIST Handwritten digit database.
http://yann.lecun.com/exdb/mnist/, 1998.

References II

[9] V. Mrazek, L. Sekanina, and Z. Vasicek.
Using Libraries of Approximate Circuits in Design of Hardware Accelerators of Deep Neural Networks.
In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 243–247, August 2020.

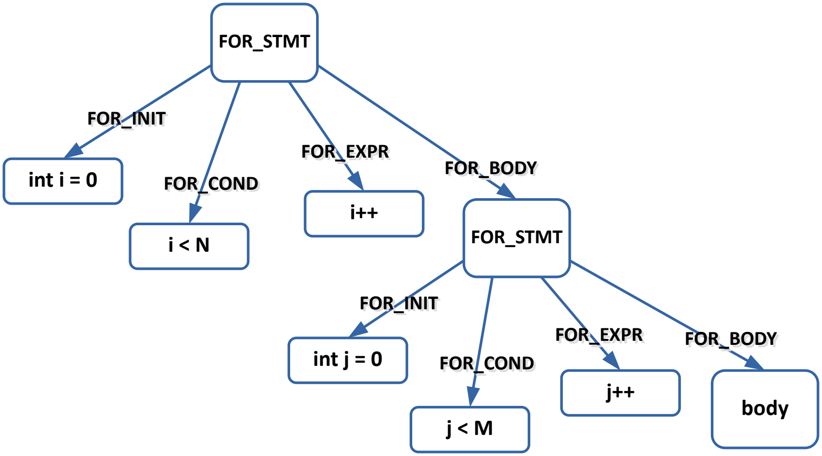
[10] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli.
Image Quality Assessment: From Error Visibility to Structural Similarity.
IEEE Transactions on Image Processing, 13(4):600–612, April 2004.

Mutators I

```c
int main (void) {
  for (int i = 0; i < N; i++) {
      for (int j = 0; j < M; j++) {
          body;
      }
    }
}
```

Listing: Precise Code

Mutators II


FOR_STMT
FOR_INIT
int i = 0
FOR_COND
i < N
FOR_EXPR
i++
FOR_BODY
FOR_STMT
FOR_INIT
int j = 0
FOR_COND
j < M
FOR_EXPR
j++
FOR_BODY
body

Mutators III

```
int main (void) {
    for (int i = 0; i < N; i+=stride1) {
        for (int j = 0; j < M; j+=stride2) {
            body;
        }
    }
}
```

Listing: Precise Code

Mutators IV

```
int ax_sum(int add1, int add2, int ax);
```

Listing: Example of approximate sum

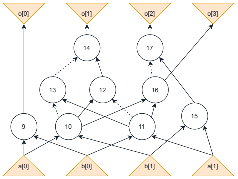
```
...
y = x + 2
z = 2 * x + 3 * y + 2;
...
```

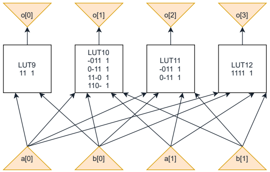
Listing: Example code to be mutated

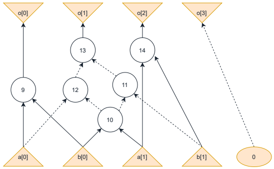
```
int ax_0 = 0;
int ax_1 = 0;
int ax_2 = 0;
...
y = ax_sum(x, 2, ax_0);
z = ax_sum(ax_sum(2 * x, 3 * y, ax_1),  2, ax_2);
```

Listing: Mutated code

Catalog-based AIG-Rewriting I







Hardware implementation of the DCT I

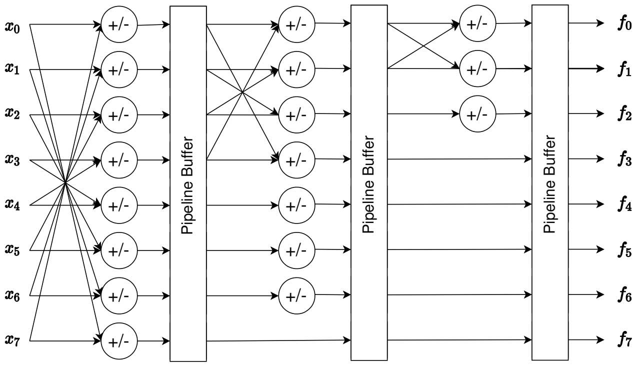
$$F = C \cdot X \cdot C' = D \cdot (T \cdot X \cdot T') \cdot D = \\ = T \cdot X \cdot T' \circ (diag(D) \cdot diag(D)')$$

$$F_Q = \lceil F \oslash Q \rfloor = \lceil T \cdot X \cdot T' \circ (diag(D) \cdot diag(D)') \oslash Q \rfloor \\ = \lceil T \cdot X \cdot T' \circ \hat{Q} \rfloor = \lceil (T \cdot (T \cdot X')') \circ \hat{Q} \rfloor$$

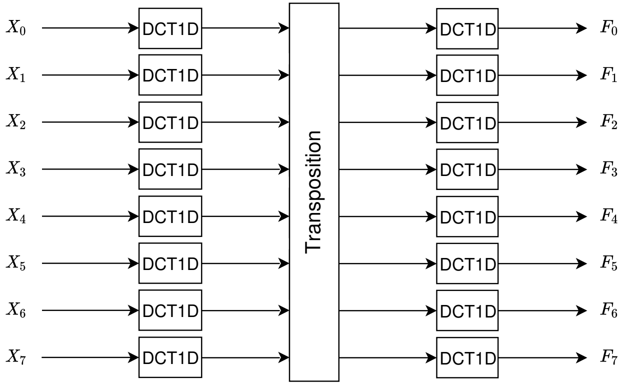
$$\hat{Q} = (diag(D) \cdot diag(D)') \oslash Q,$$

$$\begin{aligned} f_0 &= x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 & f_1 &= x_0 - x_7 \\ f_2 &= x_0 - x_1 - x_2 + x_3 + x_4 - x_5 - x_6 + x_7 & f_3 &= x_4 - x_3 \\ f_4 &= x_0 - x_3 - x_4 + x_7 & f_5 &= x_5 - x_2 \\ f_6 &= x_2 - x_1 + x_5 - x_6 & f_7 &= x_6 - x_1 \end{aligned}$$

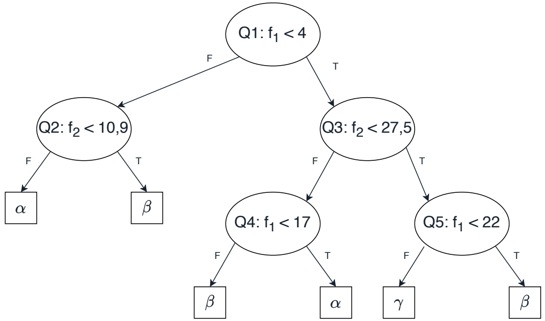
Hardware implementation of the DCT II



Hardware implementation of the DCT III



Hardware implementation of DTMCS I

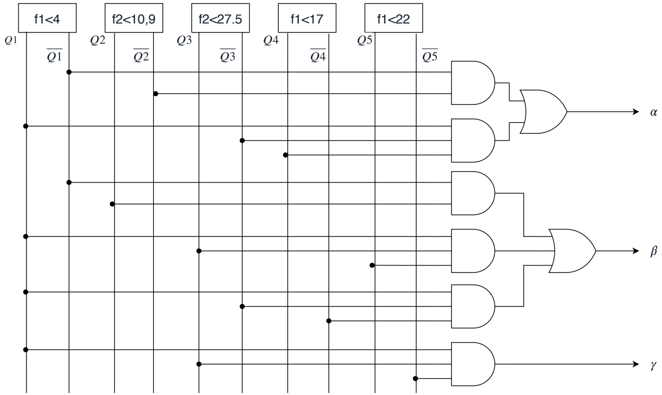


$$\alpha = (\overline{Q1} \wedge \overline{Q2}) \vee (Q1 \wedge \overline{Q3} \wedge Q4)$$

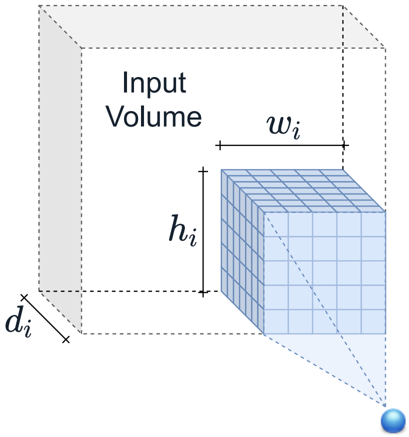
$$\beta = (\overline{Q1} \wedge Q2) \vee (Q1 \wedge Q3 \wedge Q5) \vee (Q1 \wedge \overline{Q3} \wedge \overline{Q4})$$

$$\gamma = Q1 \wedge Q3 \wedge \overline{Q5}$$

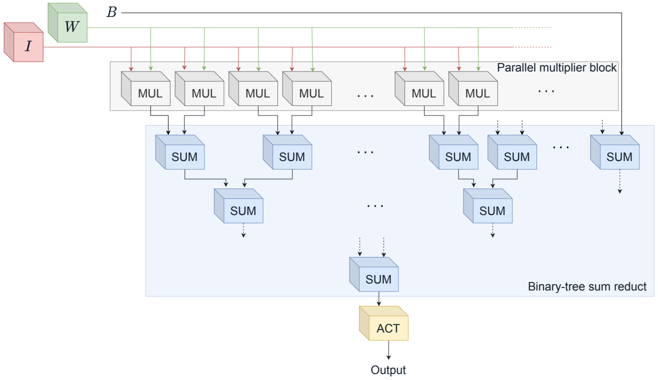
Hardware implementation of DTMCS II



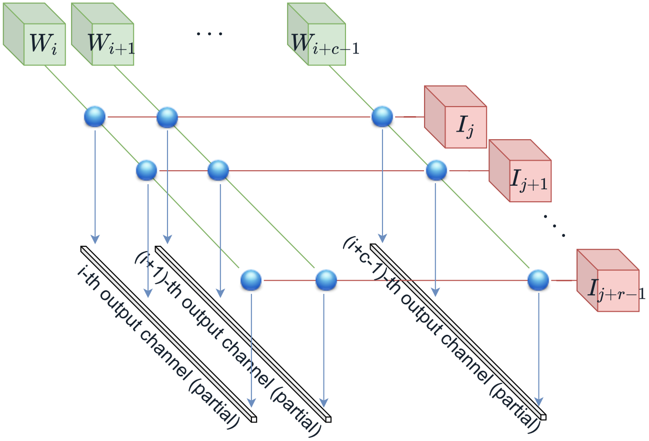
Receptive field I



Hardware implementation of NNs I



Hardware implementation of NNs II


W_i
W_{i+1}
$\cdots$
W_{i+c-1}
I_j
I_{j+1}
I_{j+r-1}
i-th output channel (partial)
(i+1)-th output channel (partial)
(i+c-1)-th output channel (partial)

Designing Efficient Computing Systems: The Approximate Computing Breakthrough

XXXIV Cycle – III year presentation

Salvatore Barone
salvatore.barone@unina.it

Tutor: Antonino Mazzeo
mazzeo@unina.it

November 5, 2021