ENERGY-EFFICIENT FEATURE EXTRACTION ENGINE AND SECURE CHIP IDENTIFICATION FOR UBIQUITOUS SURVEILLANCE

ANASTACIA B. ALVAREZ

(BS ECE, University of the Philippines, 1998)

(MS EE, University of the Philippines, 2004)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2016

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Anastacia B. Alvarez
19 December 2016

Acknowledgement

This journey has been long one, and challenging at times, and I am grateful to Him above for the give of life, love, family and friends, without whom this wouldn't have been possible. My sincere gratitude to them.

First of all, to my supervisors, for their guidance and support. Massimo, thank you for the guidance and patience especially during the critical phases of the projects, and the opportunity to work on them. Prof. Ha Yajun, thank you for accepting me to be your student, and for the advises especially during my first few years of study. Dr. Zhou Jun, thank you for the valuable feedback and advise and for sparing your time for consultations.

My heartfelt gratitude to the Professors who reviewed my research (or part of it), for their comments and valuable criticism: Prof. Xu Yong Ping, Prof. Heng Chun Huat, Prof. Yang Zhi, Dr. Rajesh Panicker, Prof. Ganesh Samudra and Prof. Vincent Lee. To all Profs in the modules I attended (official and unofficial), thank you for sharing the knowledge. To Prof. Adekunle Adeyeye and Ms. Hemamalini, thank you for helping me through the process of the NUS graduate program.

Of course this would not have been possible without the support from the University of the Philippines and the Faculty Development Program of ERDT through the Department of Science and Technology. To ate Mimi and Ms. Daezelle, for making sure we get our allowance. To my friends back home, Tess, Franz, Chrd, Marc, Benjo, Mong and to everybody at EEEI, thank you for always welcoming me each time I come home. Special mention to Louis and Rhands for being my unofficial consultants.

To Wenfeng, my sincere thanks for being a very patient and ever dependable mentor. Gopal, thanks for all your help with EQSCALE (even after you have left the group), and to Longyang, for the discussion and assistance especially with the CAD tools. The coffee, chats and good food are definitely appreciated as well, together with Kien, Saurabh, Allan, Niranjani and Jim. Thanks to Harish, for the work with ORB, and Isha for the work with the SRAM. And to the rest of the Green-IC group, thanks for company.

Finally, and most importantly, to my family (Tatang, Mommy†, Ate Ay, Kuya King, Chel, Caitlyn, Kayla and Jared), my in-laws (Tatay†, Nanay, Liza, Rhomie, Ding, Chase, Omar†, Reiza, Ria and Marco), thanks for the love and support. And to my very supportive and loving husband, Elbert, thank you for always believing and being there for me. This one's for you!

Table of Contents

Summary

Making machines more human has been a long-term research goal across different disciplines. Computer vision plays a major role in this goal by giving the machine a way to analyse and interpret an image by dividing images into smaller but meaningful segments. One critical step in computer vision is feature extraction, which identifies unique features of objects in an image or video. This could lead to a wide range of applications, including ubiquitous surveillance, which involves area monitoring, object detection, tracking, and remote sensing. For applications like surveillance, real-time processing is usually required, making the processing and analysis tasks more compute intensive, resource hungry, and therefore power consuming. To cater to battery-operated devices, or those using power from energy harvesting techniques, power consumption is pushed to sub-mW at tens of MHz frequency. Thus, designers are faced with two opposing constraints: high throughput and low energy.

An energy-quality scalable feature extraction accelerator (EQSCALE) is presented as the first chip demonstration of the Oriented FAST Rotated BRIEF (ORB) algorithm. In this accelerator, tuning knobs are introduced, allowing for adjustable balance between the energy consumption and quality of the feature extraction accelerator. As proof of concept, a 40nm testchip was designed and tested to have an energy of 55.6pJ per pixel on VGA format at 30 fps, with area of 0.55 mm^2. The effect of the different knobs on energy and accuracy, as well as some intuition on the trade-off between energy and performance is presented, to allow for scalability depending on the need of the application.

Memory also plays a role in both performance and energy consumption of the system. Leveraging on the high correlation of adjacent pixels in an image, a non-pre-charged SRAM (NPSRAM) is proposed. Compared with conventional 8T SRAM, we

show that NPSRAM can reduce energy by 30-75%, with 15% area overhead, at iso-speed.

For ubiquitous surveillance, especially with sensor nodes for detecting and tracking objects, confidential information are passed from node to node. With the ever-growing number of IoT devices and nodes, security issues like node cloning are expected to arise. There is therefore a need to ensure data authenticity, integrity and confidentiality. For this, we propose to use chip identification using physically unclonable functions (PUFs). A PUF is a function that maps an input challenge to an output response in a repeatable but unpredictable manner, leveraging on chip-specific random process variations. A novel class of mono-stable static (PUFs) for secure key generation and chip identification is presented. From a statistical quality viewpoint, the 65nm PUF testchip achieved best-in-class reproducibility and uniqueness. Energy consumption was likewise shown to be the best compared to state of the art in PUFs, at 15fJ/bit.

List of Tables

List of Figures

Chapter 1

Introduction

Computers are known to outperform humans when it comes to computations and logical operations. When it comes to analyzing images and video scenes, however, humans could easily outperform computers. Research in computer vision and image and video processing has been developing to narrow this gap [1]. Image and video processing has become a trend in the last decade, with applications ranging from image reconstruction, restoration and enhancement, image and video compression, to object classification and real-time high-resolution 3D video rendering. These applications require a lot of computations at a high rate. As such, much is required from the computing unit.

With the increasing demand for high connectivity in the internet of things (IoT) and ubiquitous surveillance, the demand for low-energy, real-time processing for computer vision applications continues to grow. Thus, designers are faced with two opposing constraints: high throughput and low energy [2]. Although desktop computers and servers are able to keep up with these demands, for mobile devices, the increasing power consumption becomes a challenge. To achieve low energy, designers have opted to aggressively scale voltages into the sub- or near-threshold region [3], [4]. However, scaling down voltage supply comes at a performance cost. To alleviate this problem, pipelining and parallelism are explored.

Section 1.1, discusses existing techniques to achieve ultra-low power (ULP) and energy-efficient designs. In Section 1.2, an overview of our research on energy-efficient feature extraction system, as well as the flow of this thesis is presented.

1.1 Energy- and Power-Limited Designs

Moore's law has predicted the scaling down of transistor sizes and increase in transistor density per chip. Although power per transistor decreases due to the decrease in transistor size, the increase in density results in an overall increase in power [5], [6]. To achieve ultra low power (ULP) consumption, aggressive voltage scaling has been the most widely used, due to the quadratic relationship between the supply voltage and power. With this decrease in V_{DD} (to reduce power and therefore energy), comes the trade-off of larger delay (therefore increase in energy). This trade-off is best illustrated in [4] and shown in Fig. 1.1, where the minimum energy point (MEP).

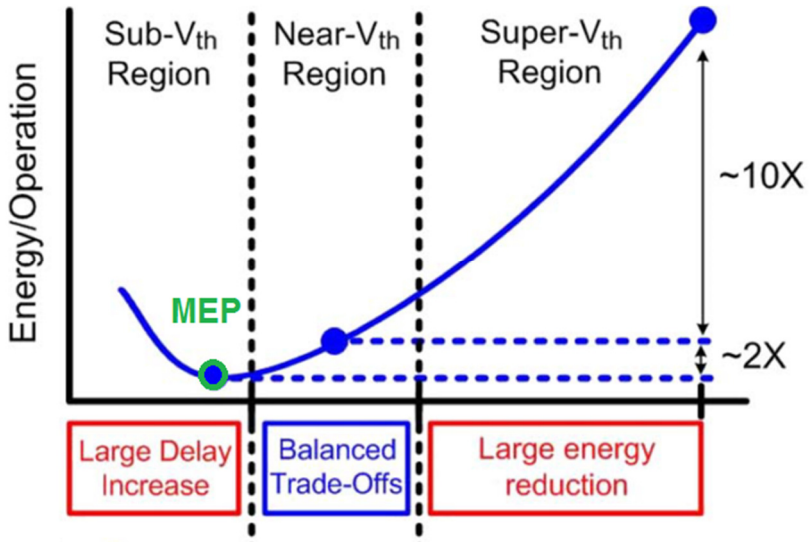


Fig. 1.1 Supply vs energy/operation, divided into super-VTH, Near-VTH and sub-VTH regions [4]

From Fig. 1.1, working in the super-V_{TH} region is the conventional region and results in high energy reduction with slight reduction in V_{DD}. In the sub-V_{TH} region, delay increases exponentially with V_{DD}. The near-V_{TH} region is where a good balance is between the change in energy and change in delay. The MEP may or may not be in the near-V_{TH} region, but the minimum delay point (MDP) is at a higher energy point [7], [8]. Traditional designs have been targeting for the MDP, until energy constraints started to be tighter, that they proposed the energy-delay (ED) product as a metric [8].

Depending on the application, some may favour energy over delay, using E^2D or E^3D as metrics, or favour delay using ED^2 or ED^3.

Energy can be reduced at different levels of the design. At the architecture level, parallelism, pipelining and instruction set architecture (ISA) design have been explored [3], [6], [9]–[11]. At the circuit level, sizing, layout, and even new circuits have been used to allow the system to operate properly at low voltages [7], [10], [12], [13].

1.2 Thesis Overview

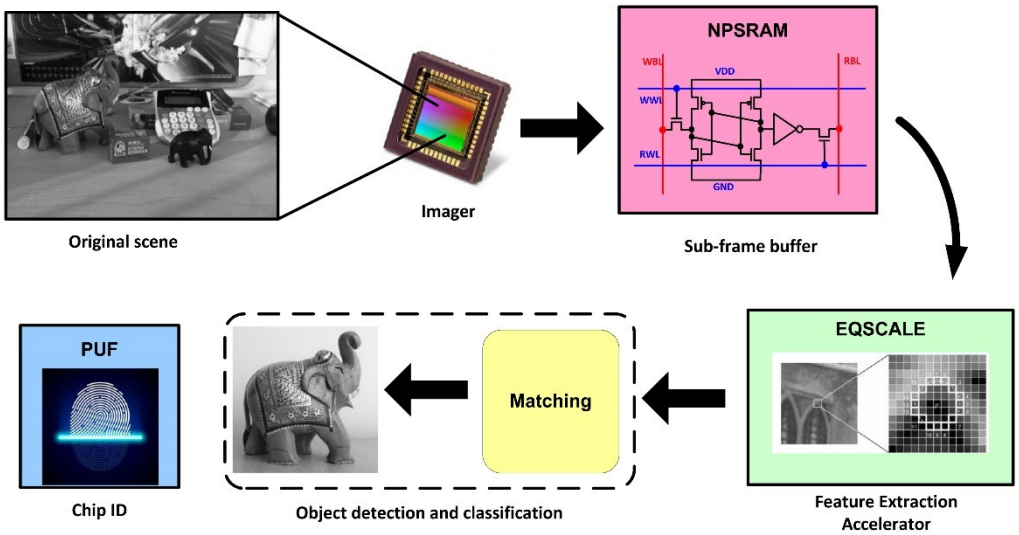


Fig. 1.2 Feature extraction system overview

An overview of the feature extraction system (Fig. 1.2) shows an imager passing data to a sub-frame buffer. Input rate to the buffer is assumed to be 30 frames/sec, by default. These pixels are temporarily stored in a sub-frame buffer and processed by a feature extraction accelerator, with an assumed input rate of 1 pixel per cycle. The feature extraction passes keypoints to an object classifier to identify objects from a database. A chip identification is needed to authenticate the device and the data it sends, a task that is essential in visual surveillance systems. To control all these processes (handshake between blocks, data security and chip authentication) a low-energy microcontroller (not shown in the figure) is needed.

At the heart of the feature extraction system is EQSCALE – an energy-quality scalable feature extraction accelerator. Chapter 2 starts with some basics and the state-of-the-art in feature extraction algorithms. Metrics for comparison as well as literatures comparing different algorithms are also presented.

Simulation results comparing the performance of three candidate algorithms, namely SIFT [14]–[16], SURF [17], [18] and ORB [19] are presented in Chapter 3. In this chapter, the concept of energy-quality scalability is introduced into the feature extraction accelerator (EQSCALE) through tuneable knobs, allowing for adjustable balance between energy and quality.

Details of the hardware implementation and chip testing results are presented in Chapter 4. Using benchmark images, a quantitative analysis of the trade-off between energy and quality for every knob, as well as for a combination of the knobs is shown.

Between the imager (currently external to the system) and the feature extraction accelerator (see Fig. 1.2) is a sub-frame buffer (as opposed to an external full-frame buffer), which is best implemented using an SRAM, for best balance between area and speed. Chapter 5 presents simulation results for a non-precharged SRAM (NPSRAM), which leverages on the high correlation of adjacent pixels in image. Different topologies that could allow for non-precharged bitlines and their corresponding issues and drawbacks are analyzed. The chapter is concluded with a comparison between the conventional 8T SRAM and the NPSRAM.

Chapter 6 covers the concept of physically unclonable functions (PUFs) and their applications in hardware security, such as for chip identification. Metrics and discussion of the state-of-the-art are likewise covered. Finally, a novel class of static, monostable PUFs is presented, together with the results and comparison with other PUFs.

Results for the energy-efficient microcontroller design utilizing a customized standard cell library for sub-/near-threshold operation is presented in Chapter 7. Finally, we summarize the contribution of this thesis in Chapter 8. Discussion on possible future work, including the object detection and classification block in Fig. 1.2, is also included.

Chapter 2

Feature Extraction Algorithms and their Suitability for Energy-Autonomous Computer Vision

One critical step in computer vision is feature extraction. Given an image or a video scene, we, as humans with all the stored experience and complex visual system, can easily distinguish and classify the objects within it, as well as a perceived depth (i.e., foreground and background). In computers, on the other hand, an image is just a 2-D array of pixel intensities. To be able to identify objects, it has to extract relevant information from the image or video frame. Similar to human vision system, feature extraction for computer vision is a continuous (always on) process, processing videos or scenes frame by frame. As such, aside from accuracy or quality of the algorithm, its complexity and therefore energy consumption also has to be considered for it to be suitable for energy-autonomous computer vision systems. It should be noted that recent researches on deep neural networks (DNNs), or specifically convolutional neural networks (CNNs), have shown remarkable results close to human accuracy in image classification and recognition [20]. However, current hardware implementations of CNNs occupy very large silicon area and suffer from high power consumption due to their complexity [21]. As such, researchers are still working on implementation optimizations for DNNs to reduce their silicon footprint and power consumption.

Feature extraction goes through three major steps in analyzing an object in an image: (1) detection, (2) description, and (3) matching. To detect objects, the computer needs to find a point or set of points that may be unique to the object, such as edges, corners and blobs. These are called interest points, keypoints or features. The next step

is representing these features so that they can be scale and rotation invariant, and unique to the feature. This is called description. Finally, matching these described points usually requires computing distances to see if they match.

Several detectors and descriptors have been proposed since the middle of 1900s. Most common of which are the Scale Invariant Feature Transform (SIFT) [14] and the Speeded Up Robust Features (SURF) [18]. Some background on these feature extraction algorithms will be presented in Section 2.1. Section 2.2 will discuss the metrics used in comparing these algorithms, as well as some comparison done using these metrics. The state of the art in hardware implementation of feature extraction accelerators is discussed in Section 2.3.

2.1. Background

The concept of feature detection was first proposed in 1950s, in their efforts to understand our complex visual system [22], [23]. Its application in computer vision was first demonstrated a decade after in the Summer Project [24], with the goal of detecting an object. Different algorithms for feature extraction have been proposed thereafter, using them for various applications, such as image detection, classification and tracking, image stitching, and augmented reality, to name a few.

Edges are one of the most intuitive features to detect objects, allowing the algorithm to draw the outline of the object. One way to detect edges is using Gaussian filters, to highlight abrupt intensity changes in the image [25]. Lindeberg [26] suggested that feature that catches the eye most are those that are stable at higher scales (greater distance). They then proposed the use of the extrema of the Laplacian of Gaussian (LoG) as an interest point detector to be scale invariant. One of the most popular interest point detector and descriptor today is the Scale Invariant Feature Transform (SIFT) [14]–[16]. Based on the work in [26], Lowe [16] proposed an approximation of LoG by creating an image pyramid through levels of Gaussian

smoothed images and then taking the difference of adjacent Gaussian smoothed images to create the difference of Gaussian (DoG). A candidate point is then considered an interest point if it is the extrema (maximum or minimum) among its 8 neighbours within the scale and the 9 neighbours each on the top and bottom scales (total of 26 neighbour pixels in the 3x3x3 region). Fig. 2.1 illustrates the image pyramid and the interest point detection.

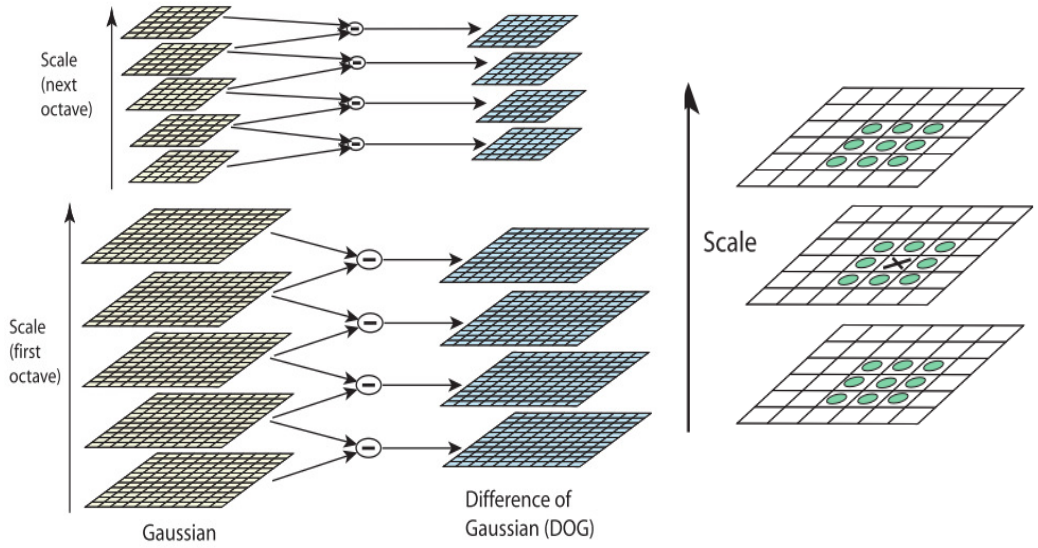


Fig. 2.1 SIFT Interest Point Detection. An image pyramid (left) is created through levels of gaussian smoothed images (scales) and sub-sampled to create the next octave. A candidate point is considered an interest point if it is the extrema within the 3x3x3 difference of Gaussian (DoG) region centered at the candidate point (right).

For a feature detected at location x,y, the magnitude, m, and orientation, θ, of pixels around the interest point are calculated using the equations in Eq. 2.1 and Eq. 2.2, respectively, where $p(x,y)$ is the pixel intensity. These are used for description, as illustrated in Fig. 2.2. The descriptor window is then divided into 4x4 sub-windows. Eight orientation bins are taken per sub-window, adding the magnitudes within the same orientation bin. These magnitudes are then concatenated to form the 128-vector (8 orientation bins x 4x4 sub-windows) SIFT descriptor.

$$m(x,y) = \sqrt{p(x+1,y) - p^2(x-1,y) + p(x,y+1) - p^2(x,y-1)} \quad (2.1)$$

$$\theta(x,y) = tan^{-1}\left(\frac{p(x,y+1)-p(x,y-1)}{p(x+1,y)-p(x-1,y)}\right) \quad (2.2)$$

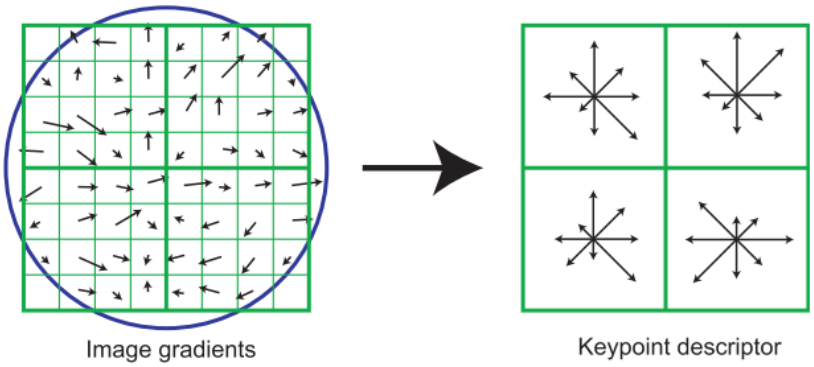


Fig. 2.2 SIFT keypoint description. Gradient magnitude and orientation is computed at each point in a region around the interest point (left). Samples are then accumulated into orientation bins over 4x4 sub-regions (right). Orientation magnitudes are then concatenated to form the descriptor vector.

Although SIFT was proposed more than a decade ago, it still remains popular because of its superior performance. An almost similar but slightly simplified version of the detector and descriptor is the Speeded Up Robust Features (SURF) [18]. Instead of using DoG as an approximation of LoG, they simplified the smoothing through box filters, as shown in Fig. 2.3. Image scales are likewise created using enlarged filter boxes. Similar to SIFT, SURF interest points are also identified by finding the extrema within the 3x3x3 neighbourhood.

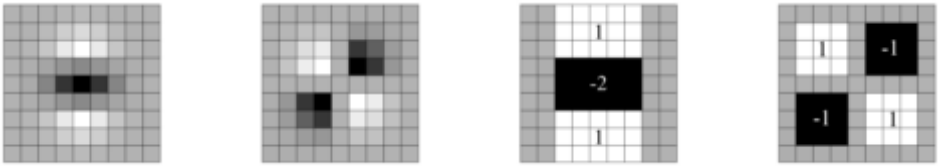


Fig. 2.3 SURF Box Filters. From left to right, discretized Gaussian second order derivatives in the y-direction, and xy-direction, and their approximations (grey regions are zeros).

The descriptor is done using Haar wavelet responses in the x- (denoted as dx) and y- (denoted as dy) directions. The descriptor vector becomes a little less complex,

that instead of having the orientation bins, the vector is formed by taking Σdx, Σ|dx|, Σdy and Σ|dy| for each sub window and concatenating these values, to form the 64-vector (4 parameters x 4x4 sub-windows) descriptor. Details of this vector is shown in Fig. 2.4.

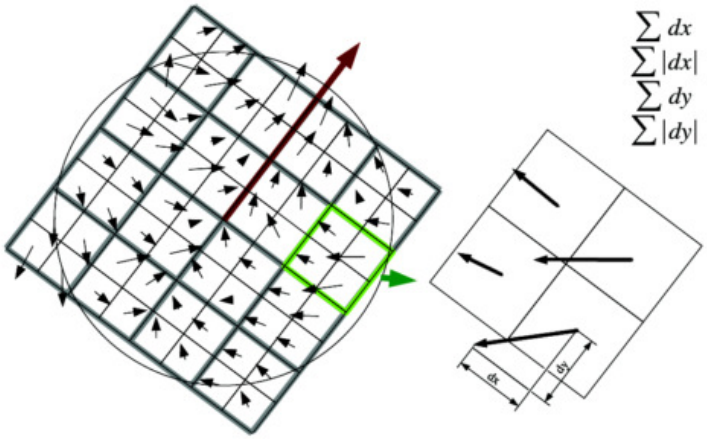


Fig. 2.4 SURF Descriptor. The grid is oriented along the dominant orientation. The cumulative response along the x- and y- directions for each sub window are computed. The descriptor vector is formed by concatenating the Σdx, Σ|dx|, Σdy and Σ|dx| for each of the 4x4 sub window.

Corners are also useful features to detect objects in an image. The work in [27] uses a combination of corners and edges to isolate objects from backgrounds. Since Gaussian filters are complex, they used a simpler [-2 -1 0 1 2] filter [1] to approximate the effect of Gaussian filtering. Another popular corner detector is the Features from Accelerated Segment Test (FAST) [28], [29]. In Fig. 2.5, the idea is to compare the intensity of the candidate pixel or interest point (labelled *C* in the figure) with those on the circumference of a circle around it (highlighted numbered pixels in the figure). Each of the surrounding pixels is labelled black if it is less (darker) than the candidate pixel by at least a threshold value; it is labelled white if it is greater than the candidate pixel by at least a threshold value; and grey if its intensity is just within a threshold of the candidate pixel intensity. A candidate point is then considered a feature if at least n ($8<n<13$) contiguous pixels are either all labelled white or all labelled black.

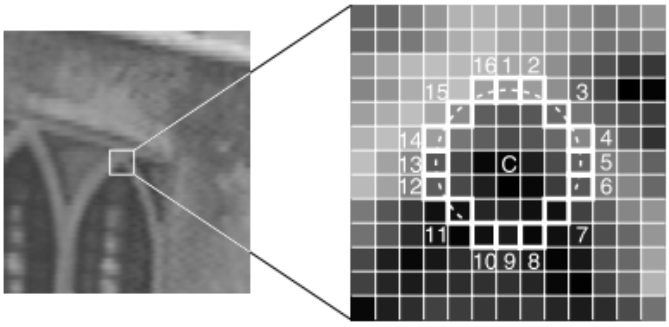


Fig. 2.5 FAST interest point detection. Point C is compared to 16 points around it to determine if C is an interest point.

FAST is only a detector. That is, it detects only the location of an interest point – it does not represent this interest point into a value or vector that can be matched with points on a different image or in the database (for the case of objection recognition). In this case, the description is done using other available algorithms. A commonly used descriptor for FAST is the Binary Robust Independent Elementary Features (BRIEF) [30]. BRIEF is done by defining a test τ on a patch p of size SxS as in Eq. 2.3, where $p(k)$ is the pixel intensity of p at k. The BRIEF descriptor $f(p)$ is then the n_d-dimensional bitstring in Eq. 2.4, where n_d is 128, 256 or 512.

$$\tau(p; j, k) = \begin{cases} 1 & if\ p(j) < p(k) \\ 0 & otherwise \end{cases} \tag{2.3}$$

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; j_i, k_i) \tag{2.4}$$

The problem with FAST-BRIEF is that it is neither scale nor rotation invariant. The group from Willow Garage proposed a FAST-BRIEF variant that is scale and rotation invariant. They called their algorithm the Oriented FAST Rotated BRIEF (ORB) [19]. Like SIFT and SURF, scale-invariance is achieved by creating an image pyramid and getting features points at each scale. Rotation-invariance, on the other hand, is achieved through the identification of orientation of the corner for each interest point, and constructing the descriptor vector based on this orientation. Orientation is done using the concept of moments and centroid. The moment, m_{pq}, of a patch and the corresponding centroid, C, are defined in Eq. 2.5 and Eq. 2.6, respectively. Having the

centroid, a vector is formed, connecting the center, *O*, and the centroid, *C*. The orientation, θ, is then given in Eq. 2.7. Given the *x*,*y* locations for the test bitstring in Eq. 2.3 and the patch orientation in Eq. 2.7, the locations are steered to get the new feature vector.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.5}$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.6}$$

$$\theta = tan^{-1}(m_{01}, m_{10}) \tag{2.7}$$

2.2. Metrics and Comparison

Several papers [31], [32] have compared different detectors and descriptors. In [31] they focused on the matching performance of the descriptors, using recall and 1-precision (shown in Eq. 2.8 and Eq. 2.9, respectively) as their metrics. The number of correspondence would be the number of pairs identified as match using the distance comparison.

$$recall = \frac{\#correct\ matches}{\#correspondences} \tag{2.8}$$

$$1 - precision = \frac{\#false\ matches}{\#correct\ matches + \#false\ matches} \tag{2.9}$$

Using the data set from [33], their comparison results (arranged according to decreasing number of nearest neighbour correct matches) are shown in Table 2.1. A snapshot of the data set is also shown in Fig. 2.6. It should be noted that the gradient of location and orientation histogram (GLOH) is a descriptor proposed by the group, which uses principal components analysis (PCA) for matching. It can be seen in the table that GLOH performs best in their 2 metrics, followed closely by SIFT.

Table 2.1 Comparison Results [31]

Descriptor	Recall	1-precision	#correct matches
GLOH	0.25	0.52	192
SIFT	0.24	0.56	177
Shape context	0.22	0.59	166
PCA-SIFT	0.19	0.65	139
Moments	0.18	0.67	133
Cross Correlation	0.15	0.72	113
Steerable filters	0.12	0.78	90
Spin images	0.09	0.84	64
Differential invariants	0.07	0.87	54
Complex filter	0.06	0.89	44

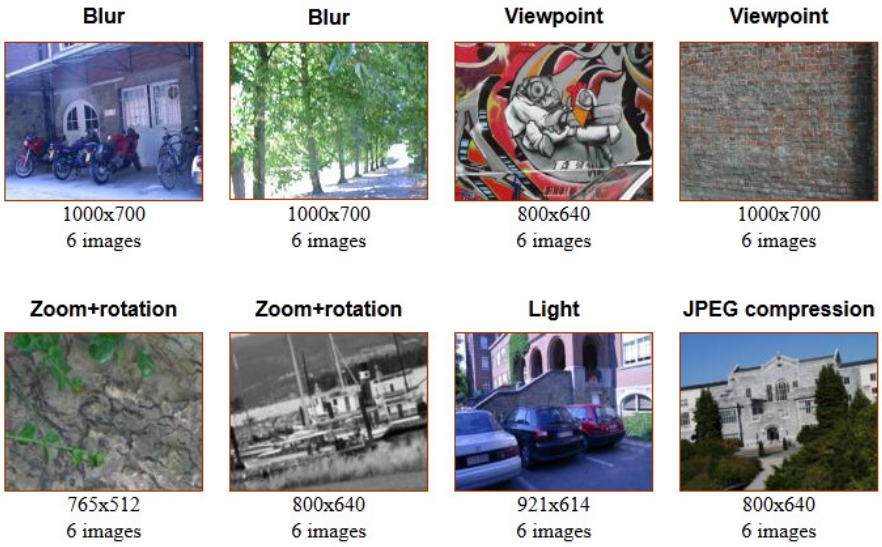


Fig. 2.6 Benchmark images used for comparison

With regards to Table 2.1, it should be noted that the *recall* metric reflects the same ranking as the number of correct matches, with the higher value corresponding to more correct matches. However, the absolute value of *recall* does not give any intuitive meaning. In [32], SIFT, SURF and PCA-SIFT, were compared using the same data set as in Fig. 2.6. Aside from execution time, they also used *repeatability* (shown in Eq. 2.10) as a metric, where $C(I_1,I_2)$ is the correspondence between image I_1 and I_2, and m_1 and m_2 are the number of features for I_1 and I_2, respectively. Their results show that SURF is the fastest, while SIFT is the best in terms of repeatability with scale, rotation and blur changes. SURF, however, is the best in terms of repeatability with change in illuminations. It should be noted at this point that while *recall* is a metric for descriptors

(how well the representation uniquely identifies the feature), *repeatability* is a metric for detectors (whether the same feature is identified under different variations of the image).

$$r_{1,2} = \frac{C(I_1,I_2)}{mean(m_1,m_2)} \tag{2.10}$$

A comparison of the detectors and descriptors were also done using the OpenCV library [34], [35]. In terms of keypoint matching, they evaluated the average number of detected features as well as the percentage tracking, for the different feature detection algorithms. These are shown in Fig. 2.7 and Fig. 2.8, respectively. We can see from Fig. 2.7 that FAST detects more than 7x features compared to others. For this simulation, ORB always gives 702 features, which is set in the algorithm. In Fig. 2.8, we can see that ORB is almost able to track all the detected features. In terms of rotation invariance, they showed that SIFT and ORB perform best, as can be seen in Fig. 2.9. In terms of scale invariance, on the other hand, SIFT and SURF perform best (Fig. 2.10).

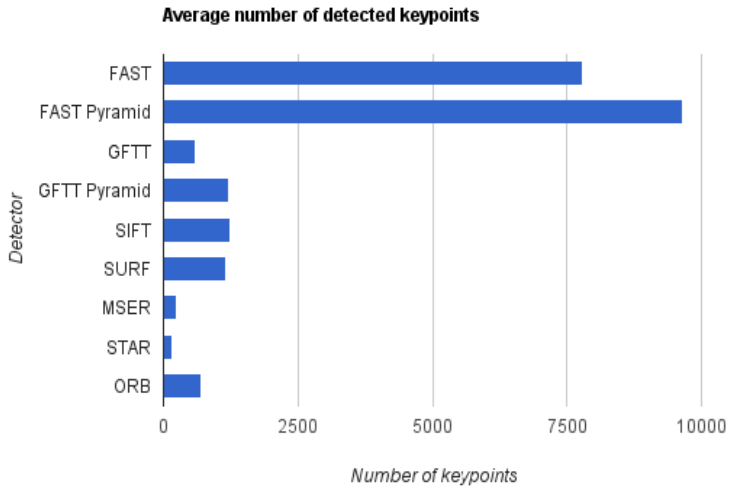


Fig. 2.7 Average number of detected keypoints

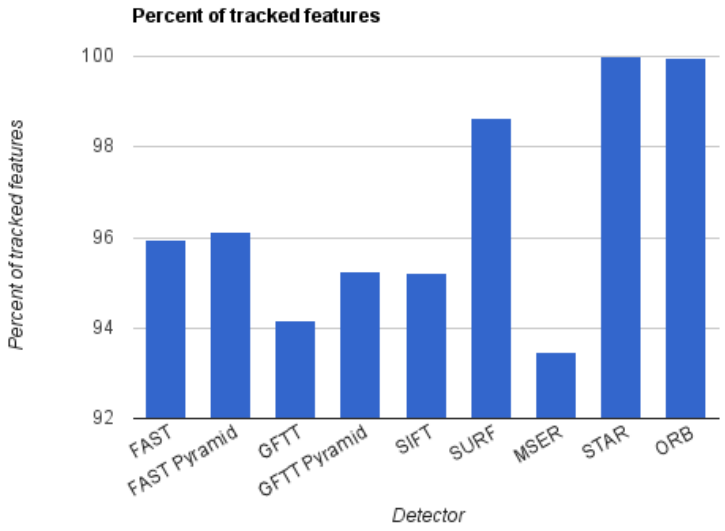


Fig. 2.8. Comparison for scale invariance

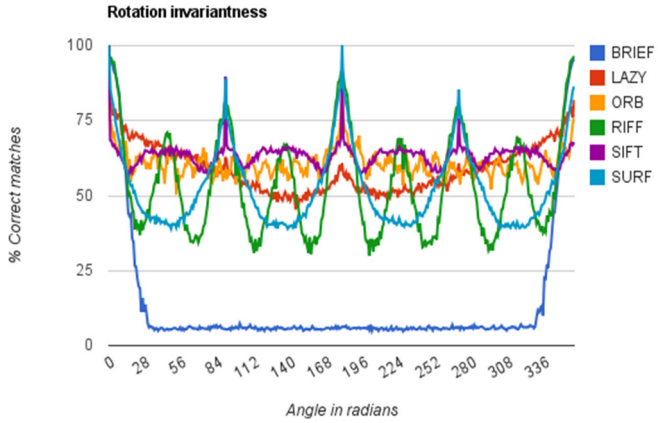


Fig. 2.9. Comparison for rotation invariance

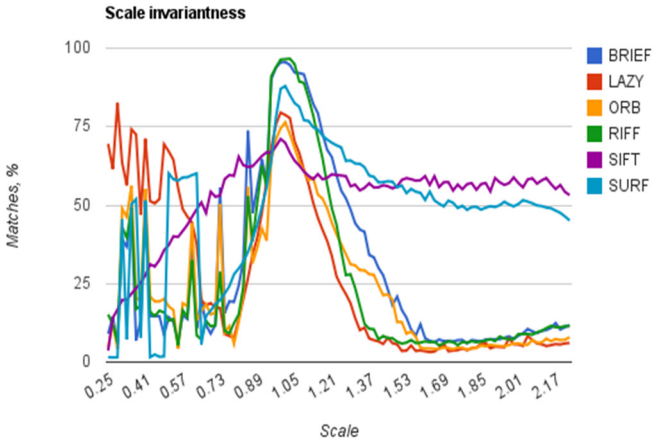


Fig. 2.10. Comparison for scale invariance

We can see from both results that SIFT does seem to be a good candidate for a feature extraction algorithm. However, the drawback is the speed or execution time, as was already shown in [32]. Using the OpenCV library, the same results are also obtained, as shown in Fig. 2.11 and Fig. 2.12. The performance test was done on Mac Book Pro 2.2 with Core 2 Duo 2.13 GHz platform.

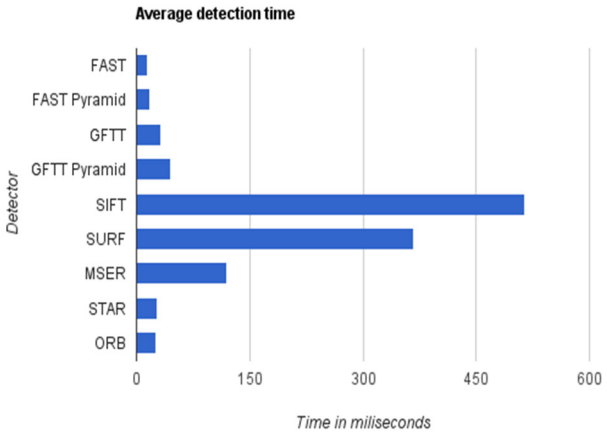


Fig. 2.11. Comparison of detection time

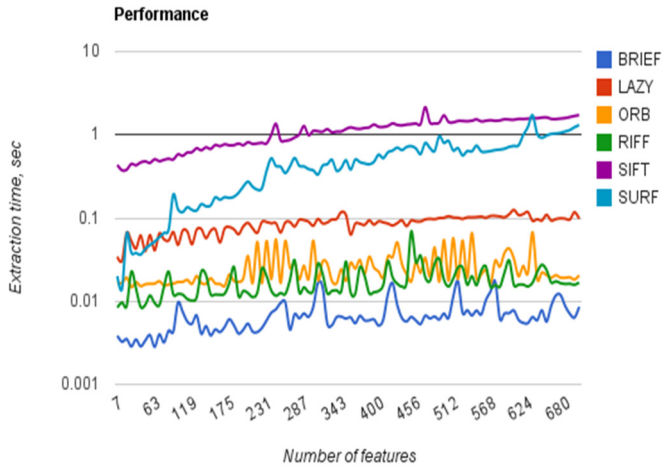


Fig. 2.12. Comparison of total execution time

Finally, performance of these descriptors using a video sequence was compared, and revealed that although SIFT and SURF still perform best, their performance still needs improvement. This result is shown in Fig. 2.13.

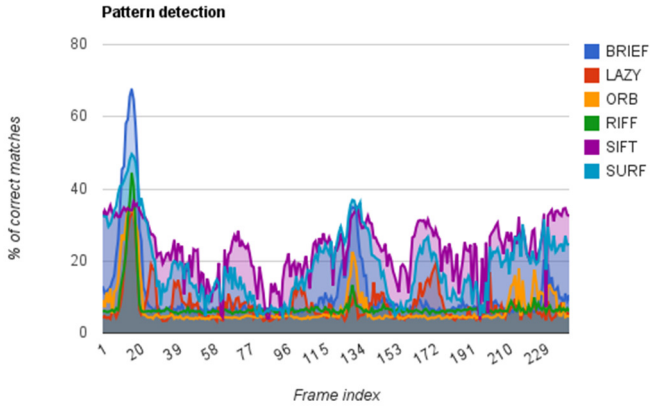


Fig. 2.13. Performance comparison using video sequence

2.3. State-of-the-Art Hardware Implementations

In this section, we discuss state of the art hardware implementations of the feature extraction methods discussed in the previous section. Specifically, we have ASIC implementations of SIFT, SURF, FAST-BRIEF, and ORB FPGA implementations. We conclude this section with a simple comparison of these implementations.

2.3.1 SIFT

As was evident from the previous sections, SIFT has complex iterative computations. It was shown in [36] that more than half of the operations in SIFT is in Gaussian filtering. Their breakdown is shown in Fig. 2.14. As such implementations of SIFT are typically done using massively parallel SIMD processors [36]–[42]. To reduce computational complexity in the implementation of the SIFT algorithm, implementations typically include a region of interest (ROI) detector and do the feature extraction only on a portion of the image [37]–[41]. As an example, authors in [39], [40] used what they call the Unified Visual Attention Model (UVAM), where they process keypoint detection only on some region of interest (ROI). Their model is illustrated in Fig. 2.15. We can see from the figure that their initial ROI selection is based on some saliency mapping. The ROI is then adjusted based on feedback from initial matching.

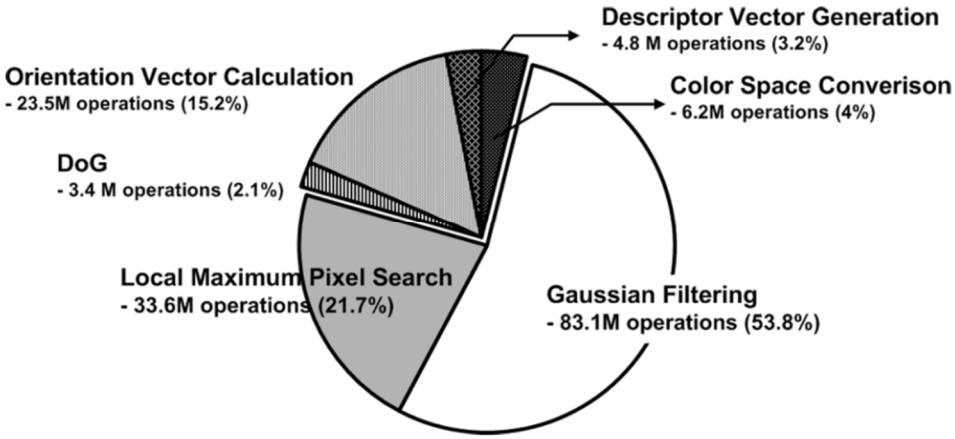


Fig. 2.14. Breakdown of computational requirement in SIFT [36]

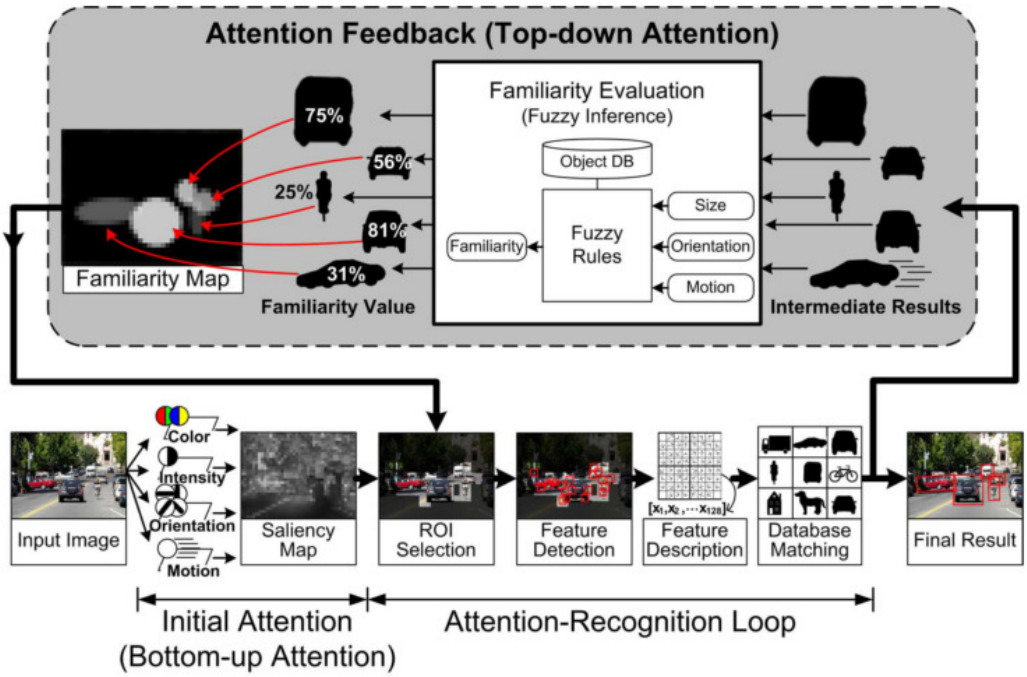


Fig. 2.15. Unified Visual Attention Model (UVAM)

To achieve 30 fps with 640x480 resolution, massive parallelism using 4 single input multiple data (SIMD) vector processing elements and 32 multiple input multiple data (MIMD) scalar processing elements was implemented. Their results show 345mW power consumption in 0.13um process. A later version of their design in [43] improved on the UVAM by proposing the Context-Aware Visual Attention Model (CAVAM), which incorporates temporal similarities between successive frames. For this implementation, 31 heterogeneous cores -- with 4 simultaneous multithreading (SMT) cores for feature extraction, were used. Increased hardware utilization was

accomplished with their 5-stage pipeline operation, as shown in Fig. 2.*16*. Each image is divided into 16x16 tiles, and each tile is assigned to a thread in each of the SMT cores for feature extraction. Their results show that they can process 30 fps of 720x1280 resolution with an average power of 320mW in 0.13um process.

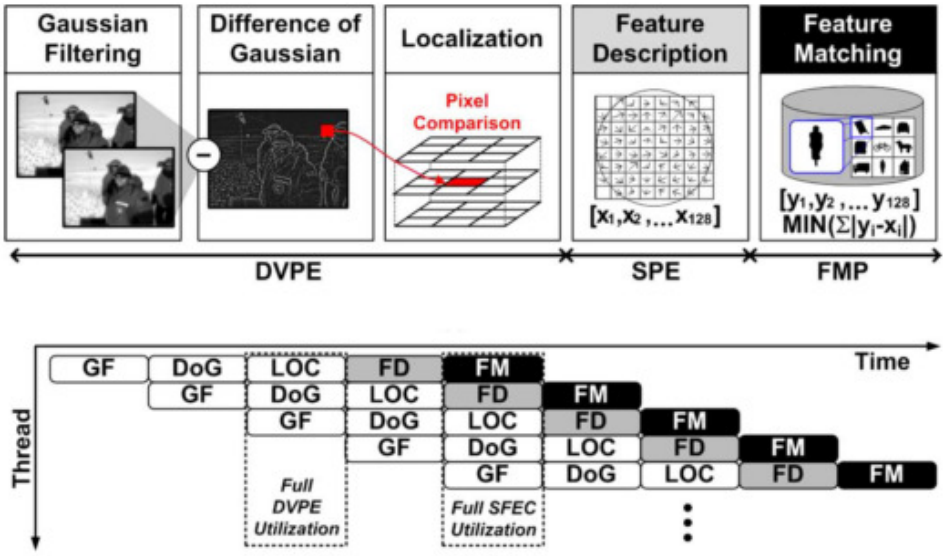


Fig. 2.16. SIFT 5-stage Pipeline Operation

2.3.2 SURF

Jeon, et. al. [44], [45] proposed to use SURF as their feature extraction engine for micro autonomous vehicle (MAV) navigation system. Unlike the work in [39] and [43], they used full frame extraction of feature point, without ROI detection. One simplification they did with the SURF algorithm is that instead of using multiple octaves with 4 scales per octave, they used a single octave with 5 scales. This is justifiable as they are targeting only 640x480 frame sizes, and therefore subsampling the image may not give very relevant information. To reduce the storage requirements, they divided the image into 11 sections with 88 pixels of overlap. Instead of SRAMs, they used a FIFO for image storage, and duplicated the image integrator for the descriptor block to eliminate the need of storing the integral image itself. In this way,

they are exchanging the additional image integrator with that of the SRAM for the integral image. Their architecture is shown in Fig. 2.17.

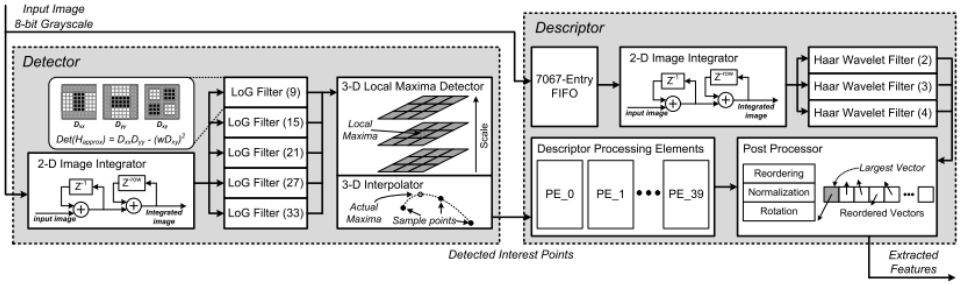


Fig. 2.17. SURF Feature Extraction Architecture

To further reduce energy consumption, they proposed a hybrid FIFO architecture using shift latches with balanced leakage compensation technique. They also operated their system in 470mV supply voltage to further reduce their power consumption. Summary of their results is shown in Table 2.2.

Table 2.2 SURF Implementation Summary

Technology	28nm LP CMOS
Vdd	470mV
Clock Freq.	27MHz (102FO4)
Core Area	0.85x2.61mm^2
Input Video	640x480 30fps
Power	2.7mW
Performance	149.3GOPS
Efficiency	55.3TOPS/W

2.3.3 FAST-BRIEF

Park, et. al. [46] implemented FAST with BRIEF, using pattern-based matching. After labelling the surrounding pixels as white (brighter than the center pixel by a threshold), black (darker than the center pixel by a threshold) or grey (intensity within threshold from center pixel), they assemble the labels into a string and compare with a string of all whites and a string of all blacks (both of length n). The block diagram of the detector is shown in Fig. 2.18. To speed up the process, they included an early rejection hardware, which detects patterns that are definitely not corners. This is done

by examining the four compass directions (pixels 1,5, 9 and 13 of Fig. 2.5). To be a corner in FAST-12 (n=12), at least 3 of these pixels have to be all white or all black. In FAST-9 (n=9), at least two contiguous compass directions must be either both white or both black. Otherwise, the candidate point is rejected as not a corner. With this implementation, a segment test requires 1-3 cycles to complete. They also proposed a unified hardware platform for interest point detection of FAST and matching with BRIEF (Fig. 2.*19*). With this unified hardware platform, some resources (such as memory storage) are shared between the interest point detection and matching hardware, at the same time, load is somewhat balanced, resulting in an even pipeline operation. Table 2.3 shows their implementation summary. It should be noted that more than 90% of their chip's area is occupied by the SRAM for the descriptor buffer.

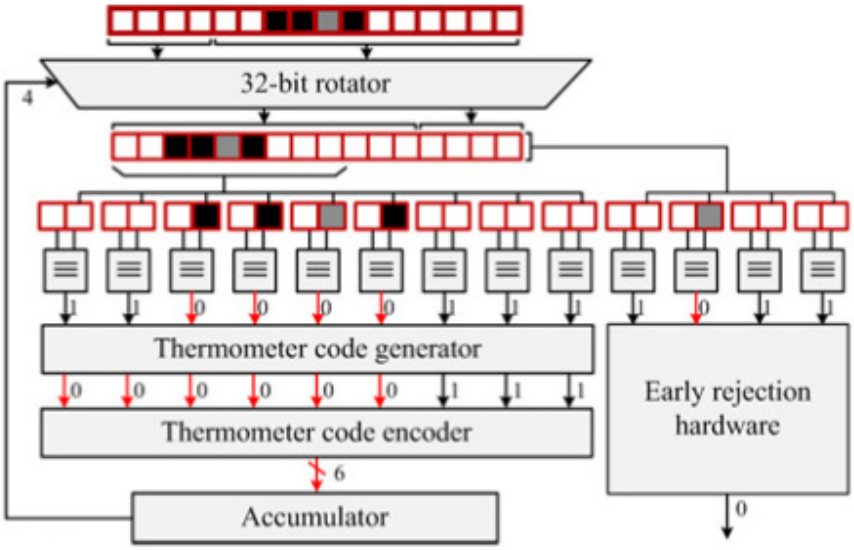


Fig. 2.18. FAST Corner Detector

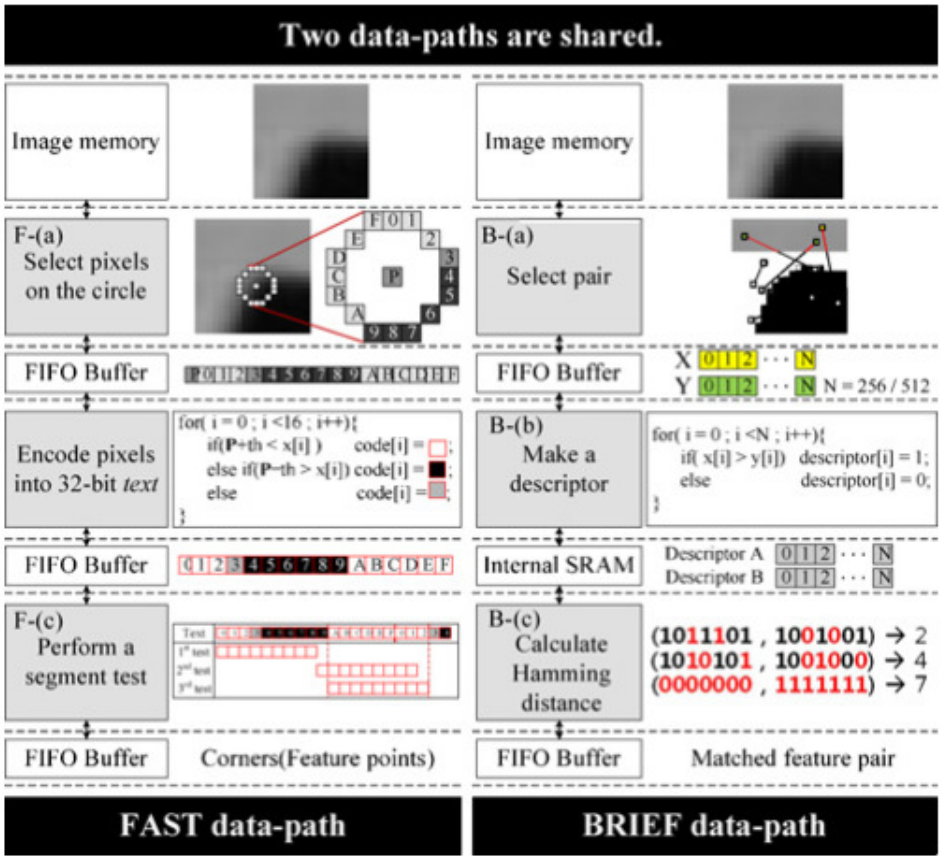


Fig. 2.19. FAST and BRIEF Unified Hardware Platform

Table 2.3 FAST-BRIEF Implementation Summary

Process technology	0.13-μm CMOS (1P6M)
Supply voltage	Core: 1.2 V, I/O: 3.3 V
Chip size	$4.0 \times 4.0\,\text{mm}^2$
Core size	$3.2 \times 3.2\,\text{mm}^2$
Logic gate count	861 K (2-input NAND gate) Including SRAM
On-chip SRAM	128 kbyte (Descriptor buffer)
Operating frequency	200 MHz
Power consumption	182 mW

2.3.4 ORB

To the best of our knowledge, ASIC implementation of the ORB algorithm has not yet been proposed. There are, however, several FPGA implementations such as the works in [47]–[50]. A comparison between SURF and ORB was done and it was shown that ORB is 2-3x faster than SURF while occupying ~4x less FPGA area.

Chapter 3

ORB Algorithm and IC Design Considerations

Having established the necessary background on feature extraction algorithms and their metrics in the previous chapter, this Chapter continues with simulation results of the ORB algorithm implementation. Section 3.1 will cover some simple comparison among SIFT, SURF and ORB, to justify the choice of implementing ORB as our feature extraction algorithm. Analysis of the effect of individual knobs is presented in Section 3.2. Section 3.3 covers the hardware model to show the effect of implementation simplification or approximations, and how the chosen knobs interact with the hardware implementation.

3.1. Comparison of SIFT, SURF and ORB

OpenCV [51] was used to compare results for SIFT, SURF and ORB. Although comparison of some (or all) of these algorithms have been done in previous works, as discussed in the previous chapter, doing the comparison on our own allows us to verify the comparisons as well as identify weaknesses and strengths of algorithms that would likewise not be indicated in the comparisons. Knowing that each algorithm will have some tuneable parameters that affect its performance, default values for these parameters were used during comparison. For succeeding comparisons, we use execution time and recall as our metrics. Execution time is divided into the 3 major steps, namely, detection, description and matching for more intuitive analysis. It should be noted that the actual execution time or execution cycles of the final hardware implementation will be different from what we obtain from software simulations using

OpenCV, but the relative values for the different algorithms may still be indicative of their relative performance.

The succeeding figures show the relative execution times of the 3 algorithms. All values are normalized with respect to SIFT, which is the slowest of the three. It can be seen from Fig. 3.1 that SURF is ~4x faster than SIFT in detection, and ~1.75x faster in description. ORB, on the other hand is ~20x faster than SIFT in both detection and description. Taking the total execution time (Fig. 3.1d), including time for matching, SURF is ~2x faster than SIFT while ORB is ~20x faster than SIFT, despite its higher matching time compared to both SIFT and SURF (Fig. 3.1c).

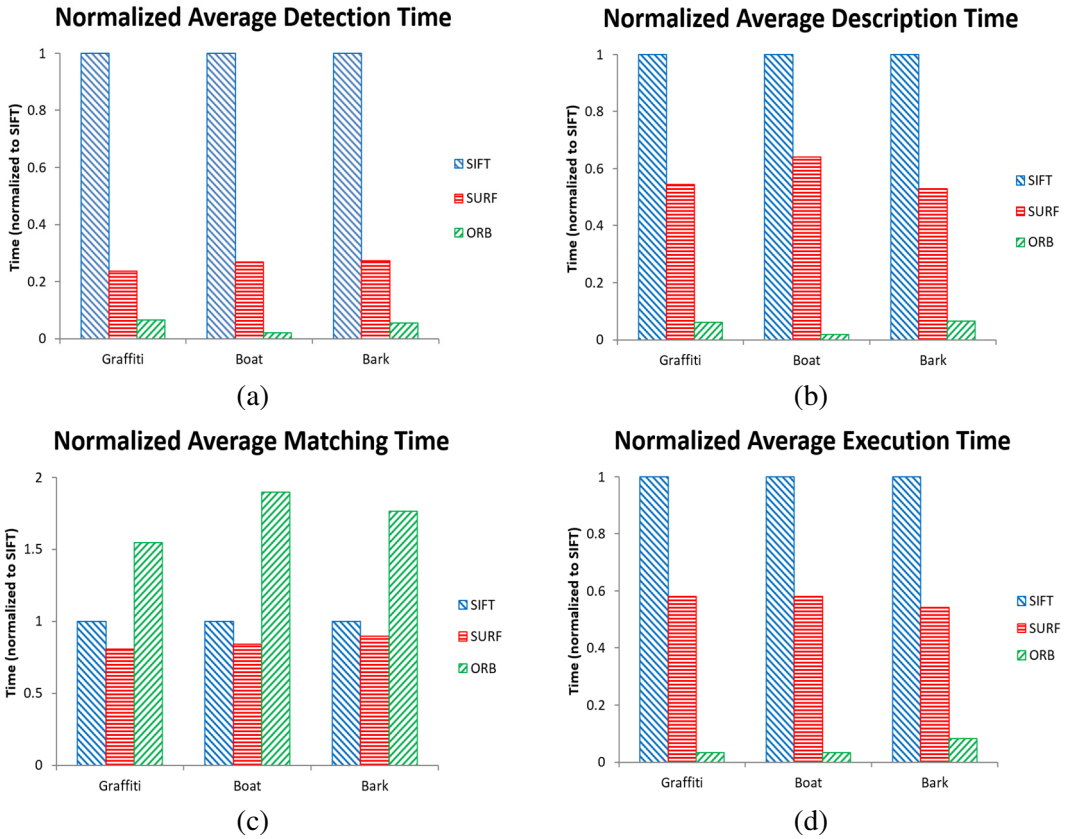


Fig. 3.1 Normalized execution times of SIFT, SURF and ORB: (a) average detection time; (b) average description time; (c) average matching time; and (d) total execution time.

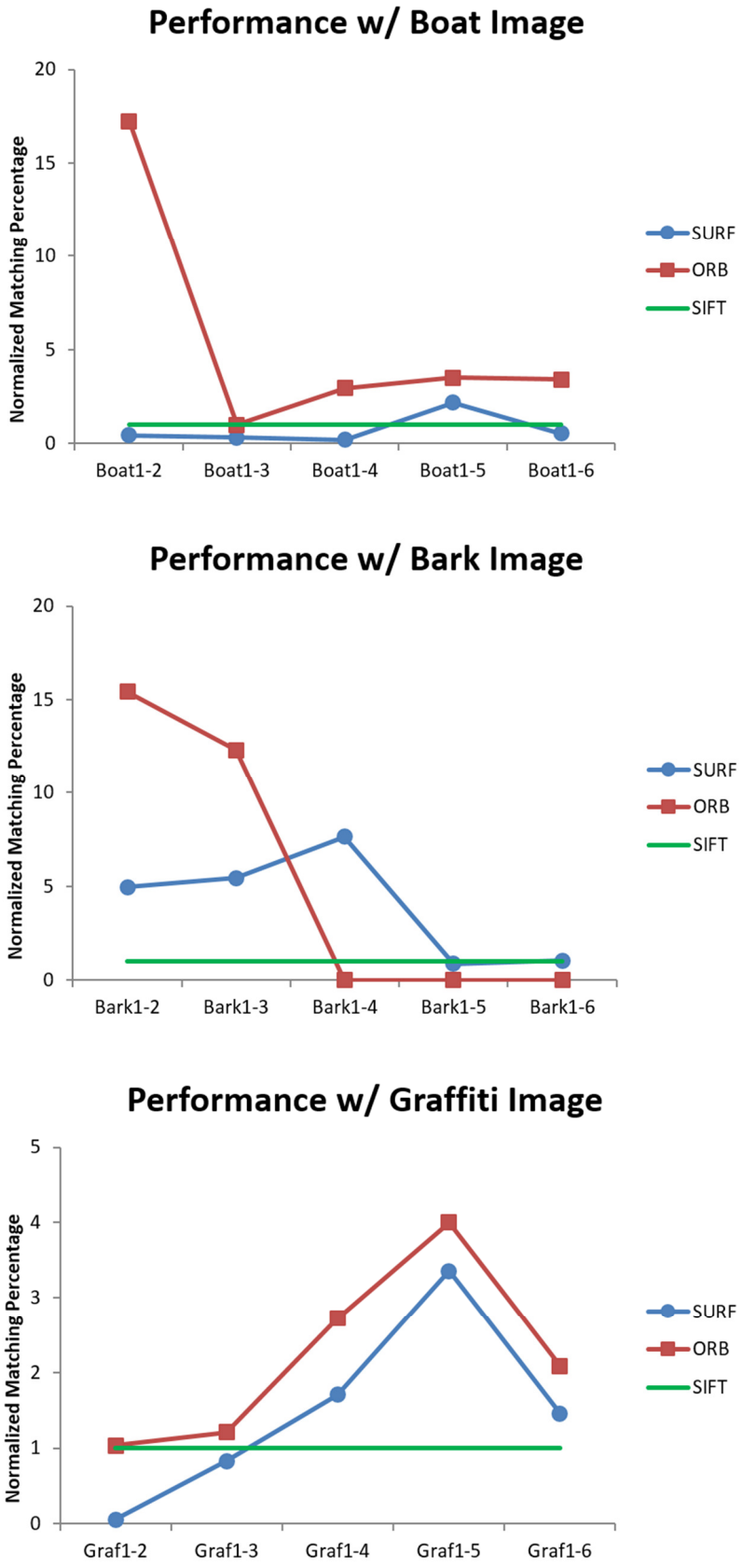


Fig. 3.2. Normalized matching performance of SIFT, SURF and ORB using recall metric, using different image variations: (a) boat image, for zoom and rotation; (b) bark image, for zoom and rotation; and (c) graffiti image, for viewpoint vairiation.

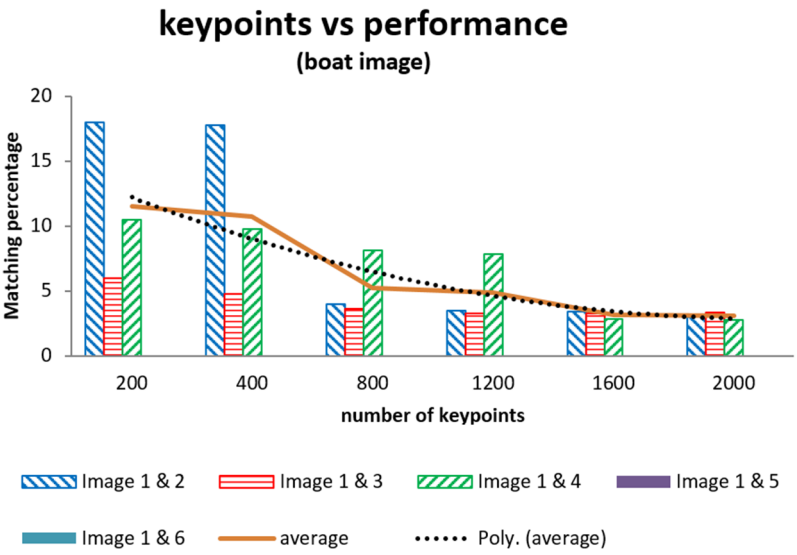
Matching performance is evaluated using recall, which is simply the percentage of correct matches (as discussed in Chapter 2), as metric. Fig. 3.2 shows the results for 3 different image sets: boat and bark for zoom with rotation invariance, and graffiti for viewpoint invariance. All values are normalized with respect to the SIFT performance. The x-axis represents increasing degree of variation (e.g., for rotation, higher degrees of rotation from the base image 1). In terms of invariance to zoom and rotation (Fig. 3.2a and Fig. 3.2b) , it can be seen from the figure that ORB outperforms SURF and SIFT for slight variations in rotation (i.e., for 1-2 comparison), but all 3 algorithms perform almost equally for higher degrees of rotation. With regards to viewpoint changes, SIFT and ORB perform equally better than SURF at lower degrees, while ORB and SURF outperform SIFT at higher degrees.

3.2. Tuneable Knobs in ORB

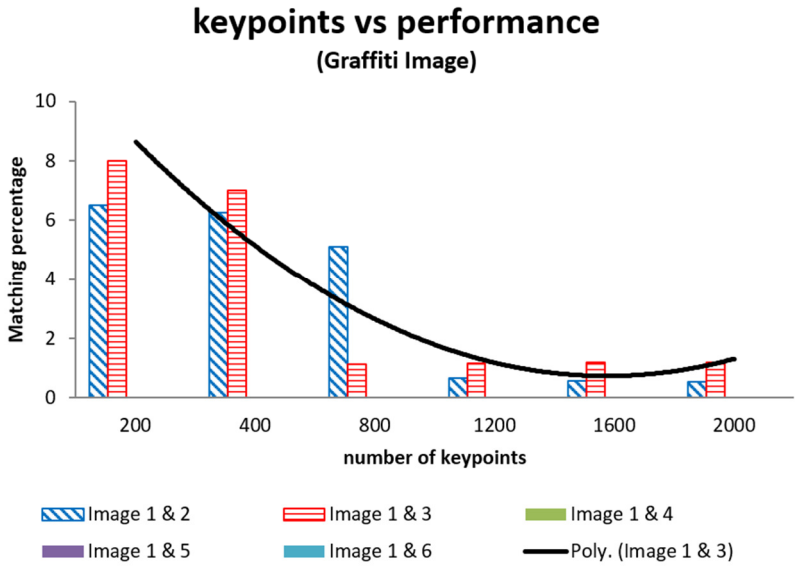
Having established ORB's superior speed and comparable performance with SIFT and SURF, tuneable knobs that would allow us to play with energy-quality trade-offs were identified. Several knobs are available both in the algorithm and in the hardware implementation. Investigation of each one of these knobs, for their suitability as energy-quality trade-off knobs, are presented in this section.

3.2.1. Number of keypoints

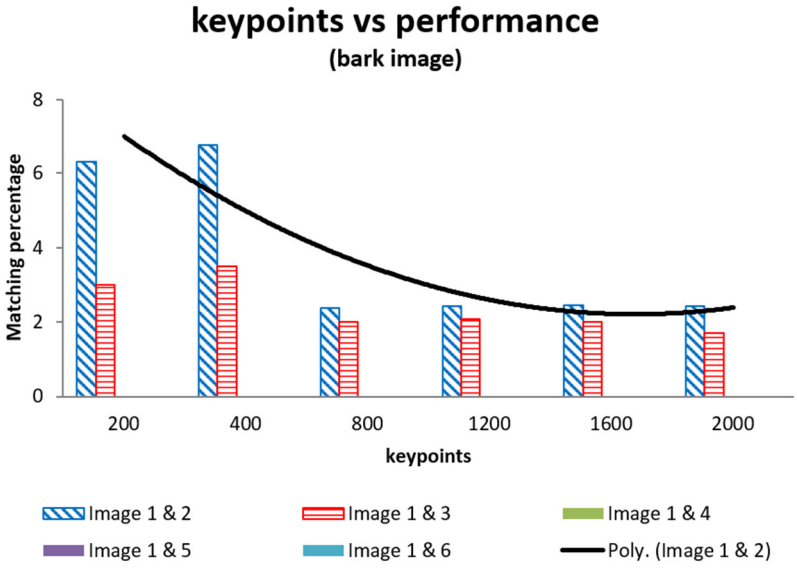
To reduce the number of keypoints in the database for matching, ORB implements ranking of the keypoints and retains only a fixed number of keypoints for description (and eventually matching). An obvious way to increase the number of good matches is by increasing the number of keypoints. However, increasing the number of keypoints comes at a penalty of larger memory requirement with more comparisons for ranking, and therefore, higher energy.



(a)



(b)



(c)

Fig. 3.3 Effect of number of keypoints on performance for (a) boat, (b) graffiti and (c) bark images

The default value of the number of retained keypoints is 400, and for our purpose, we call this knob *nfeat*. Fig. 3.3 and Fig. 3.5 show the effect of varying *nfeat*. The same 3 images (boat, graffiti and bark) were used for this analysis and for all succeeding analyses in this chapter. As can be seen consistently for all three images in Fig. 3.3, the performance in terms of matching percentage (percentage of good matches relative to declared matches) decreases quadratically with increasing number of retained keypoints, despite the increase in the number of good matches. This is best explained by the fact that ORB uses the brute force method for matching, thereby forcing a match for every retained keypoint (therefore increasing the divisor of performance metric). Indeed, in practical image matching, there will most likely be some keypoints that will not have a match from the keypoint database (i.e., occluded images). As such, a thresholding method for matching is proposed and will be discussed in more detail in the next chapter.

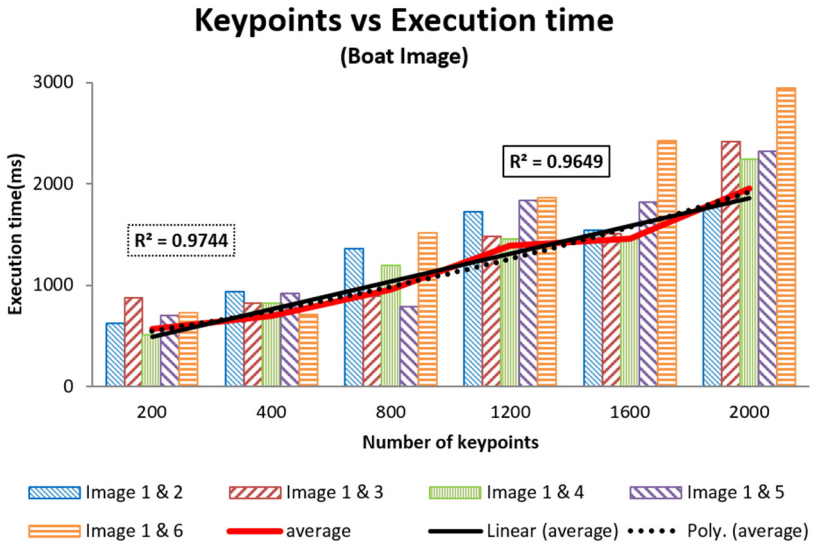
One thing to notice in Fig. 3.3 is that the last few image pairs (images 5 and 6 in Fig. 3.3a, and images 4-6 in Fig. 3.3b and Fig. 3.3c) have no matches. A closer look at these matches with the boat image (Fig. 3.4), shows that although there were a few correct matches, they were not enough to detect an object and therefore OpenCV erroneously reports a high value for number of correct matches. As a remedy, we set this value to zero instead, to indicate that the object was not detected. Fig. 3.4a shows the matching between boat image 1 and image 2, where almost all filtered matches are correct. On the other hand, Fig. 3.4b shows the matching between boat image 1 and image 6, where only a few of the filtered matches are correct, and therefore, the object is not detected after using PROSAC [52] or RANSAC [53]. Thus, although there are some correct matches (albeit very few), they are recorded as 0 since no object was detected.

(a)

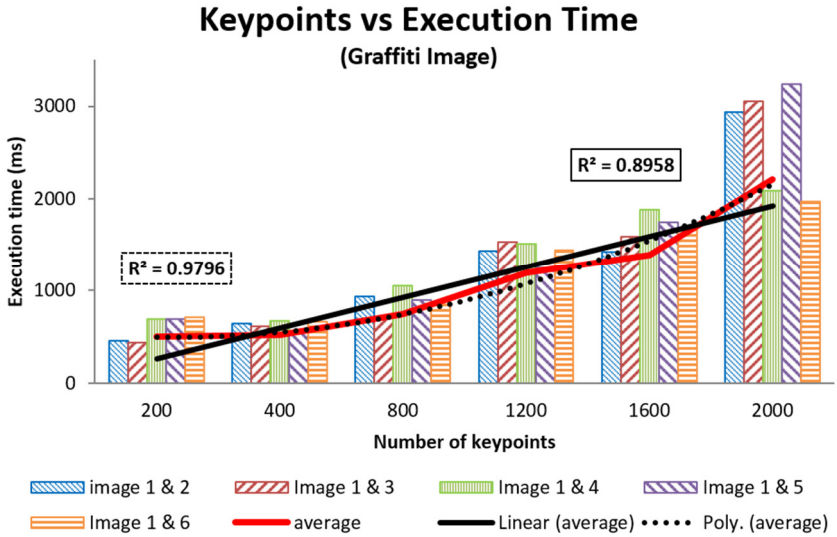
(b)

Fig. 3.4 Image matching in boat: (a) image 1 (left) & image 2 (right); and (b) image 1 (left) & image 6 (right). Lines connecting the two images are the declared matches.

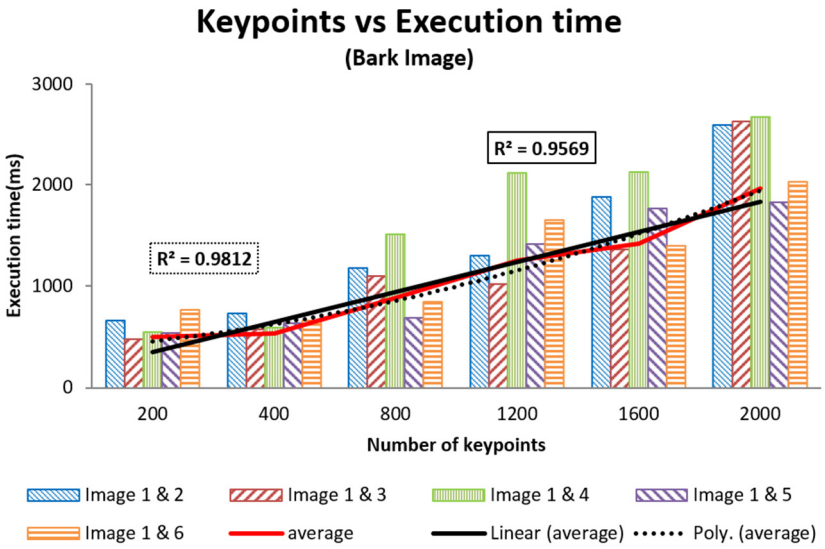
With respect to the performance (Fig. 3.5), the time taken for computation of the keypoints increases quadratically (as indicated by an R^2 value closer to 1 for poly compared to linear trendline) with the number of keypoints. So, care must be taken when one is planning to increase the number of keypoints. Thus, instead of increasing the number of keypoints to improve performance, we might also reduce the number keypoints to reduce energy consumption. From the plots, retaining 200 to 400 keypoints seems to be reasonable, with only a slight increase in energy and ~2x good matches.



(a)



(b)



(c)

Fig. 3.5 Total execution time of ORB for different images: (a) boat, (b) graffiti, (c) bark

3.2.2. Threshold

The number of keypoints is also affected by the threshold used to classify surrounding pixels as either bright, dark or grey. The higher the value of threshold, the lesser the number of keypoints detected. The default value is 20, and for our purpose, we call this knob *thresh*. Fig. 3.6 shows the effect of threshold on the number of keypoints. It can be seen from the figures that indeed, the number of keypoints decreases with increasing threshold value. The relationship between threshold and number of keypoints (Fig. 3.7) is a quadratic decay down to a threshold value less than 40 (indicated by the red trend line), after which, the number of keypoints decays exponentially. This shows that for this set of benchmark images, threshold values below 40 may not give enough keypoints, causing the matching to fail. Of course, this conclusion is image-dependent, and the reasonable threshold value may vary depending on the image contrast.

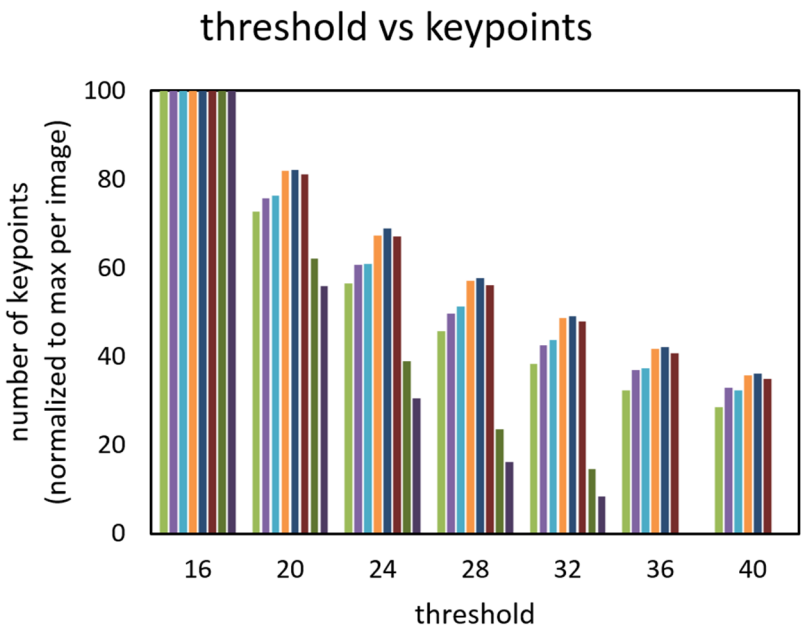


Fig. 3.6 Effect of threshold on number of detected keypoints

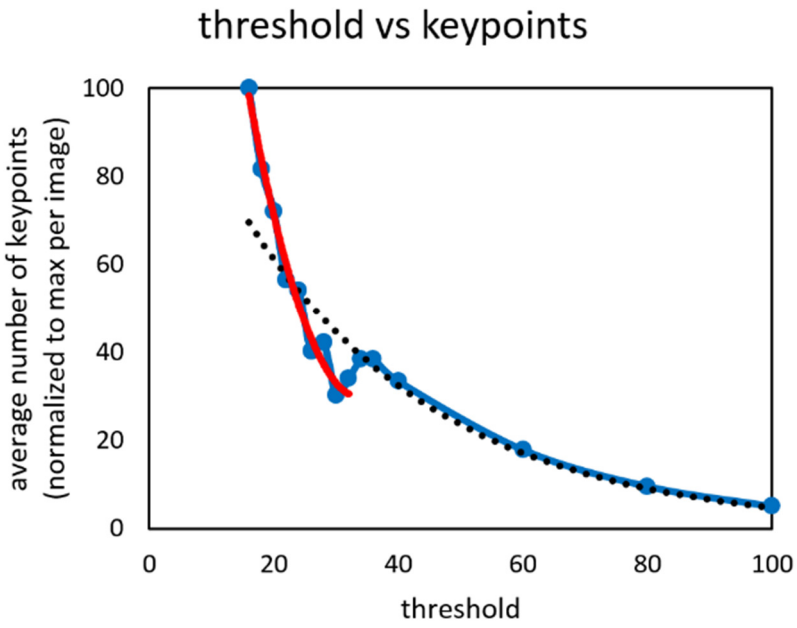


Fig. 3.7 Relationship between threshold and number of keypoints. For threshold <40, relationship is quadratic (red); for threshold >40, relationship is exponential.

Fig. 3.8 shows the effect of threshold on execution time. From the figure, a slight decrease in detection time with increasing threshold is observed, which can be attributed to the decrease in number of clustered keypoints, thereby reducing the time needed to do the non-maximal suppression. However, no apparent trend can be seen in terms of the effect of threshold on total execution time.

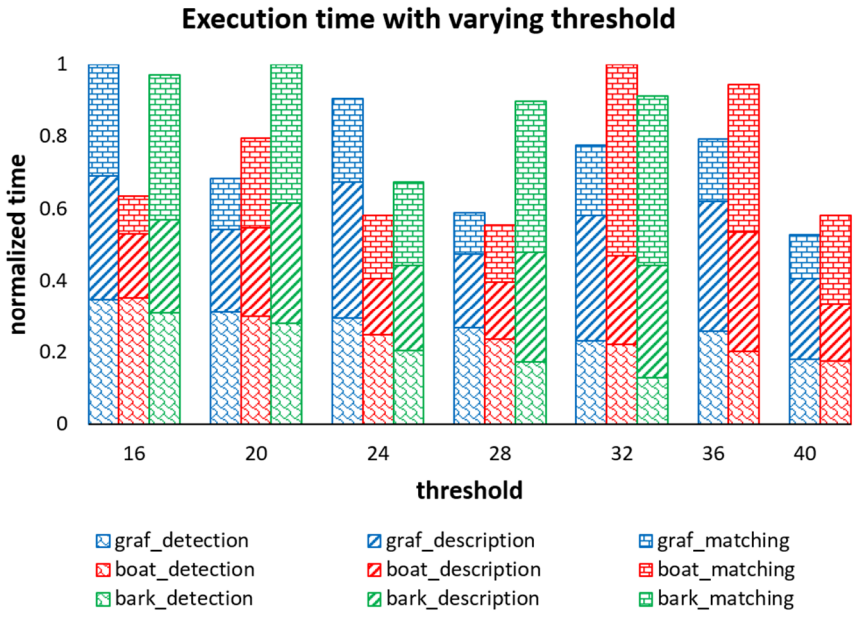


Fig. 3.8 Effect of threshold on execution time

3.2.3. Descriptor length

ORB uses 32 bytes for description. These bits are results of comparisons of pairs of pixel intensities, which are chosen based on their correlation (with the lower correlation preferred). The pairs are arranged in order of increasing correlation, with the least correlation pair corresponding to the MSB of the description vector. Thus, reducing the number of bits for the descriptor (keeping the lower correlations and discarding the higher ones) could have less impact on the accuracy of the descriptor for matching. For our purpose, we call this knob *nlength*, and we considered 2 possible values: 32B and 16B. This knob affects only the description, and is proportional to the size of the memory used for the comparison database as well as the computation effort to do brute force matching. Similarly, since description is done by bitwise comparison of the pixel pairs, reducing the length of the descriptor reduces the number of comparisons needed, and therefore, the time to do description, or the number of comparators needed if done in parallel.

Fig. 3.9 shows the effect of descriptor length on the description time (detection time is unaffected by this knob). Indeed, we can see that the description time for the 16B is always lower than that of the 32B, and the average description time for 32B length is 1.34x that of the 16B length. This shows great potential to reduce energy consumption of the ORB feature extraction accelerator. In terms of performance, on the other hand, no apparent trend can be found, as shown in Fig. 3.10. For the graffiti image pairs, the 16B length shows higher number of correct matches compared to the 32B descriptor length. This could be due to the dissimilarity in the keypoints within the image, therefore the most uncorrelated half of the descriptor length is enough to declare a match. The opposite, however, is seen in the boat image pairs, while the bark image pairs shows higher for some pairs and lower for others. For these image pairs, reducing the descriptor length to 16B from 32B could cause mismatches, thereby reducing the

number of correct matches. In terms of averages, however, the average number of correct matches for the 32B descriptor length is slightly higher (additional 3 correct matches on average) than that of the 16B.

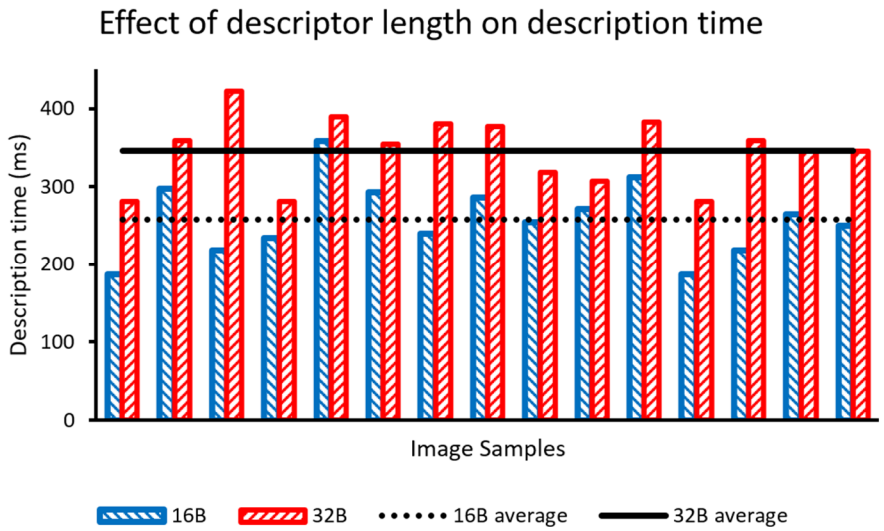


Fig. 3.9 Description time with varying descriptor length

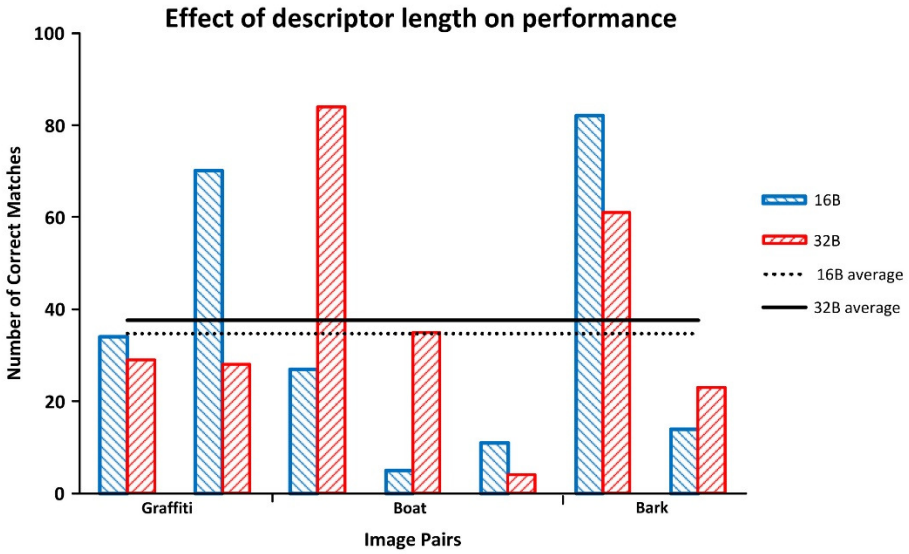


Fig. 3.10 ORB performance with varying descriptor length

3.2.4. Number of pyramid levels

Like SIFT, ORB addresses scale invariance by creating several scales of the image or pyramid levels, and detecting keypoints on those pyramids. Each of these pyramid levels is smoothened by a Gaussian filter to reduce the response along its

edges. Fig. 3.11 shows the number of good matches with respect to number of pyramid levels. It can be seen from the figure that, contrary to what we would expect, increasing the number of levels in fact reduces the number of good matches. This could be explained by the fact that the number of keypoints are almost *N*-tupled for *N* pyramid levels, but is eventually filtered into a fixed number of features, as dictated by *nfeat* (section 3.2.1).

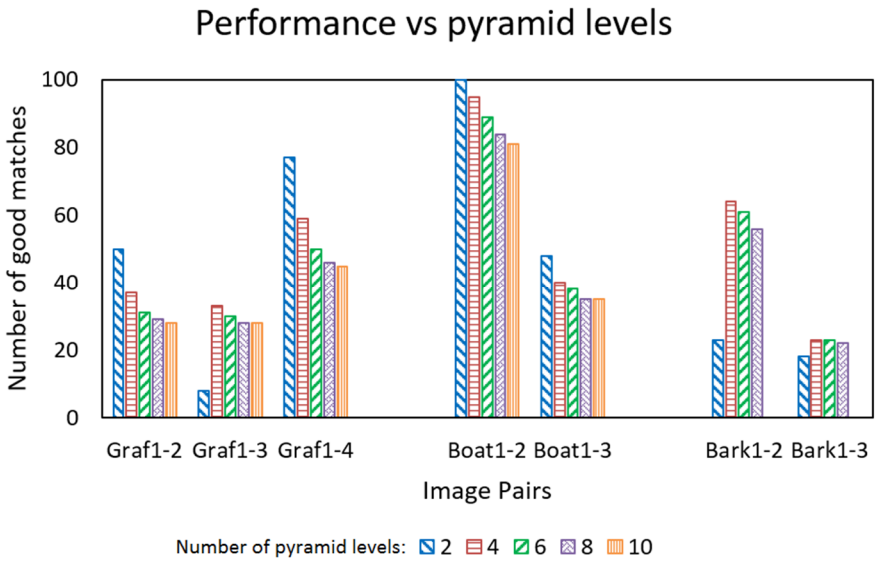
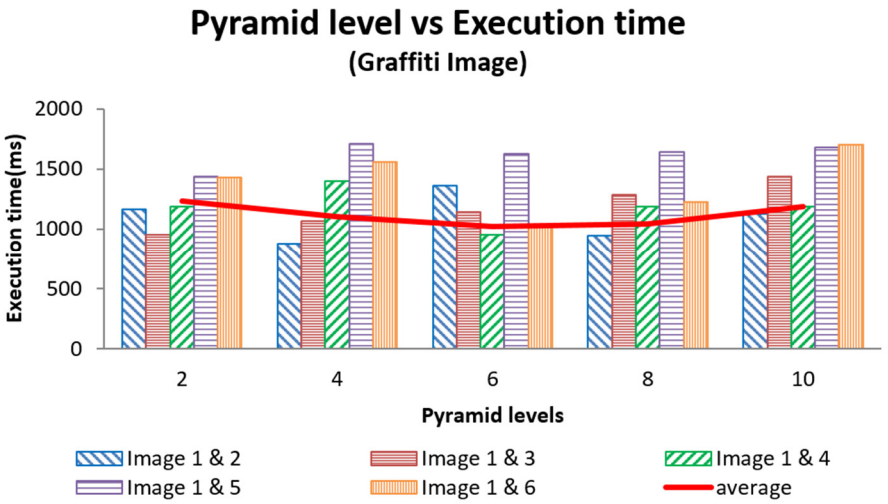
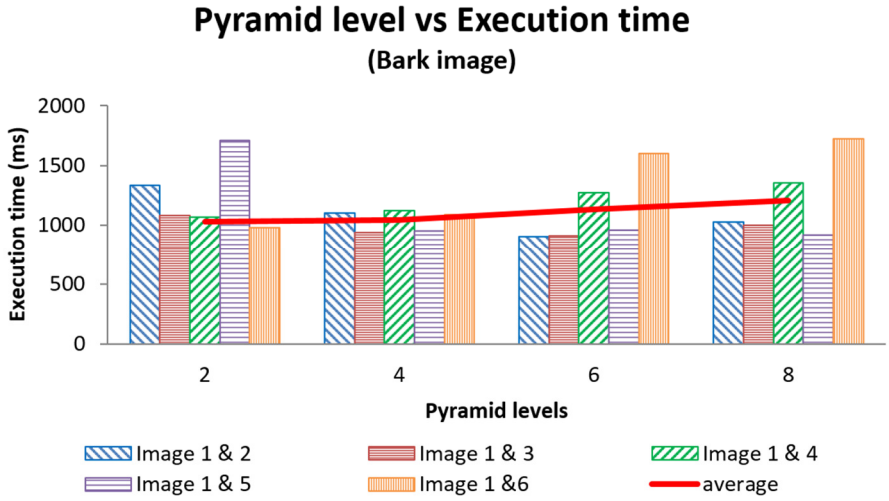


Fig. 3.11 ORB performance vs number of pyramid levels

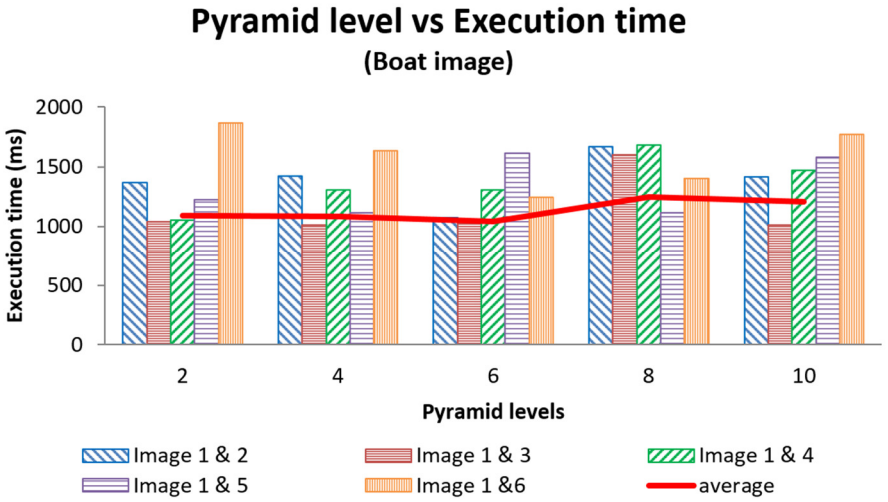
Looking at the execution time in Fig. 3.12, this trend in performance does not translate to any trend in execution time. Translating to energy, although execution time is almost constant, each additional pyramid is a new set of Gaussian filters and corresponding memory – therefore additional hardware resource and power, and thus higher energy. Thus, for our implementation, the pyramid levels was set to 1, without tuneability. It should be noted that in the current ORB as proposed in [19], analysis of the effects of the pyramid levels on the performance was not done in details. Adding octaves, similar to SIFT, and resolving keypoints across scales and octaves may need to be further explored. As such, until a revised algorithm for this is added onto ORB, having the additional pyramid levels may not be as useful as it was for SIFT.



(a)



(b)

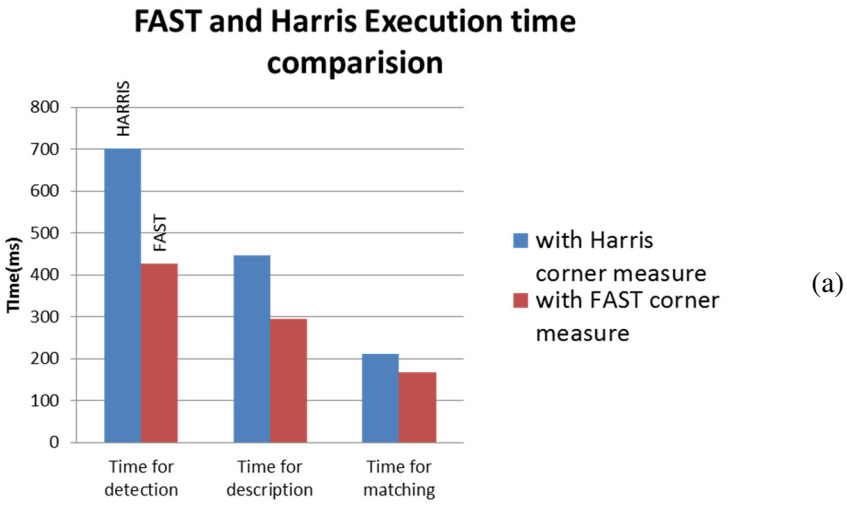


(c)

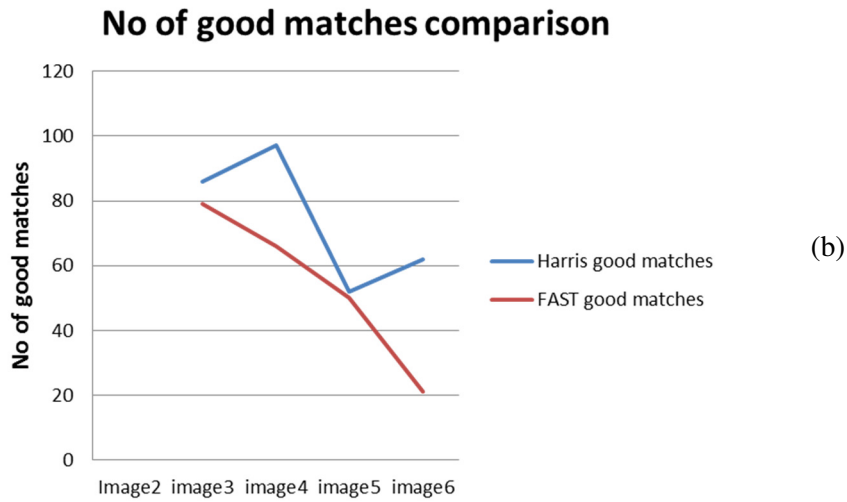
Fig. 3.12 Total Execution time vs number of pyramid levels for different images: (a) graffiti; (b) bark; and (c) boat images

3.2.5. Corner measure

Two corner measure algorithms offered with OpenCV were chosen from: Harris [27] and FAST [29]. This corner measure is used for NMS after the initial FAST feature detection. The computational complexity of the Harris corner measure is higher than FAST corner measure, as depicted by the higher execution time of the Harris corner measure (Fig. 3.13a). This, however, comes at a penalty in performance (Fig. 3.13b).



(a)



(b)

Fig. 3.13 FAST vs Harris corner measure in terms of (a) execution time and (b) number of good matches

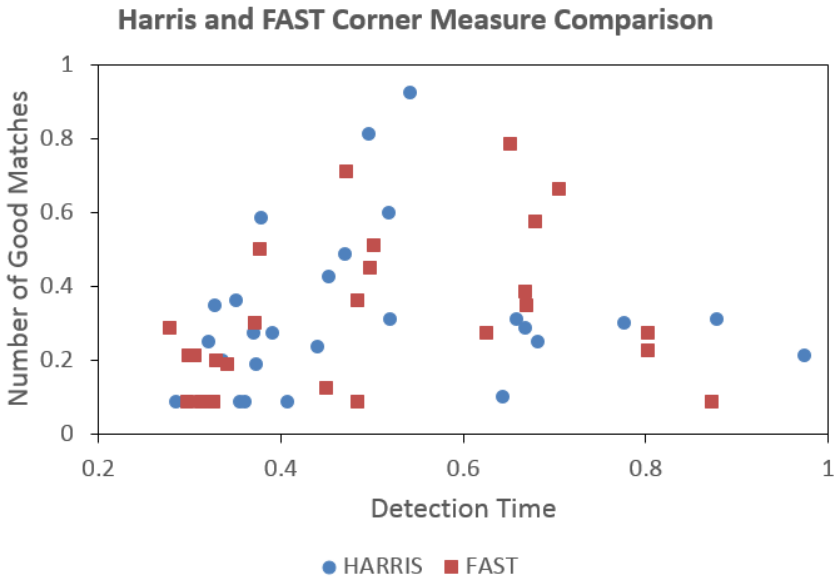


Fig. 3.14 Comparison of Harris and FAST speed and performance using several image pairs

Comparing good matches and execution times for different pairs of images, we can see in Fig. 3.14 that no apparent clustering is found, thereby leading us to conclude that both are comparable. Since FAST corner measure would be easier to implement, with less compute resource requirement, compared to Harris, all succeeding data will be based on the FAST corner measure, unless otherwise specified. We summarize all the tuneable parameters we have discussed, and their relative effects in Table 3.1 below.

Table 3.1 Parameters considered and their trade-off

Parameter	Definition	Range of values (default in bold)	Effect
nfeat	no. of retained keypoints	200, **400**, 800, 1200, 1600, 2000	~3x increase in execution time and performance from 400 to 2000 keypoints
levels	no. of pyramid levels	2, 4, 6, **8**, 10	~20% decrease in performance from 2 to 10 levels
nlength	descriptor length (bytes)	16, **32**	1.3x higher description time from 16 to 32
thresh	threshold	**20**, 30, 40	slight decrease in detection time and quadric decrease in detection time with decrease in threshold

3.3 Hardware Model

To approximate the effect of simplifications or approximations done for the hardware implementation, a Matlab model of the ORB hardware was implemented. Detection is done using string pattern comparison, as discussed in [46]. This design presents a simple implementation requiring only a 32-bit register, 16-bit adders, comparators and a simple FSM. The threshold knob *thresh* plays a role in this stage, as an input to the comparators to determine whether a pixel is to be labelled as dark, bright or grey. The lower the value of *thresh*, the more keypoints there will be. It should be noted at this point that one issue introduced by the detector is the repetitive access to same pixels, as each pixel will be needed by 16 different possible keypoints. Aside from these, pixels will also have to be re-accessed for description, in case of a keypoint. Details of the cache design considerations will be covered later in this section.

A non-maximal suppression (NMS) block processes all identified keypoints from FAST detection. This process removes redundant points, and retains the one that is the most representative of the corner. As was shown in the previous section, FAST corner measure is faster than Harris corner, although the later may have more correct matches. For the hardware implementation, the FAST corner was chosen, reusing the value measured from detection. A keypoint is considered a corner if it has the highest corner measure within its *nxn* neighbourhood, where *n* could be 3 or 5. For simpler discussion, we refer to them here as NMS-n. In the OpenCV implementation of the ORB algorithm, they used NMS-3. It can be seen in Fig. 3.15 that the number of keypoints is reduced, while keeping the white spot feature of the butterfly. As point of comparison, we show the number of keypoints of the base image (image_1) of all the benchmark images in Table 3.2. We include the resulting number of filtered keypoints using NMS-3 and NMS-5. Comparing the data from Table 3.2, NMS-3 reduces the number of keypoints to somewhere 25.49% to 38.96%, while NMS-5 reduces the number of keypoints to 21.55% to 34.27%.

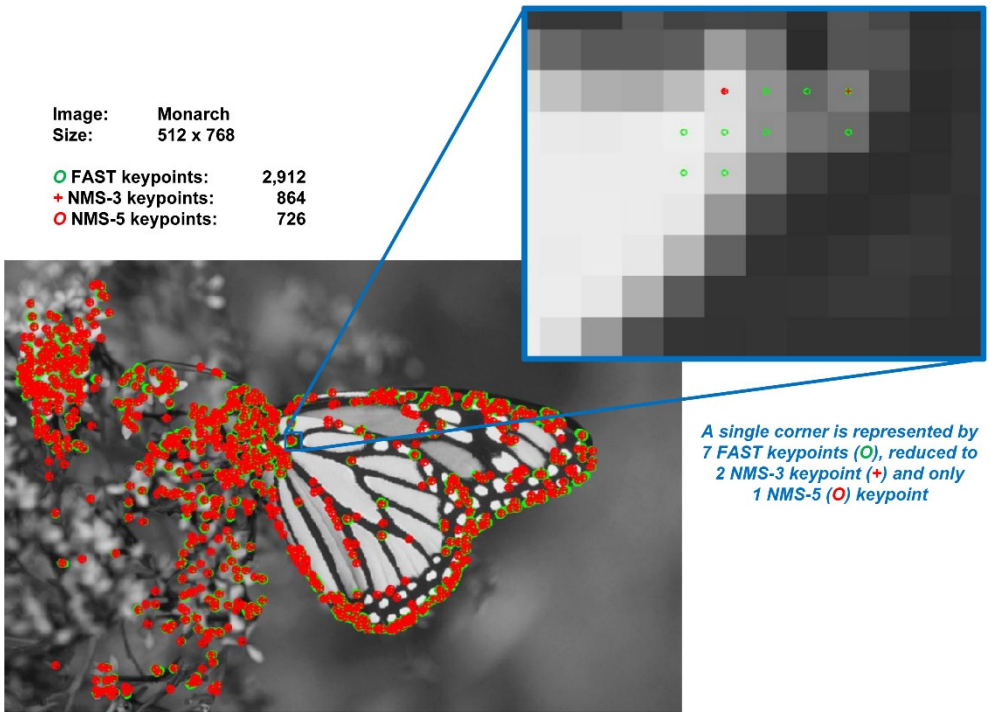


Fig. 3.15 Comparison of NMS-3 and NMS-5. NMS-5 reduces the number of keypoints while keeping the relevant feature.

Table 3.2 Number of keypoints with NMS-3 and NMS-5

Image	Size	# FAST keypoints	#keypoints after NMS-3	#keypoints after NMS-5
Boat	680 x 850	21128	6510	5034
Cars	600 x 900	5675	2146	1642
UCB	600 x 800	14941	5802	4695
Trees	700 x 1000	42628	13620	10362
Bike	480 x 640	3624	1412	1242
Graffiti	640 x 800	3503	893	755
Linus	640 x 480	992	384	326
Monarch	512 x 768	2912	864	726
Monarch2	480 x 640	2641	732	617

The difference in NMS-3 and NMS-5 filtered keypoints is further illustrated in Fig. 3.16. The pixel values of the inset image on the upper right corner of Fig. 3.15 is shown in the left image of Fig. 3.16. The identified FAST keypoints are highlighted in red. The corresponding scores of the pixels are shown on the right image of Fig. 3.16, where only the keypoints have non-zero scores. For NMS-3, the keypoint with score of 1366 (highest in the area) invalidates all adjacent keypoints. The keypoint with score of 495 do not have adjacent neighbours that are keypoints and therefore remains a valid keypoint. For the case of the keypoint with score of 1208 (lower left), although it is not

within the close neighbour of 1366, it is still invalidated because it has adjacent neighbours that have higher scores. Thus, for NMS-3, both keypoints with scores 1366 and 495 are passed on as valid keypoints. For the NMS-5, all keypoints within the 5x5 neighbourhood of the keypoint with a score of 1366 are invalidated (including that with a score of 1208). Although the keypoint with score of 495 (upper right) is not within this neighbourhood, it still becomes invalidated because there are keypoints within its 5x5 neighbourhood that have higher scores. Thus, for NMS-5, only the keypoint with score of 1366 is passed on as a valid keypoint. Indeed, this is what we should expect, since this is one corner and therefore should only have one representative keypoint.

51	50	50	51	51	49	49	49	50	47	45	80	96	50	48	46	47
54	54	52	52	52	52	53	54	54	47	47	57	80	57	47	48	49
55	56	64	59	59	55	56	85	67	71	70	48	85	72	45	46	49
67	108	160	132	135	103	89	89	93	157	117	44	84	83	47	49	52
141	194	230	224	224	193	174	171	188	223	146	104	125	72	44	48	50
217	229	233	237	237	233	232	235	238	225	143	117	108	51	45	52	53
234	235	235	236	236	236	235	235	235	201	113	62	49	45	47	52	50
234	234	236	236	236	236	236	235	230	152	68	47	45	48	53	50	48
233	234	236	236	236	234	236	231	183	97	50	48	53	59	58	50	49
233	235	236	236	236	236	230	176	89	52	52	53	57	58	51	49	51
233	234	235	235	237	221	153	80	52	51	52	50	50	50	48	48	48
233	233	233	224	189	126	65	49	50	50	50	48	49	50	50	48	49
232	234	218	150	80	55	50	50	50	49	48	46	46	48	48	47	49

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1366	630	0	495	0	0	0	0
0	0	0	0	0	0	0	0	1243	1361	582	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1208	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 3.16 NMS-3 and NMS-5 illustration. Pixel values (left) of a corner from Monarch image results to corresponding scores (left). Identified FAST keypoints are highlighted in red.

After NMS, the filtered keypoints within a frame are ranked. The ranking is based on the corner measure used in the NMS block, therefore no additional compute resource is needed for the corner measure. The *nfeat* parameter (default value of 400) indicates how many features or keypoints to retain per frame. To reduce extra logic for this implementation, we use only 1 bit for *nfeat* to choose between 200 or 400 keypoints (‘0’ = 200; ‘1’ = 400). The insertion sort algorithm was used for this implementation as it is well suited for data being added only at runtime. To reduce the number of cycles needed to rank the keypoints, the whole stack was divided into 10 bins such that the worst-case number of comparison is reduced to 50.

Description starts by taking the orientation or angle of rotation of the corner feature, based on the intensity centroid of the 15x15 patch around the keypoint. As

shown in the previous chapter, the computation resource needed would be multipliers (for m_{pq} computation), and a divider and trigonometric computation for atan2. In the OpenCV implementation of ORB, the trigonometric function itself was used. For the trigonometric computation, an 8-bin look-up table was used. Since the angle is used to compute for rotated *X* and *Y* values, the cosine and sine values of the angles were stored in the LUT instead of the atan2. Table 3.3 below shows the look-up table. Data is accessed by specifying the address (last column). As can be seen from the table, fixed-point representations were used to avoid the complexity of floating point arithmetic.

Table 3.3 Look-up table for atan2

|m01/m10|	|D|*2^4	sin	cos	cos*2^7	sin*2^7	Addr
0	0	0	1	128	0	0
0.2124456	3	0.20781	0.97817	125	27	1
0.4449743	7	0.40654	0.91363	116	54	2
0.726056	11	0.58753	0.8092	103	75	3
1.1096644	17	0.74286	0.66945	85	95	4
1.7299292	27	0.86576	0.50046	64	110	5
3.0710252	49	0.95086	0.30962	39	121	6
9.4468186	151	0.99444	0.10527	13	127	7

Aside from Table 3.3, another LUT for the 256-pairs of pixels to be compared is also included. Fig. 3.17 shows these points, with the keypoint as center, and within a 25x25 patch. For each pair, the pixel 'O' is compared with pixel '◊'. The 256-bit description is a concatenation of all O-◊ comparisons (1 if greater, 0 otherwise).

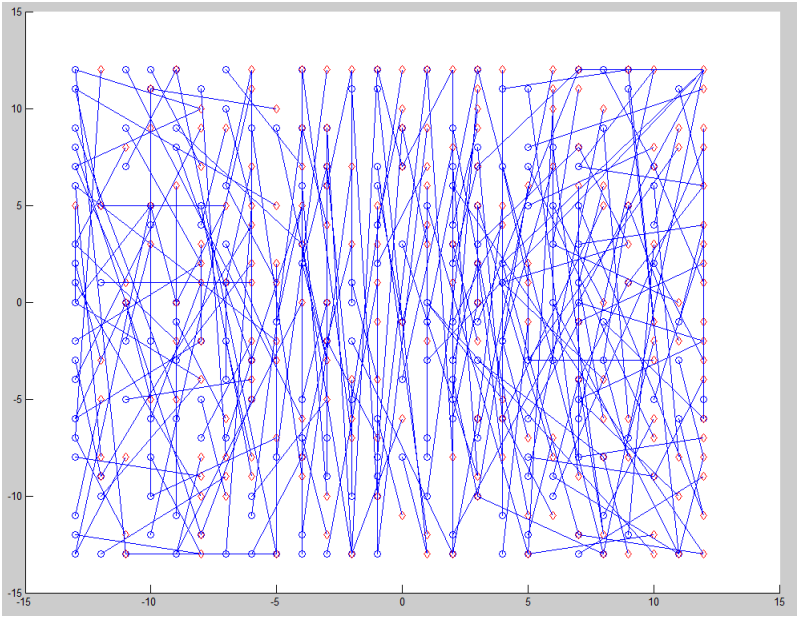


Fig. 3.17 Representation of LUT for 256-pair pixels. The 25x25 patch is centered on the keypoint at (0,0). The comparison between O and ◊ for each of the 256 lines dictate the bit value for description.

Aside from the comparison, the Descriptor block also needs to ensure that cache accesses are from different banks. This is because the cache banks are designed to have only one read port, despite the cache having two read ports (which means that each read data must come from separate banks). Since the pixel pairs that need to be accessed are non-deterministic (unlike in the Detector and Orientation and blocks), there is no telling whether the two pixels are from the same bank. In such cases where they are from the same bank, they have to be accessed at separate read cycles, thereby further increasing the time needed to finish the description time. Since each access is 7-pixels wide, determining which particular pixel to use for comparison also has to be performed. Due to limitation in number of pads, the descriptor vectors are outputted 4 bits at a time.

Another possible contention is the access to KEYPTS, where the keypoints for description are stored. The Ranking block accesses KEYPTS both for reading (to compare and perform ranking) and writing (to store newly processed keypoints). The Orientation block, on the other hand, accesses KEYPTS to read the keypoint

coordinates, X and Y. Thus, a KCtrl block was added to facilitate arbitration between these accesses. Priority is given to the Ranking block as it is the more critical block in terms of timing. Handshaking between the Orientation and Descriptor blocks are handled within the blocks themselves and no further arbitration is needed.

It should be noted that since we can only keep a patch of the image (equal to the size of the cache), we need to compute for the orientation and descriptor vector of the keypoints before we can replace the patch. Thus, we end up describing a lot more than the number of keypoints specified by *nfeat.* From simulations using the benchmark images, we get 5x more keypoints.

3.4. Summary

In this chapter, SIFT, SURF and ORB feature extraction algorithms were compared using OpenCV. It was shown that SURF is 2x faster than SIFT while ORB is 20x faster that SIFT, with all three having comparable performance in terms of matching percentage. This justifies the choice for using ORB as the base feature extraction algorithm. To allow for energy-quality scalability, several knobs were examined in terms of their effectivity in reducing energy without sacrificing too much on performance. Three knobs were eventually identified: the number of retained keypoints, *nfeat*, the length of descriptor, *nlength*, and detection threshold, *thresh*.

In terms of approximations or simplifications with regards to hardware implementation of ORB, doing NMS within a 5x5 neighbourhood (NMS-5) was shown to be better than NMS-3 (NMS within a 3x3 neighbourhood) by effectively reducing the number of keypoint by ~4% without losing the feature. In terms of orientation computation, atan2 is approximated using an 8-bin LUT for angle computation.

Chapter 4

EQSCALE Silicon Implementation and Results

The hardware model for the ORB accelerator was presented in the previous chapter. In this chapter, we will cover more details of the hardware implementation, including other issues and design considerations with regards to parallelism and cache architecture. Details of the RTL design are presented in Section 4.1. Section 4.2 covers the RTL simulation and measurement with the tuneable knobs identified in the previous chapter. Energy-quality scalability using the tuneable knobs, is shown using our 40nm CMOS test chip, which we call EQSCALE. This is presented in Section 4.3. To improve the cycle timing of EQSCALE, a second version was designed, increasing CACHE capacity by 3x. The results and comparison of the 2 versions are presented in Section 4.4. Finally, Section 4.5 covers further optimizations that can be done on the feature extraction accelerator to improve in area and energy efficiency.

4.1 RTL Design

Fig. 4.1 shows the proposed architecture of the ORB accelerator. The CACHE and KEYPTS blocks are latch-based memories for low-voltage operation. The CACHE stores a patch of the image, while KEYPTS stores the coordinates and scores of ranked keypoints. The CORE comprises five pipelined blocks: Detector, NMS, Ranking, Orientation and Descriptor.

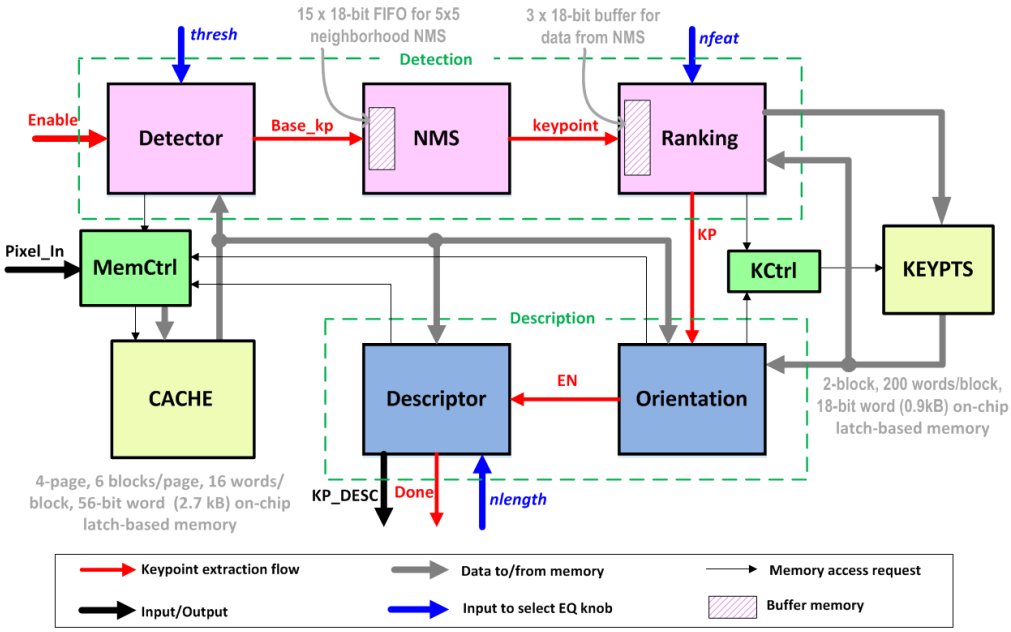


Fig. 4.1 EQSCALE architecture comprises a 2.7kB latch-based memory (CACHE) and a 0.9kB latch-based KEYPTS memory. Input rate is 1 pixel per cycle, with 8 bits per pixel. Knobs to alter the EQ trade-off are the threshold, *thresh*, number of retained features, *nfeat*, and length of descriptor, *nlength*.

4.1.1 CACHE and KEYPTS

CACHE and KEYPTS are latch-based memory using the standard cell memory (SCM) from [54], [55]. KEYPTS, on the other hand, is designed to have 18-bit words, arranged in 2 banks with 200 words/bank (~0.9kB). The simplified schematic of the SCM is shown in Fig. 4.2. For write operation (highlighted in red), address decoding (through WAD) is done in half a cycle and then the latches become transparent, taking *DataIn* as input. Data from the output of the latches go straight to read multiplexers (highlighted in blue). Read addresses are clocked in through registers and are used to decode (through RAD) the select signals for the read multiplexers.

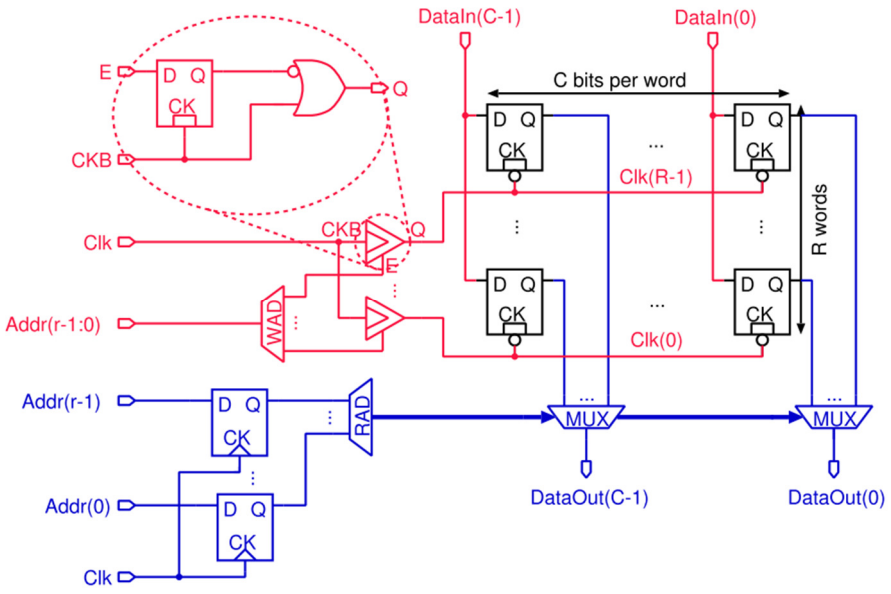


Fig. 4.2 Standard Cell Memory Schematic. Items in red highlight the write circuitry while items in blue show the read circuitry.

One energy consumption component not typically considered is the access to external memory. Access to external memory requires two orders more than on-chip cache access [56]. For this reason, we want to minimize or completely remove the external memory accesses. In EQSCALE implementation, we do this by performing description on the keypoints while they are still in the cache and before they are replaced. Thus, the need for an external frame buffer DRAM is eliminated. As mentioned in the previous chapter, this comes at a disadvantage of doing description on keypoints even if they are not among the top ranked keypoints. CACHE is designed to have a 7-pixel word (56 bits), arranged in 4 pages of 6 16-word banks/page. Thus, at one time, CACHE holds 64 rows of 42 pixels of the image (~2.7 kB). CACHE has 2 read ports, with 2 read addresses *RAdd1* and *RAdd2*. Each bank, however, has only one read port. Therefore, *RAdd1* and *RAdd2* should access different banks at a time. From Fig. 4.1, 3 different blocks request read access from CACHE through the MemCtrl: Detector, Orientation and Descriptor. An illustration of the re-use of CACHE data (detection then description) and accessed locations is shown in Fig. 4.3.

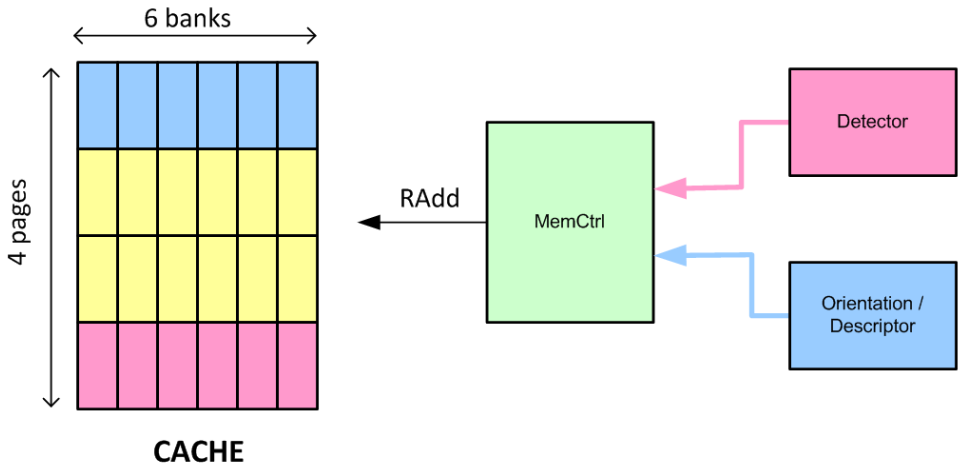


Fig. 4.3 CACHE re-use access illustration. Detector accesses a word within the pink shaded area of the CACHE while Orientation and Descriptor access a word within the blue-shaded area of the CACHE.

The Detector block accesses pixels as they are written onto the CACHE. As an illustrative example in Fig. 4.3, Detector accesses a pixel within the pink-shaded page of the CACHE. In case of a keypoint, Orientation and Descriptor will need to access pixels around the keypoint. In Fig. 4.3, for example, the keypoint will be somewhere in the blue-shaded page of CACHE. After filling the whole CACHE, the contents at the top will be replaced, and Detector will move on to process the newly refilled data. However, if Descriptor is not done with this area yet, it will issue a *STALL*, thereby halting the overwriting of data. Once Descriptor is done with the page, the operation can proceed. Although this results in extra logic (to determine when to stall and when to resume), it allows Detector then possibly Orientation and Descriptor to reuse the data currently held in CACHE, and therefore once the data has been replaced, there will be no need to re-access it (thereby eliminating the need for an external DRAM storage.

4.1.2 CORE Design

Detection process is done in the Detector, NMS and Ranking blocks, while description is done in the Orientation and Descriptor blocks. The Detector block receives the data (2 sets of 7 pixels per access) directly from the cache. Detection is done using string pattern comparison, as discussed by Park, et. al. [46]. This design

presents a simple implementation requiring only a 32-bit register, 16-bit adders, comparators and a simple FSM. To reduce the number of cache accesses (and therefore reduce access energy), detector implementation is done in parallel (7 parallel pattern detectors). The process starts by converting the pixels to bits, based on the threshold value. This is performed in the 2-stage pipelined Formatter block, which requires 7 cache accesses in 13 cycles. In Fig. 4.4, it is shown that detection of 7 pixels can be done with only 2-word accesses (middle). The 42-pixel row of an image patch in the CACHE, divided into 6 banks (1 word per row), of which 7 interest points in the middle are processed for detection, is shown (top). For each 2-word row access (banks 2 and 3), the pixels for the 16-pixel circle and its corresponding center is sent to the corresponding Det_unit block, which converts the 16-pixel circle to their corresponding bit strings.

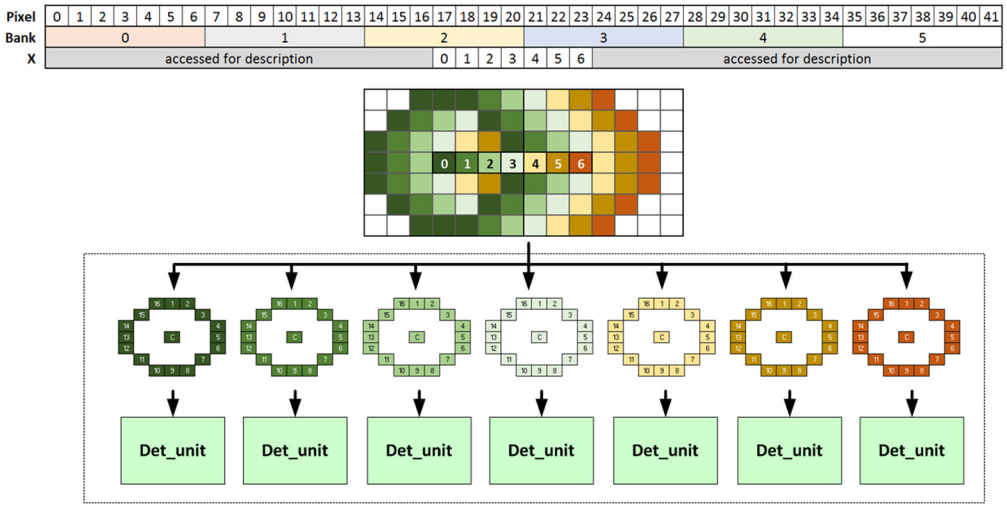


Fig. 4.4 Parallel detection of 7 pixels. Image window with 42 pixels per row is stored in CACHE, of which 7 pixels are keypoint-detected (top). For each access, pixels for each 16-pixel circle and the center interest point are sent to corresponding detector units (below).

The operation of the Det_unit is shown in Fig. 4.5. For each interest point, for every row access from CACHE (first access is the middle row), the interest point (labelled C in each of the 7 circles in Fig. 4.4) and the 2 pixels belonging to the 16-pixel circle are sent to the Det_unit (as A_i in the left circuit in Fig. 4.5). It should be

noted that for the case of the top and bottom rows, there are 3 pixels per circle. For these cases, the corresponding *RAdd1* and *RAdd2* values to CACHE are kept for 2 accesses, where for the second round, the output of the 2nd parallel structure is ignored. Each pixel is then converted into 2 bits, *text_bright_i* and *text_dark_i*. Aside from the bit strings for detection, the corresponding scores (*sum_bright* and *sum_dark*) are also needed in case the interest point is indeed a keypoint (top right). To ensure proper timing, a simple 3-state FSM generates the needed control signals (bottom right).

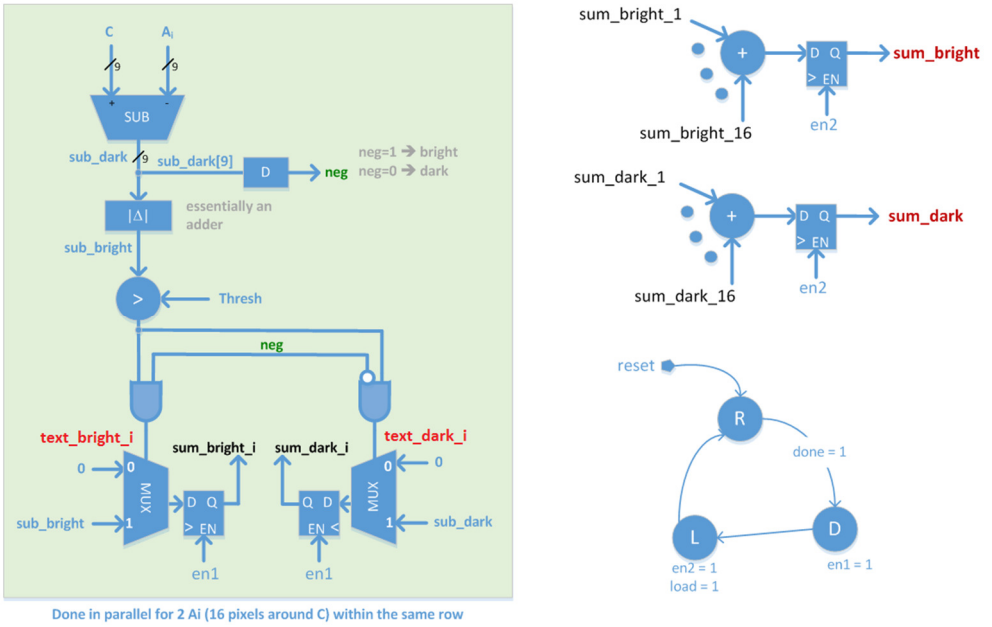


Fig. 4.5 Det_unit operation, to convert the pixels to their corresponding bit strings (text_bright and text_dark) for detection (left). Scores are also computed (top right). Generation of the control signals for handshake is also shown (right bottom).

After the Formatter block is the Rotator block, which is also implemented as 7 parallel units, one for each *text* output from Formatter. This block requires 3-6 cycles, depending on input pixel. Lastly, an 8-state SerialFSM block is added to give the output (valid/invalid) per pixel and necessary control signals. The simplified block diagram for the detector is shown in Fig. 4.6.

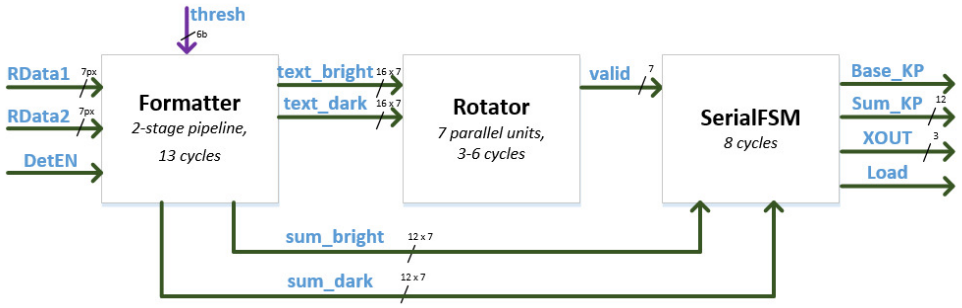


Fig. 4.6 Detector Block Diagram

As can be seen from the figure, the whole detection process takes a maximum of 28 cycles (and minimum of 25). Due to parallelism, however, this brings the detection time to 28/7 = 4 cycles per pixel. Synthesizing using a commercial 40nm CMOS technology shows an area estimate of 0.015 sq.mm and an estimated total power of 2.61 mW at 330MHz with 0.9V supply. The breakdown of power and area for each sub-block is shown in Table 4.1 below.

Table 4.1 Area and Power Estimates for Detector Block

Block	Leakage Power (uW)	Total Power (mW)	Area (sq. um)
Formatter	2.44	1.43	8053.72
Rotator	1.53	0.95	5894.94
SerialFSM	0.37	0.23	1276.96
DETECTOR	***4.35***	***2.61***	***15225.61***

The NMS block performs non-maximal suppression of the identified keypoints to remove redundant points, and retain the one that is the most representative of the corner. As discussed in the previous chapter, a 2D NMS for 3-neighborhood (NMS-5) was implemented. For this implementation, a 7x5 entry memory was used. An entry contains the relative *X*, relative *Y*, and corner measure (which we refer here as *sum*) for each keypoint. In case a pixel is not a keypoint, its corresponding entry is simply invalidated and not included in the comparison. The controller is an 8-state FSM, and at worst-case, it takes 28 cycles to perform NMS on one row of data. A row of data is essentially 7 pixels (as with the detector), although the number of keypoints per row would vary depending on the input image. Due to the image scanning pattern assumed

for the cache, a row of data would have a maximum of 3 keypoints. A new row is indicated by the assertion of an *EOR* signal. After receiving 5 of these signals (i.e. 5 rows of data from detector block), the keypoints in the middle row (if any) are compared with its neighbouring 5x5 region. After processing the middle row (row 3), the first row (row 1) is discarded and the same process is done when the next EOR signal is received. To ensure proper timing between the detector and NMS block, a separate array was used to store the keypoints (*X*, *Y* and *sum*) and *EOR count* (essentially a row indicator) from detector whenever a new keypoint is received while the NMS block is executing.

It should be noted, however, that the left (right) edge of the scanned image will not be compared with the left (right) column of the image, thereby possibly having more keypoints than when doing NMS in software. This limitation, being due to the finite cache memory. Simulations with benchmark images, however, show that this limitation adds only at most 10% of keypoints and does not affect the overall detection performance. Synthesizing the design using a commercial 40nm CMOS standard cell library gives a total area of 8898 um^2 and total estimated power is 1.57 mW at 330 MHz with 0.9V supply.

After the NMS block is the Ranking block, which simply ranks all keypoints within the frame. The ranking block uses the *sum* from NMS block for ranking. The *nfeat* parameter, which indicates the number of retained features or keypoints per frame, is used as input to this block. For this implementation, we use only 1 bit for *nfeat* to choose between 200 or 400 keypoints (‘0’ = 200; ‘1’ = 400).

The ranked keypoints (identified by the relative *X* and *Y* values), as well as their corresponding *sum* (corner measure) are stored in a separate memory, we refer here as KEYPTS. The RTL implementation uses pointer arrays (*next* and *prev*) to store and keep track of the rank of incoming keypoints. These are stored in a memory array in

the Ranking block to perform sorting. The insertion sort algorithm was used for this implementation as it is well suited for data being added only at runtime. As with any other sorting algorithm, the first *N* (where *N* is the number of keypoints to be retained) will be stored in KEYPTS and their relative ranking will be determined by the pointers in the Ranking block. To reduce the number of cycles needed to rank the keypoints thereafter, the whole stack was divided into 10 bin such that the worst-case number of comparison is reduced to 50. Index values of every bin's last sum is stored in a separate memory named as *checkpoint* array. Once a new keypoint sum is received, it is compared with existing sum values that are fetched from KEYPTS by going through the *checkpoint i*ndexes to determine the bin that the sum belongs to. Each sum value in that bin is then obtained from KEYPTS and compared against the input sum one at a time until the final rank is determined. For the first 400/200 values (nfeat 1/0), index of every sum is stored in the array pointers after ranking and once the array is full, the index of the sum with least value is discarded whenever an input sum is greater than the least sum value. To reduce the memory read access count, the value of the least sum is stored locally so that the input sum is discarded if its value is less than the least sum value.

Synthesizing the design using a commercial 40nm CMOS standard cell library gives a total area of 0.067 mm^2, of which 40% is from non-combinational cells. The total estimated power at 330 MHz with 0.9V supply 8.42 mW, most of which is from the registers.

As discussed in the last section of the previous chapter, the Orientation block uses LUTs to implement the angle computation. Synthesizing the design using a commercial 40nm CMOS standard cell library gives a total area of 8382 um^2 and the total estimated power at 330 MHz, 0.9V supply is 0.823 mW.

The Descriptor block, takes the input from the Orientation block to get data from CACHE and form the descriptor vector of a keypoint. The descriptor may be either 128 bits or 256 bits, depending on the *nlength* knob ('1' for 256 and '0' for 128). Description is done by simply comparing pairs of pixels within a 32x32 patch centered on the keypoint. For this implementation, the pixel pairs to be compared are already determined in the Orientation block, and therefore the Descriptor block only needs to access the specific pixels, do a comparison, and output either a 0 (if first pixel is lighter than the second) or 1 (otherwise). Aside from the comparison, the Descriptor block also needs to ensure that cache accesses are from different banks. This is because the cache banks are designed to have only one read port, despite the cache having two read ports (which means that each read data must come from separate banks). Since the pixel pairs that need to be accessed are non-deterministic (unlike in the Detector and Orientation and blocks), there is no telling whether the two pixels are from the same bank. In such cases where they are from the same bank, they have to be accessed at separate read cycles, thereby further increasing the time needed to finish the description time. Since each access is 7-pixels wide, determining which pixel to use for comparison also needs to be performed. Due to limitation in number of pads, the descriptor vectors are outputted 4 bits at a time.

RTL implementation of the Descriptor block use an 8-state FSM. One of the states takes care of the second read access in the case where both pixels are from the same cache bank. Another state takes care of determining which pixel from the 7-wide cache data is to be used for comparison. Comparison of the pixels is done in another state. The rest of the states take care of cache access and ensuring proper timing of signals. Synthesizing the design using a commercial 40nm CMOS standard cell library gives a total area of 1397 um^2, of which 70% is from register. The total estimated power at 330 MHz, 0.9V supply is 0.25 mW, most of which is due to register dynamic power.

To integrate all these blocks together, several blocks had to be added. First, since detection does not start with every first pixel (description will need a 32x32 patch around the pixel), a MemCtrl block is added to generate an enable signal (*DetEN*) to tell the Detector block to start. It is also in the MemCtrl block where address to cache is generated. Furthermore, to avoid contention in cache access from either Descriptor or Orientation blocks, the MemCtrl block also acts as an arbiter for cache access. Priority is given to the Descriptor block since keypoint description needs to be completed first before a new keypoint from the Orientation block can be handled.

Since both Detector and NMS require 28 cycles to process a row of 7 pixels, no further handshake is needed between them. This is not true, however, between the NMS and Ranking blocks. Since the NMS block could have a maximum of 3 keypoints per row, a 3-entry buffer (which we call KPBUF) was included to ensure that each keypoint is passed to the Ranking block only when the Ranking block is done processing the previous keypoint. Due to ranking process taking a longer time, it is still possible for the buffer to be filled before the Ranking block finishes processing. As such, a stall signal is generated within KPBUF and sent to MemCtrl to stall the detection process, and therefore generation of new keypoints. Another possible contention is the access to KEYPTS, where the keypoints for description are stored. The Ranking block accesses KEYPTS both for reading (to compare and perform ranking) and writing (to store newly processed keypoints). The Orientation block, on the other hand, accesses KEYPTS to read the keypoint coordinates, *X* and *Y*. Thus, a KCtrl block was added to facilitate arbitration between these accesses. Priority is given to the Ranking block as it is the more critical block in terms of timing. Handshaking between the Orientation and Descriptor blocks are handled within the blocks themselves and no further arbitration is needed.

It should be noted that since we can only keep a patch of the image (equal to the size of the cache), we need to compute for the orientation and descriptor vector of the

keypoints before we can replace the patch. Thus, we end up describing a lot more than the number of keypoints specified by *nfeat*. From simulations using the benchmark images, we determined we are getting 5x more keypoints.

Summary of the area and power estimates (at 330MHz, 0.9V supply) of CORE block is shown in Table 4.2. From the table, we can see that the Ranking block occupies 66% of the CORE area, and contributes 62% of its total power.

Table 4.2 Power and Area Estimates of CORE

Block	Leakage Power (uW)	Total Power (mW)	Area (sq. um)	PnR Area (sq. um)	% Area
WCtrl	0.128	0.098	526.73	816.56	0.50
MemCtrl	0.338	0.075	914.46	970.02	0.59
Detector	5.89	2.33	18440.68	25648.38	15.55
NMS	2.5	1.53	9172.62	9978.07	6.05
Ranking	20.7	8.50	72077.75	109200.8	66.22
KCtrl	0.196	0.13	701.72	1001.6	0.61
Orientation	3.15	0.84	8890.56	12621.95	7.65
Descriptor	0.379	0.22	1431.66	1887.13	1.14
Others	0.219	0.086	685.14	2791.35	1.69
CORE	**33.5**	**13.81**	**112841.3**	**164915.8**	**100**

The die photomicrograph, with the corresponding dimensions of the blocks (after 90% shrinkage) is shown in Fig. 4.7. The die is 850umx1850um prior to shrinkage (1.27 mm^2 after shrinkage), with an active area of 0.55 mm^2. It can be seen from the figure that the CORE area is ~33% of total active area (without pads), while the CACHE is just slightly smaller at ~30%.

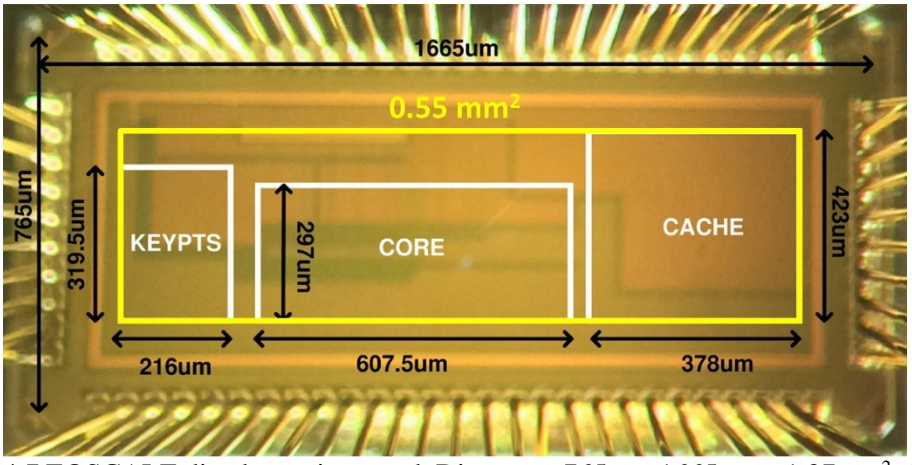
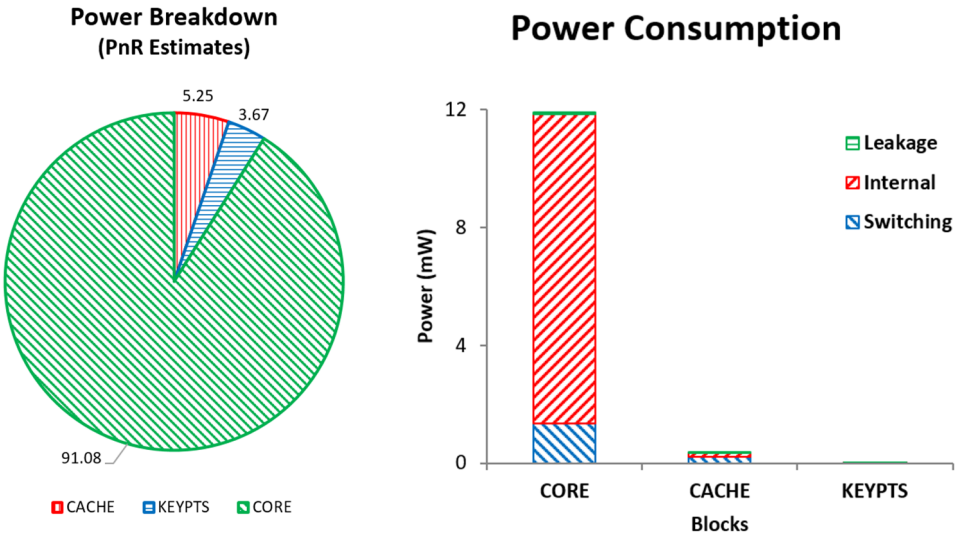


Fig. 4.7 EQSCALE die photomicrograph Die area = 765umx1665um = 1.27mm^2. Core area = 607.5umx297um = 0.18mm^2. Active area = 1304umx423um = 0.55mm^2

4.2 RTL Simulations with Tuneable Knobs

Simulations were done on the placed and routed design to get the power and energy estimates with the different knobs identified the previous chapter. Fig. 4.8 shows that CORE consumes more than 90% of the total power (and even higher when considering actual image vectors as inputs). From Fig. 4.9, PnR simulation results follow the same trend as that of the OpenCV simulation in terms of execution time. In terms of the descriptor length knob, *nlength*, Table 4.3 shows that the normalized execution time is considerably reduced compared to the OpenCV simulations, although both consistently show a decrease in energy with shorter descriptor length, as was predicted in Chapter 3. Fig. 4.10 shows the effect of the different knobs (*nfeat*, *nlength* and *thresh*) on the number of execution cycles. The default configuration (*thresh*=20, *nfeat*=1, and *nlength*=1) has the highest execution cycles. When increasing the *thresh* knob from 20 to 30 (30 to 40), the number of keypoints detected is reduced, causing the number of cycles required by execution to be reduced by 25% (31%). Changing from *nfeat* from 400 (N1) to 200 (N0) reduces execution cycles by 34%. This is due to the reduced comparisons required in the Ranking block. When reducing *nlength* from 32 bytes (L1) to 16 bytes (L0), the number of comparisons and cache accesses needed in the Descriptor block is reducing, thus, the execution time is cut down (by 34%).

Combining *nlength* (L0) with a *thresh* (higher value) gives the best result, with a 60% reduction in execution cycles.



(a) (b)

Fig. 4.8 Power consumption breakdown from PnR estimates (a) assuming arbitrary data and (b) using actual image as input

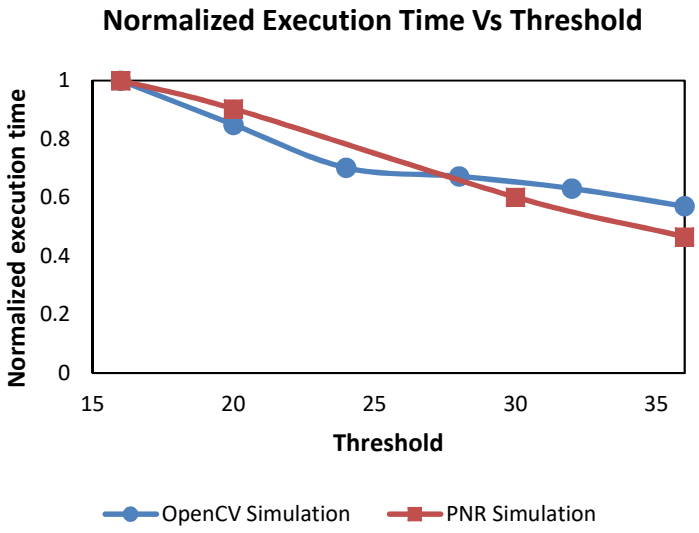


Fig. 4.9 Normalized Execution Time vs Threshold

Table 4.3 Normalized execution time vs descriptor length

nlength	Normalized Execution Time	
	PNR Simulation	OpenCV Simulation
0	0.623348055	0.825352113
1	1	1

Table 4.4 shows the power consumption of the three blocks with different knob settings. It is interesting to note that the total power consumption does not change much

with knob settings. This can be explained from what we saw in Fig. 4.8, where majority of the power consumption is from internal power of CORE. This means that the energy of the accelerator will be dictated by the corresponding execution time as dictated by the knob settings. As an illustration, taking a VGA image as input, the corresponding energy at 0.9V and 330MHz frequency is shown in Table 4.5. As was predicted in the previous chapter, the highest energy is when *thresh* is set at the lowest (most number of detected keypoints) value and both *nfeat* and *nlength* are set to 1 (400 retained keypoints and 32B descriptor length). From the table, we can also see the same decrease in energy when *thresh* is increased from 20 to 30 (30 to 40). With regards to descriptor length, *nlength*, the consumed energy is reduced by 24-33% when the descriptor length is reduced from 32 bytes to 16 bytes. Lastly, in terms of *nfeat*, energy consumption is reduced by upto 34% for the case of *thresh*=20.

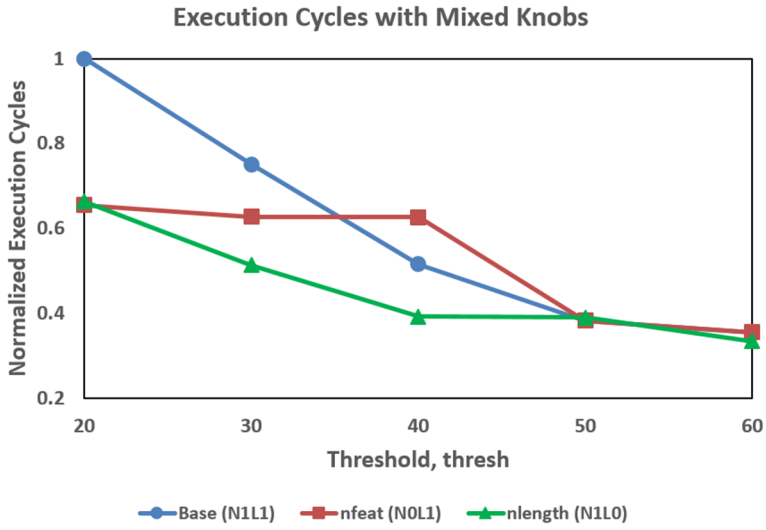


Fig. 4.10 Execution cycles for different knob settings

Table 4.4 Power consumption with different knob settings

Block	Total average power (mW)				
	nlength=0 thresh=20	**nlength=1 thresh=16**	**nlength=1 thresh=20**	**nlength=1 thresh=30**	**nlength=1 thresh=36**
CORE	12.00	12.00	12.00	12.00	12.00
CACHE	0.93	0.96	0.96	0.96	0.94
KEYPTS	0.56	0.56	0.56	0.56	0.56
Total	**13.50**	**13.52**	**13.52**	**13.52**	**13.51**

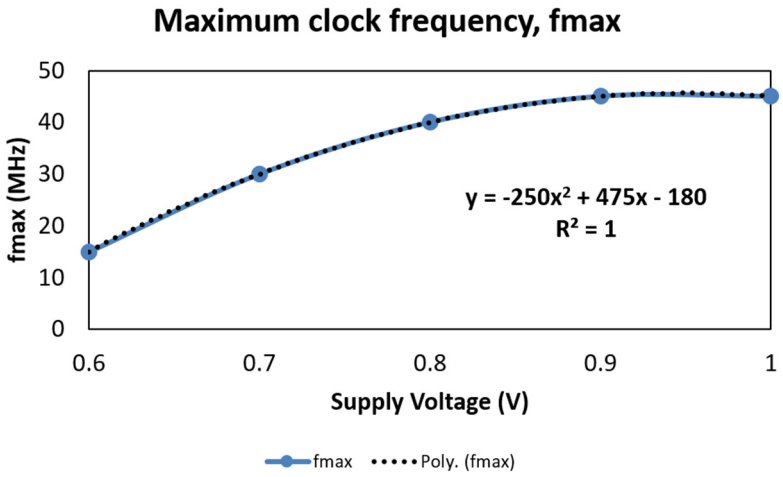
Table 4.5 Energy consumption at 0.9V, 330MHz with different knob settings

Knob Settings			Energy per frame (uJ)				Energy/pixel (nJ)
thresh	*nfeat*	*nlength*	CORE	CACHE	KEYPTS	TOTAL	
20	0	1	77.94	6.24	3.65	87.83	0.2859
20	1	0	78.92	6.32	3.70	88.94	0.2895
20	1	1	118.96	9.52	5.58	134.06	0.4364
30	0	1	74.63	5.97	3.50	84.10	0.2738
30	1	0	61.08	4.89	2.86	68.84	0.2241
30	1	1	89.36	7.15	4.19	100.70	0.3278
40	0	1	61.29	4.91	2.87	69.07	0.2248
40	1	0	46.67	3.74	2.19	52.59	0.1712
40	1	1	61.29	4.91	2.87	69.07	0.2248

It should be noted that for Table 4.5, the energy consumption was computed based only on one frame, without considering the frame rate requirement. For example, having a high threshold value will result in shorter execution time (and therefore lower energy) compared to that with a lower threshold value. As such, one configuration will finish a frame faster than the other, and will therefore have to wait longer, which is the case of over-margined designs. For EQSCALE, we propose to leverage on this by relaxing the supply voltage, thereby further reducing energy consumption. This will be further discussed in the succeeding section.

4.3 EQSCALE Results

Fig. 4.11 shows the measured maximum operating frequency, fmax, for different supply voltages, and the corresponding power consumption of the ORB accelerator testchip in 40nm CMOS. In Fig. 4.11a, the maximum frequency at 0.9V and 1V are the same, which we attribute to the testing setup limiting the operating frequency of the system. Thus, we consider only supply voltages from 0.6V to 0.9V. The trendline equation was included to allow for interpolation between measured points. It can be seen in Fig. 4.11b that, consistent with the simulation results in Table 4.4, the CORE consumes most of the power (~90%).



(a)

f_{max} (MHz)	Supply (V)	Power (mW)			
		KEYPTS	CACHE	CORE	TOTAL
45	1	0.18	0.30	3.14	3.6190
45	0.9	0.14	0.23	2.52	2.8820
40	0.8	0.09	0.16	1.76	2.0077
30	0.7	0.16	0.09	1.02	1.2765
15	0.6	0.04	0.09	0.39	0.5152

(b)

Fig. 4.11 Measured (a) maximum clock frequency f_{max} and (b) power consumption of different blocks at f_{max}

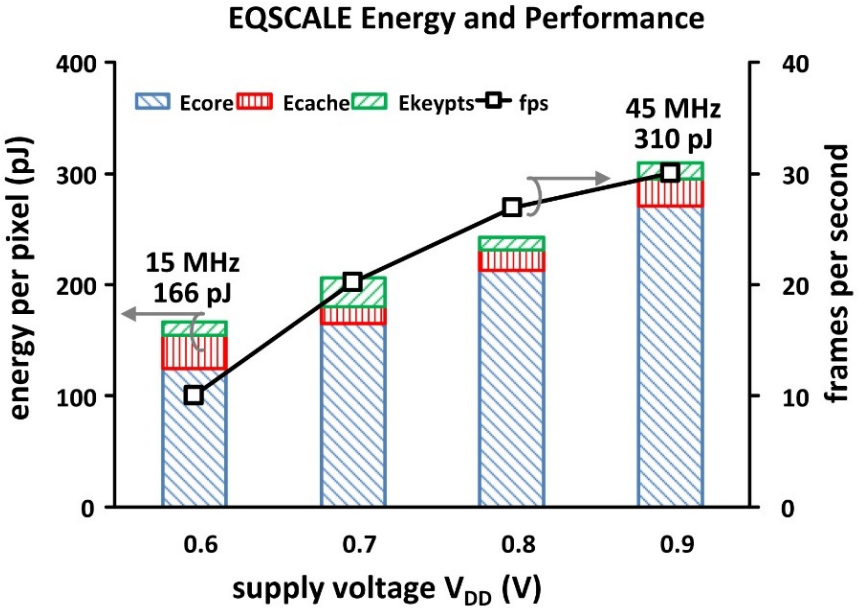


Fig. 4.12 Effect of V_{DD} scaling on frame rate and energy per pixel.

Energy per frame can be calculated using the data in Fig. 4.*11* and the number of execution cycles needed per frame. Energy reduction is then possible through V_{DD}

scaling, which in turn reduces frequency of operation. This translates to a degradation in performance either through the reduction in image resolution or frame rate. Fig. 4.12 shows the impact of the V_{DD} scaling on the frame rate and energy per pixel, for a VGA image. As can be seen from the figure, energy per pixel can be scaled from 310 pJ to 166 pJ when scaling VDD from 09V to 0.6V, and frame rate from 30 fps to 10 fps.

In EQSCALE, we use quality as a third dimension to this energy-performance tradeoff. Through the tunable knobs, we can tradeoff quality with energy and/or performance. Fig. 4.13 shows the energy-quality tradeoff when EQ knobs are individually swept in a 40nm testchip. Using recall as the matching performance metric, it can be seen from the figure that ORB turns out to have approximately the same quality as SIFT, when tuned to maximum quality (Q=1). At nominal V_{DD} and maximum quality, EQSCALE consumes a power of 2.9mW at VGA and 30fps, which results to an energy per pixel of 310pJ/pixel. When reducing *nlength* from 256 down to 128 bits, the energy is reduced by 34% compared to the 256-bit default value, with a quality degradation of 10% (see red curve in Fig. 4.13). Analogously, when reducing *nfeat* from 400 down to 200, the energy decreases by 35% with a quality degradation of 42% (see green curve in Fig. 4.13). Increasing *thresh* knob from 20 to 40 (60) reduces energy by 48% (64%) and degrades quality by 19% (53%). At *thresh*=60, ORB achieves approximately the same quality as SURF. At such quality, successful image recognition is still achieved, as confirmed by the bounding box around the recognized object, as generated by an offline RANSAC [53] algorithm done in MATLAB. As an example, Fig. 4.14 shows sample matching images using the graffiti image from [33] for different quality targets. At maximum quality (i.e., Q=1), the image is properly matched, as indicated by the green box on the image on the right. At minimum allowable quality (i.e., Q=0.4), many keypoints are missed and object detection starts failing, as indicated by the smaller size of the bounding box. At lower quality (e.g.,

Q=0.12), keypoints are no longer properly matched and thus no object is detected (i.e., the green bounding box disappears).

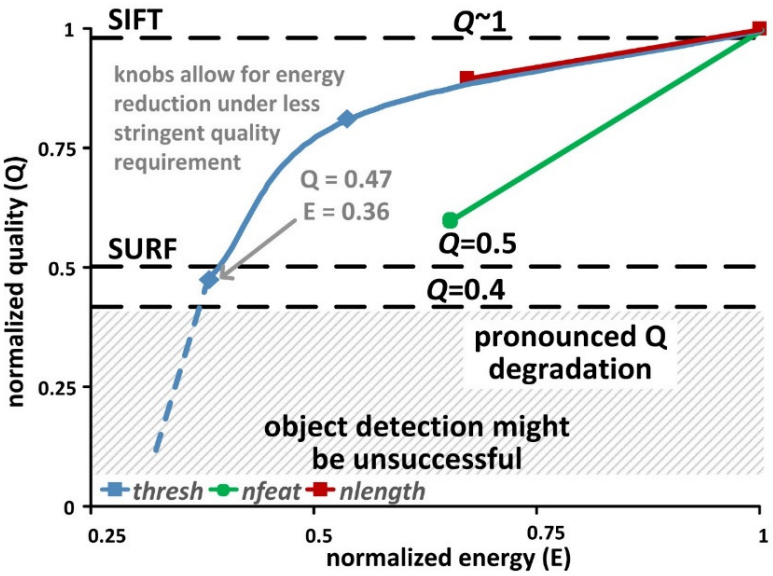


Fig. 4.13 Energy-Quality tradeoff when tuning knobs *nfeat*, *nlength* and *thresh* at nominal V_{DD}.

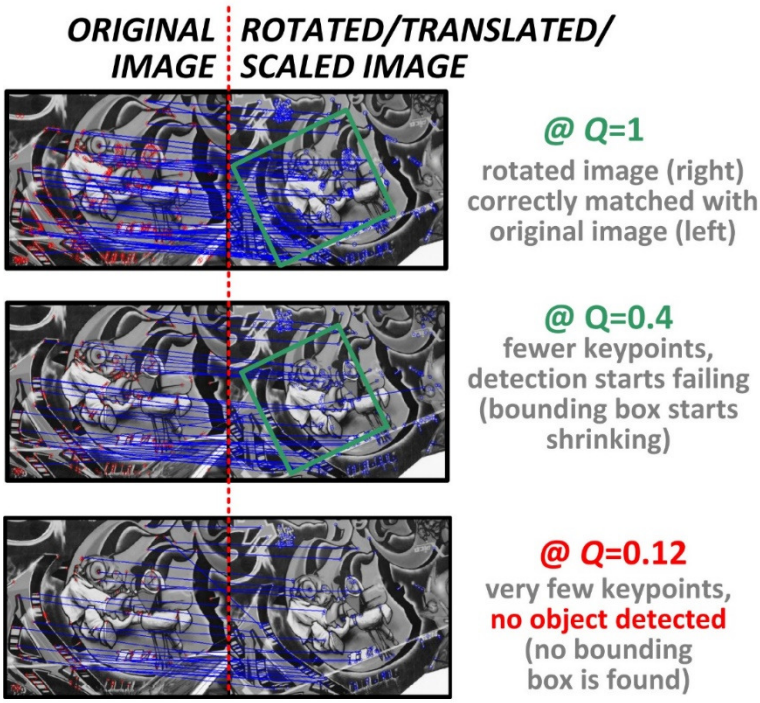


Fig. 4.14 Illustration of image matching at different values of Q.

The energy reduction in Fig. 4.13 is determined by the reduced number of execution cycles per frame, which also increases the throughput. Such excess

throughput can be used to relax the clock cycle, enabling more aggressive voltage scaling and further energy gains. When co-optimizing multiple knobs and V_{DD} scaling, energy is further reduced, as shown in Fig. 4.15. Comparison between the solid lines in Fig. 4.15 and those in Fig. 4.13 reveals that V_{DD} scaling offers an additional ~20% reduction in energy. When all knobs are adjusted for minimum energy, the 40nm testchip shows an energy/pixel of 55.6pJ and power consumption of 513uW on VGA format and 30fps. It should be noted that some combinations of EQ knobs (specifically, combining *nlength* and *nfeat*) may result in below minimum allowable quality (red boxes in Fig. 4.15).

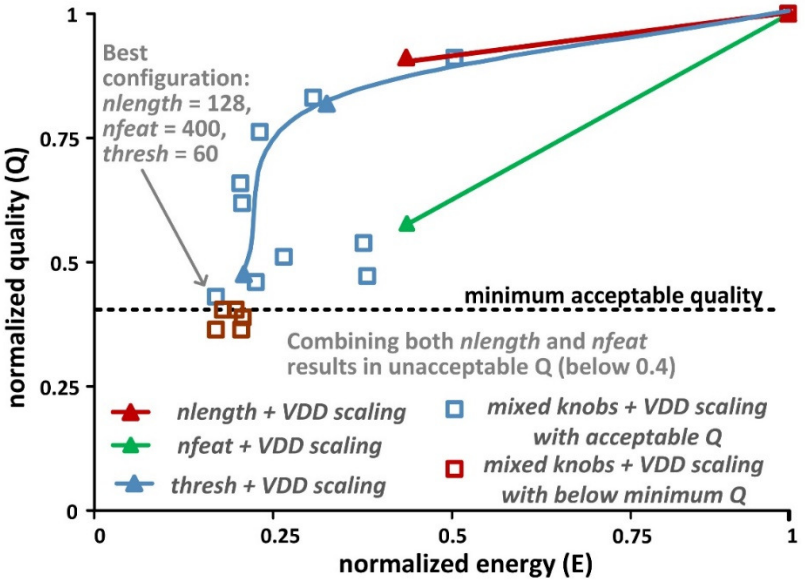


Fig. 4.15 Quality vs. energy with joint EQ knobs combined with voltage scaling.

Table 4.6 shows the comparison of EQSCALE with state of the art feature extraction ASIC implementations discussed in Chapter 2. Some parameters are normalized with respect to 40nm for easier comparison. Compared to [44], EQSCALE achieves 5.3X power reduction, thanks to the lower complexity of ORB. EQSCALE has the lowest energy/pixel at iso-technology, with an energy reduction of 5.7X and 7.5X over [46] and [44], respectively. EQSCALE also shows 1.8X area reduction compared to [46], thus exhibiting a favorable area/energy tradeoff as needed in IoT

applications.

Table 4.6 Comparison of Results

Parameter	JSSC'13 [57]	TCASVT'13 [46]	JSSCC'14 [44]	This Work
technology	0.13 µm	0.13 µm	28 nm	40 nm
supply voltage	0.65 ~ 1.2V	1.2V	0.47V	0.6-0.9V
on-chip memory	382kB SRAM	128kB SRAM	~7kB FIFO	~4kB Latch-Based
clock frequency	50~200MHz	200MHz	27MHz	15-45MHz
normalized clock frequency*	17.15-68.6MHz	68.6	38.57	15-45MHz
energy-quality scalable	NO	NO	NO	YES
power	320mW	182mW	2.7mW	0.51-2.9 mW
energy/pixel	11.57 nJ	0.93 nJ	0.293 nJ	55.6-310 pJ
normalized energy/pixel*	3.97 nJ	0.32 nJ	0.42 nJ	55.6-310 pJ
area (mm^2)	32	10.24	2.22	0.55
normalized area** ($F^2/10^6$)	1893.49	605	2831.63	343.75
frame rate (resolution)	30fps (HD)	94.3fps (HD)	30fps (VGA)	30fps (VGA)
targeted application	Unmanned Aerial Vehicles	Embedded Vision System	Micro Autonomous Vehicles (MAV)	MAV, Smart Cameras
algorithm	SIFT	FAST-BRIEF	SURF	ORB
operation	matching with external database	maching with descriptor cache	upto description	upto description

* normalized with respect to 40 nm technology, assuming 0.7X energy decrease/generation
** F is the minimum feature size of the technology

4.4 Effect of Cache Size

For the case of the EQSCALE implementation in the previous section, detection is done in parallel for 7 pixels within a row. Since description works within a 31x31 patch around a keypoint, 42 pixels are accessed per row. This means that 5/6 of the

pixels will need to be re-accessed in later time for detection. A possible improvement would be to widen the effective cache width to reduce the cache re-access ratio per row. We refer to Fig. 4.4 for illustration, to do detection on 7 pixels, we need to access 42 pixels of the row. For a detection on 14 pixels, we access 42x2 = 84 pixels. In general, for detection of 7*N* pixels, we access 42*N*. Thus, the re-access ratio is 42*N*/7*N* = 6. This means that we are accessing each pixel 6 times to perform that whole feature extraction. Widening the width reduces this re-access ratio, which is proportional to the energy consumption. This is better illustrated in Fig. 4.16, where EQSCALE v1 (the implementation in the previous section where width = 42) re-access is 6. Increasing the CACHE width by 3x gives a re-access ratio of 1.38, which is 4.3x better than EQSCALE v1. This is labelled as EQSCALE v2.

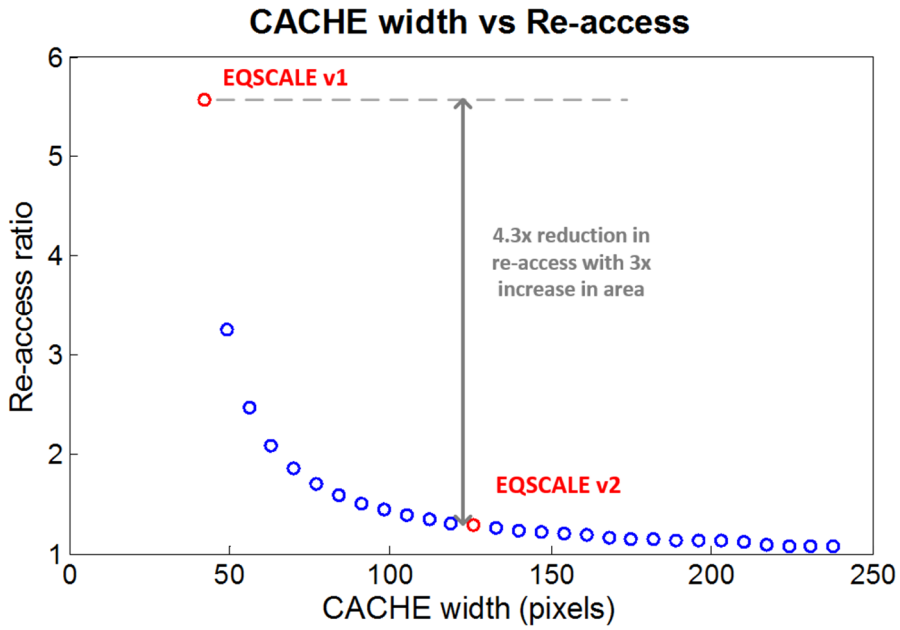


Fig. 4.16 Effect of cache width on re-access ratio. Corresponding points for EQSCALE v1 and EQSCALE v2 (3x size) are indicated.

The re-access of pixels affects energy consumption through increase in access energy, as well as through increase in number of cycles to do the operation. Table 4.7 extends this phenomenon to the overall performance of the ORB accelerator. The column *N* represents the number of columns, same as the variable N in the previous paragraph. The frequency column is the required frequency to work with the indicated

image resolution. As can be seen from the table, EQSCALE v2 is ~4x faster than EQSCALE v1, at the cost of 3x area. It should be noted that the frequency indicated in the table is an optimistic value (actual required frequency will be higher).

Table 4.7 Effect of increasing CACHE width by 3x

		EQSCALE v1			EQSCALE v2		
Format	Image Size (pix*pix)	N	Re-access	Frequency (MHz)	N	Re-access	Frequency (MHz)
HQVGA	120 x 240	9	4.2	3.63	1	1.05	0.91
QVGA	240 x 320	13	5.08	11.69	2	1.05	2.42
VGA	480 x 640	26	5.6	51.61	5	1.31	12.10
WVGA	480 x 768	31	5.6	61.93	5	1.31	14.52
XGA	768 x 1024	42	5.74	135.48	8	1.31	30.97
HD	900 x 1600	66	5.79	249.98	10	1.4	60.48
FHD	1080 x 1920	79	5.79	360.46	11	1.28	79.83

One issue with the EQSCALE v1 implementation is that it seems far too slow compared to others in Table 4.6. Indeed, this is also confirmed in Table 4.7. Another reason (which is not considered in Table 4.7) is because access to memory is halted each time keypoint description is in progress. This increases the execution time, and therefore reduces the frame resolution possible given a fixed frame rate and same knob settings or accuracy. One solution is to add another set of read ports for CACHE. Doing all these changes to the CACHE affects not only the CACHE but the other blocks as well.

Table 4.8 Area Comparison between EQSCALE versions

Block	EQSCALE v1		EQSCALE v2	
	Synthesis Area (μm^2)	Layout Area (μm x μm)	Synthesis Area (μm^2)	Layout Area (μm x μm)
CORE	99930.60	675 x 350	134323.66	785 x 480
CACHE	82127.60	420 x 470	252675.88	1010 x 520
KEYPTS	27833.45	240 x 355	38406.16	240 x 415
TEST	6199.58	375 x 100	11936.81	375 x 100
CLKGEN	451.05	230 x 100	326.69	230 x 100
SCANCHAIN	825.73	260 x 100	825.73	260 x 100
TOTAL		**1850 x 850**		**2500 x 880**

Area is one of the obvious changes from EQSCALE v1 to EQSCALE v2. The area comparison between the two versions is shown in Table 4.8. For EQSCALE v2,

indeed, CACHE is ~3x larger. To support the increase in CACHE size, and therefore the address width, CORE area likewise increased. From the table, CORE is 1.6x larger and KEYPTS is 1.2x larger. The microphotograph of the die is shown in Fig. 4.17.

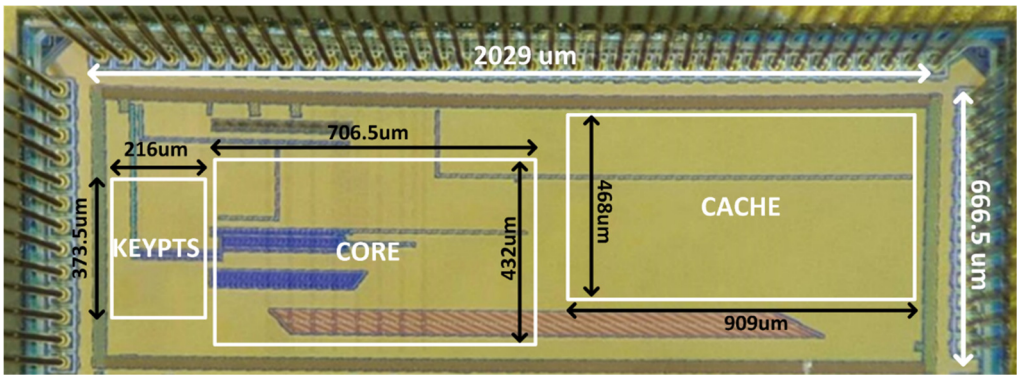


Fig. 4.17 EQSCALE v2 chip microphotograph

The Detector block remains the same, taking 7 parallel detections at a time. To simplify memory access arbitration, the two sets of read ports are separated such that one pair is for detection (Detector + NMS) while the other pair is for description (Orientation + Descriptor). For NMS, the FIFO size is expected to contain all pixels in 5 rows. Having 91 interest points per row, 5 rows would require a FIFO size of 455 entries. To reduce the needed size, statistical simulations were done to determine the optimal FIFO size that would be as small as possible but without interrupting the operation due to overflow. Histogram of needed NMS buffer size considering consecutive 5 rows for the different images is shown in Fig. 4.18. It is shown in the figure that a maximum size of 129 was determined for the images considered. Since this is image dependent, margin was added, making the NMS buffer size of 140. In the unlikely case that an overflow still occurs (number of keypoints in 5 consecutive rows is greater than 140), a stall is generated to halt the detection until space is cleared out.

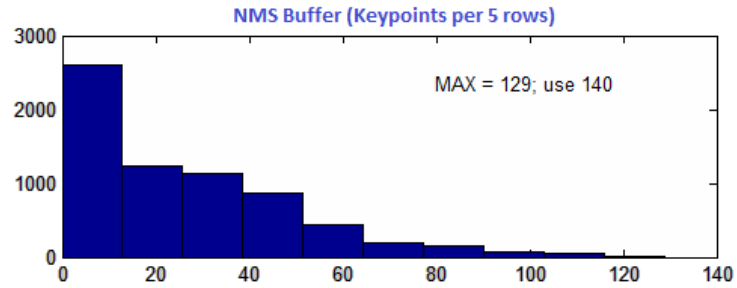


Fig. 4.18 NMS buffer size histogram. Maximum size needed for the considered data is 129.

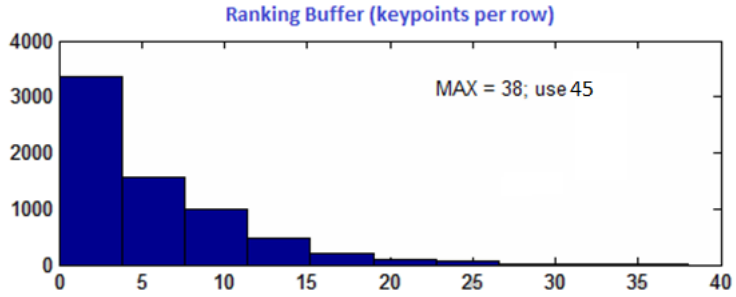


Fig. 4.19 Ranking buffer size histogram. Maximum number of entries needed is 38.

Operation of the Ranking block remains the same, except that similar to the NMS block, the buffer size has to be increased. In EQSCALE v1, the buffer size was 3 entries, corresponding to the maximum number of keypoints per row. For the case of EQSCALE v2, the rate at which the Ranking block can process data also has to be considered. After similar extensive simulations, it was determined that a buffer size of 45 entries should suffice (maximum seen is 38). The histogram is shown in Fig. 4.19. Like in the case of the NMS block, a stall signal (to halt NMS and then possibly Detector) is generated in the unlikely case that an overflow occurs.

4.5 Further improvements to EQSCALE

As illustrated in Chapter 1 (Fig. 1.2), an object detection and classification block is needed to complete the image analysis process. Matching is currently done offline, following the brute force method used in the original ORB. There are several other possible approaches for implementation, and we mention some of them in Section 4.5.1. From previous sections, the Ranking block occupies most of the CORE area and

consumes most of its power. Therefore, improving the Ranking block would be a worthy endeavour, and is discussed in Section 4.5.2. Finally, other energy-efficient schemes related to the energy-quality scalability of EQSCALE are covered in Section 4.5.3.

4.5.1 Object Detection and Matching

As mentioned in the previous chapter, matching in ORB is done via brute force method, forcing a match for each retained keypoint by comparing them with every keypoint in the database. As such, the recall values were low, despite successful object detection. One advantage of ORB over SIFT and SURF is the short descriptor length. Thus, bitwise comparison can be done in parallel (at the expense of additional hardware). Instead of forcing a match between described keypoints and the database, a nearest neighbour threshold can be applied, below which, no match will be declared. The histogram of hamming distances between pairs of keypoint 256-bit descriptors is shown in Fig. 4.21, showing a mean of 127 and standard deviation of 14.3. This means that we can use 127-3*14.3 = 84 as a threshold, above which they are not a match.

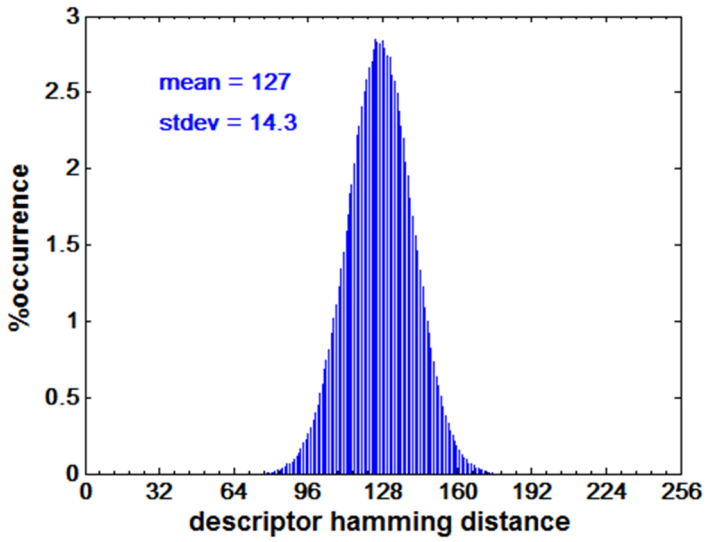


Fig. 4.20 Descriptor hamming distance histogram shows a mean of 127 and standard deviation of 14.3 for 256-bit length descriptor.

Object detection and classification can be done by removing outliers using PROSAC [52] or RANSAC [53], and/or using bag of words or keypoints [58]–[60]. One area that can benefit from using the bag of words is classification using a large database. Object classification can be done, using machine learning, to match inexact histograms with histograms from the database. Region-of Interest (ROI) detection can be done in parallel, to reduce the effective size of the image to be processed.

4.5.2 Ranking

Cache data reuse to eliminate the need for external frame buffer memory dictates that description be done on some keypoints (before they are replaced in CACHE) even if they are not one of the top ranked keypoints within the frame. Thus, the *nfeat* knob, which limits the described keypoints only to those with high corner scores, does not really offer the best energy reduction advantage it is supposed to offer. An alternative would be to reduce the complexity of the Ranking block and keep only a few of the lower scores within the top nfeat keypoints. In this way, the KEYPTS block can be reduced or completely removed, at the same time reducing the number of cycles needed for the Ranking block. This idea is simplified below in Fig. 4.21. Fig. 4.21a shows the current implementation of the Ranking block, where a new item (*score*) is compared with *checkpoint* values to find the corresponding 40-entry bin that the new item belongs to. Since there are 10 checkpoints, with 40 entries per bin, maximum number of comparison to insert the new item is 50.

In this proposed algorithm, a new entry is simultaneously compared with 3 pointer values: *MAX*, *MID* and *MIN* (Fig. 4.21b). *MAX* is the highest score in the frame. *MID* is the highest score within the KEYPTS memory. *MIN* is the lowest value with the *nfeat* retained keypoints in the frame. It should be noted that *nfeat* is used as a counter and a new knob enqueue may be added to indicate the size of the KEYPTS buffer. Fig. 4.21c shows the simplified flowchart for each new item. If *score*<*MIN*, the

entry is simply discarded. A *counter* for the number of retained keypoints is included, if it is less than *nfeat*, then data are simply inserted. Otherwise, values in the KEYPTS memory may be replaced. If score is between *MIN* and *MID*, it is inserted onto the KEYPTS memory. If the memory is full, *MID* will be pushed out and replaced if *counter* < *nfeat*; otherwise, *MIN* will be pushed out and replaced. If *score*>*MAX*, *MAX* is replaced by score. If *counter* = *nfeat* (*counter* stops at *nfeat*), *MAX* is pushed into KEYPTS (replacing *MID* with *MAX* and pushing *MIN* out). If score is between *MAX* and *MID* and *counter*=*nfeat*, score is pushed into KEYPTS (similar to *MAX*, without replacing *MAX*). If, however, *counter*<*nfeat*, then the KEYPTS entries and pointers are retained as is.

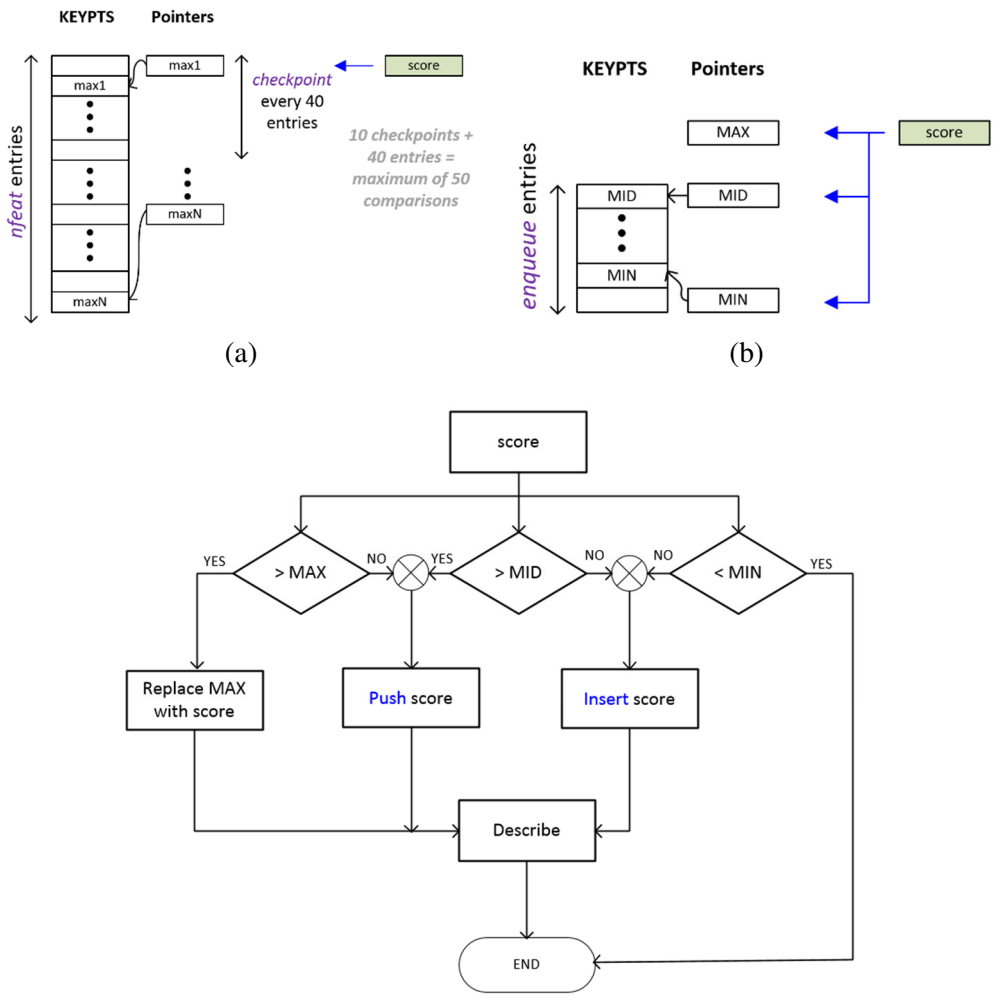


Fig. 4.21 Illustration of proposed ranking implementation. EQSCALE ranking implementation is shown in (a). Modified version (b) uses less number of pointers and less KEYPTS entries. Algorithm is shown in (c)

It should be noted that in all these cases, a single cycle simultaneous comparison is enough to determine if the new entry will be described or not. The complexity of the actual ranking is also reduced to the first comparison plus the number of *enqueue* items. The total number of described keypoints will be equal or less than the current EQSCALE implementation.

4.5.3 Other Energy-Efficient techniques

The possibility of having an ROI detection was mentioned in Section 4.5.1. Aside from reducing the effective size of the image to be processed, and therefore reduce the number of detections needed per frame, ROI detection also gives more space and freedom for energy-quality scaling. For example, feature extraction can start with a lower quality target prior to ROI detection, and eventually (after ROI detection or estimation) increase the quality target with the smaller image size. Since the effective size is smaller, execution time will be less and therefore EQ knobs can be adjusted to improve the quality. Similarly, the ROI also serves as an EQ knob, since the image size is approximately proportional to the number of execution cycles. By adjusting the voltage (and therefore frequency), depending on the size of the ROI, energy is effectively reduced for a given quality target.

A voltage-frequency-architecture co-optimization completes this cycle, allowing for some blocks (i.e., parallel detector units, KEYPTS bank) to be switched off to reduce power consumption, for the same voltage and frequency settings. This of course requires careful exploration of the methodology to determine optimal configuration of voltage, EQ knobs and power gate switches for a given quality target, image resolution and image type or application.

Chapter 5

SRAM for Image and Video Application

Memory contributes to a significant percentage of area and power consumption in digital systems. The universal choice for dense on-chip memory is the static random access memory (SRAM). We have shown in the previous chapter that the feature extraction accelerator can do away with the external DRAM for the frame buffer. However, due to the necessary re-access and possible stalls, a sub-frame buffer is needed. For our purpose, an SRAM is a suitable option, giving the best balance between area and energy.

In this chapter, we will cover the design and simulation results of our proposed low energy SRAM for image and video applications. We start with some background and metrics on SRAMs in Section 5.1. Section 5.2 will cover the state of the art in SRAM bitcells for low energy and near-/sub-threshold operation, including application-specific SRAMs for image and video. Finally, we discuss simulation results for our proposed low energy non-precharged SRAM (NPSRAM) for image and video applications.

5.1 SRAM Basics and Metrics

An SRAM has three possible operations: (1) standby; (2) read; and (3) write. Associated with the 3 operations are the 3 possible failures. Although not common in normal strong inversion operation, lowering the V_{DD} could cause a hold failure, where the cell is unable to retain its value. A read failure, on the other hand, happens when the cell value is flipped during a read operation, due to the precharged bitlines. A write

failure is the opposite of the read failure, where a write operation fails to flip the cell value as intended.

The de facto implementation of SRAMs is the 6T SRAM [61] as shown in Fig. 5.1. For a read operation, the wordline *WL* activates the access transistors M5 and M6, allowing the *Q* and *Q'* to be written to *BL* and *BL'*, respectively. Since the access transistors and NMOS transistors, which are not good passers of logic 1, the bitlines (*BL* and *BL'*) are precharged to V_{DD} prior to activating *WL*. For a write operation, the intended values for *Q* and *Q'* are placed in *BL* and *BL'*, respectively, prior to activating *WL*.

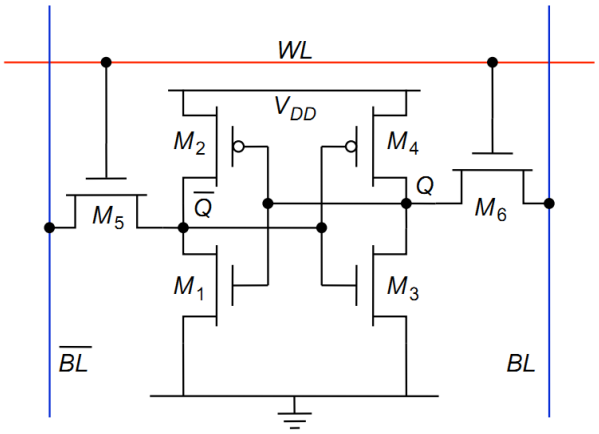
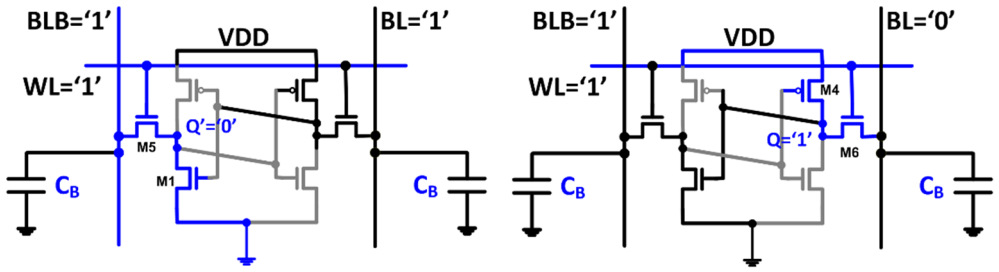


Fig. 5.1 Conventional 6T SRAM

It should be noted that the only difference between a read and a write access as discussed in the previous paragraph, is the value on BL or BL' (one of is 0 during a write operation, while both are 1 during a read operation. As such, care must be taken in sizing the transistors to avoid read or write failures. This is illustrated in Fig. 5.2, showing. Fig. 5.2a shows a possible read 0 contention, where, where the internal value is 0, and because of the precharge in the bitline, the internal value (rather than the bitline) could flip. To avoid this, M1 and M5 should be sized such that the voltage at the internal node (in this case, Q') does not go higher than the switching threshold of the forward inverter. Fig. 5.2b, on the other hand, shows a possible write 0 contention, where the internal value (in this case Q) can flip the bitline instead of copying it. This

can be avoided by sizing M4 and M6 such that the internal node voltage (Q) will be pulled down enough to switch the feedback inverter.



(a) (b)

Fig. 5.2 SRAM sizing considerations for (a) read 0 and (b) write 0 contention

To quantify the robustness of the SRAM, the static noise margin (SNM) is used [62]. Read (write) margin indicates the amount of noise the SRAM can tolerate before a read (write) failure occurs. This can be visually illustrated using the butterfly curve [63], [64], such as that shown in Fig. 5.3. The SNM is the length of the side of the largest square that can fit in the eye of the butterfly curve.

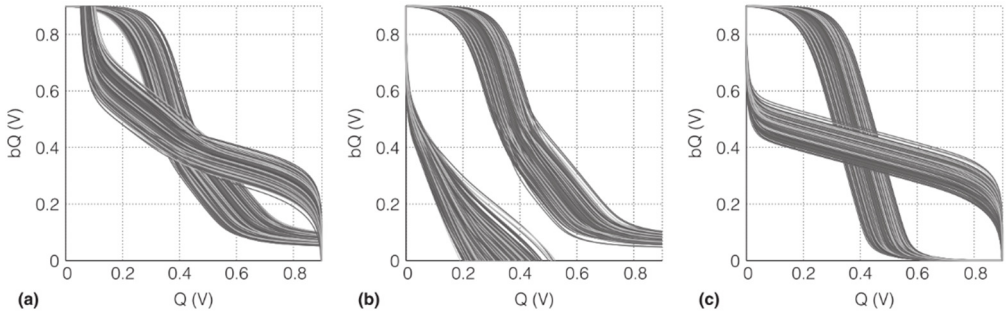


Fig. 5.3 Static noise margin for (a) read, (b) write, and (c) hold [64]

5.2 State of the Art

Although guidelines for sizing the 6T SRAM minimizes the probability of failure, this alone is not enough when operating at near- or sub-threshold voltages. Fig. 5.4 shows a sample butterfly curve of a 6T SRAM at near-threshold. It can be seen that the SNMs are degraded compared to that in Fig. 5.3, and especially for read SNM, no the curves overlap, indicating that a read failure may occur. With the demand for low power and low energy, operating at near-/sub-threshold is almost mandatory, and the

memory system should also be compatible. As such, several other bitcell topologies have been proposed.

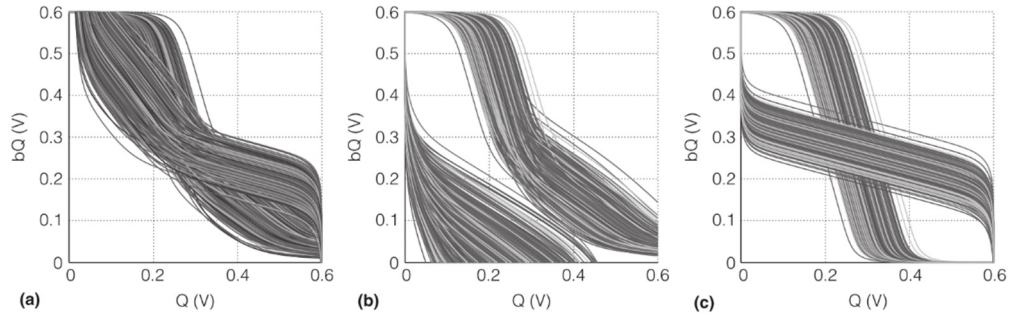


Fig. 5.4 SNM for (a) read, (b) write and (c) hold at near-threshold voltage

5.2.1 Near-threshold SRAMs

To alleviate the read static margin degradation in near-threshold operation, a 7T transistor was proposed [65], as shown in Fig. 5.5. Their approach is to break the forward path during a read operation, by adding a data protection transistor as highlighted in the figure. As discussed in the previous section, the problem with read operation is that the precharged bitline results in a possible read 0 failure. In this case, because of the added transistor, the forward inverter is disconnected from ground, and therefore *V2* will not go low enough to flip the inverter. Read 0 failure is thus avoided.

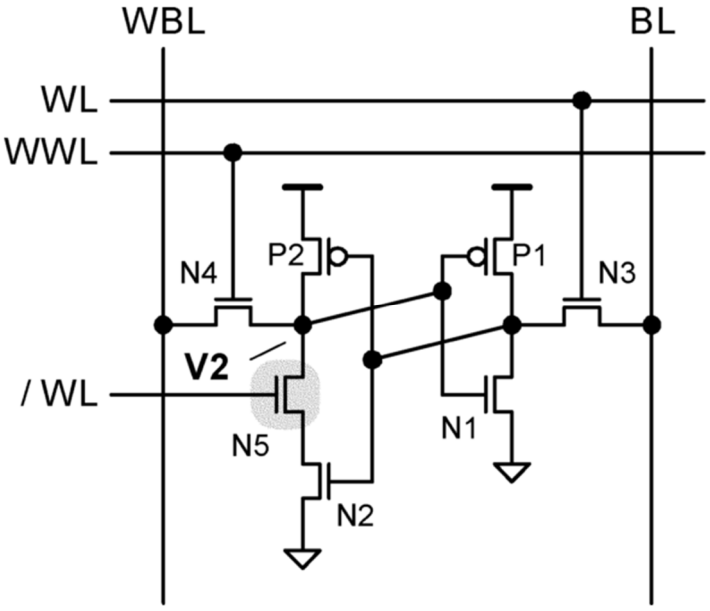
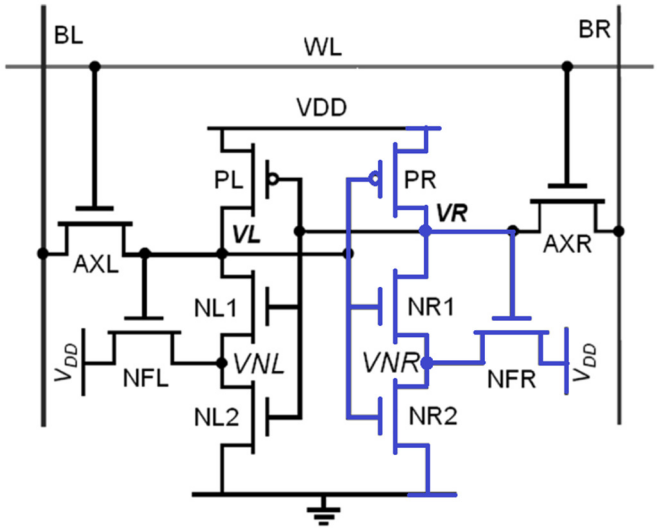
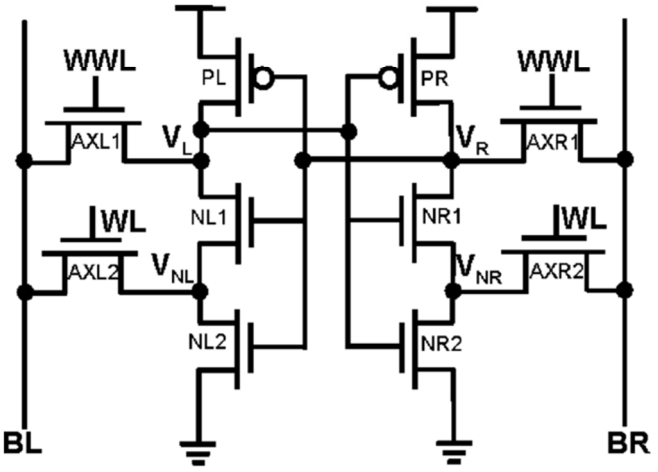


Fig. 5.5 7T SRAM [65] with added data protection transistor to remove read 0 failure.

Another solution to improve SRAM robustness is to replace the inverter with a Schmitt trigger, as shown in Fig. 5.6. Using a Schmitt trigger allows the switching threshold up or down depending on the direction of the data switching. They modified the Schmitt trigger to reduce the number of transistors, as highlighted in Fig. 5.6a, and were able to show successful operation down to 160mV [66]. They further modified the circuit to improve the read margin, as shown in Fig. 5.6b [67].



(a)



(b)

	WL	WWL
Read Mode	1	0
Write Mode	1	1
Hold Mode	0	0

Fig. 5.6 Schmitt trigger based SRAM: (a) replacing inverter with a modified Schmitt trigger [66], (b) modified version for improved read margin [68].

Another 6T SRAM (Fig. 5.7) uses transmission gate instead of the NMOS access transistor [69], [70], however, it requires virtual supplies for proper operation. They were able to show proper operation down to 193mV.

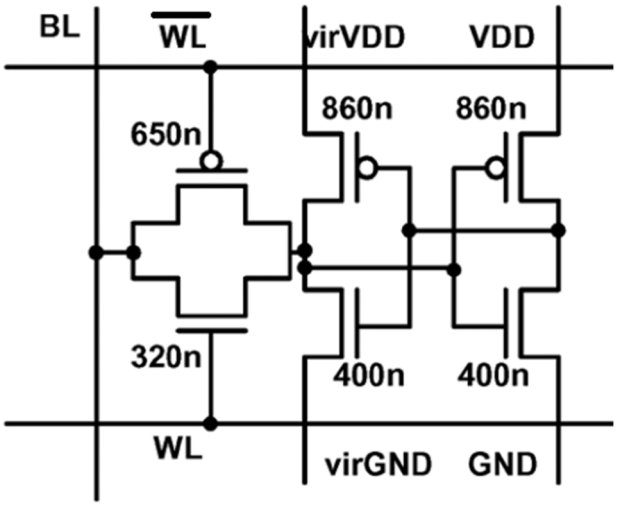


Fig. 5.7 Single-ended 6T SRAM using transmission gate as access transistors [69].

One of the most popular topology to date for near-threshold SRAM is the 8T [71], where the read bitline is separated from the write bitlines. Thus, the internal node is not affected by the read port, and therefore read margin is greatly improved. The schematic is shown in Fig. 5.8. To reduce leakage power in the read bitline (and therefore allow more bitcells to be connected), a footer can be used [72]. Other modifications in the read buffer circuitry have been proposed to further reduce the minimum operating voltage and reduce leakage power [73]–[76]. A summary of the different topologies is presented in Table 5.1 for better comparison.

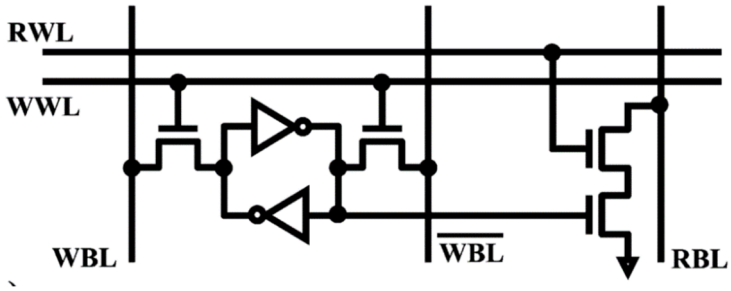


Fig. 5.8 8T SRAM with separate read port [71]

Table 5.1 Comparison of SRAM bitcells

SRAM	number of transistors	number of ports	technology (nm)	array size (bits)	$V_{DD,min}$ (mV)	speed	normalized area (wrt 6T)
7T [65]	7	5	90	32x8x16	440	50MHz @ 0.5V	1.13
ST1 [66]	10	3	130	256x16	160		2
ST2 [68]	10	4	130	128x16	150	270kHz @300mV	2
TG [70]	6	3	130	128x16	193	5.6MHz @0.5V	1.42
8T [71]	8	4	65	32kb	410	295MHz @0.41V	
F-8T [72]	8	4	65	8x256x128	350		1.3
SE10 [74]	10	5	65	256k	400	475kHz @0.4V	1.66
D10 [73]	10	5	130	480k	200	120kHz @0.2	
RS [75]	10	6	180	64x32	300		

5.2.2 Application-Specific SRAMs

When it comes to image and video data, one area we can leverage on is the high correlation of neighbouring pixel values. For a greyscale image, a pixel is usually represented using 8 bits and, per literature [77], more than 50% of the variations in pixel value lie within 3 bits of the data, showing the high correlation of the pixel values. As such, they proposed a prediction-based scheme to reduce the bitline switching [77], [78]. Their bitcell is similar to the footed 8T [72], but with 2 read ports (Fig. 5.9). The footer values are dictated by the predicted values (*pred* and *predB*). During a write operation, values are sent to *BL* and *BLB* then *WWL* is asserted to write *BL* and *BLB* to *Q* and *QB*, respectively. During a read operation, *RBL0* and *RBL1* are precharged to V_{DD} and the predictions, *pred* and *predB* are set before *RWL* is asserted. If the prediction is correct, the bitline with a 0 prediction will be discharged, while the other bitline is disconnected from ground. If the prediction is wrong, both bitlines will remain at V_{DD}. Obviously, the prediction mechanism is critical to their system. Indeed, their results

show that if all predictions are wrong, the access energy will be ~20% more than that of 8T. In the opposite extreme, if all predictions are correct, the resulting access energy will be ~20% less at nominal voltage and up to ~40% less at 0.6V.

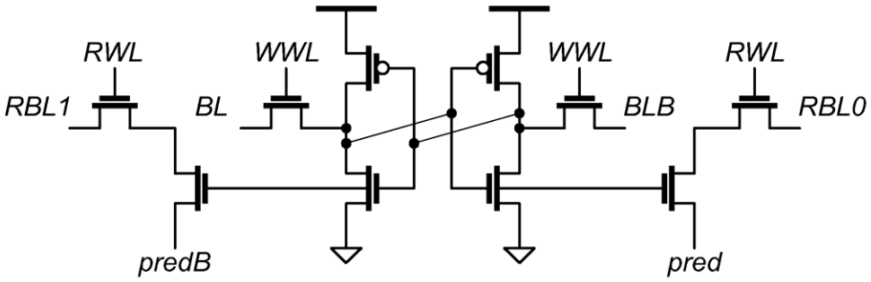


Fig. 5.9 Prediction-based SRAM for reduced bitline activity (PB-RBSA) [77]

Another application-specific SRAM design for image and video leverages on the fact LSBs of a pixel will not affect the image that much, and therefore can be allowed to make errors. Thus, they proposed a heterogenous SRAM sizing [79], using larger sized 6T SRAMs for LSBs and smaller 6T SRAMs for MSBs. Similarly, work in [80] proposed to use 8T SRAMs for MSBs and 6T SRAMs for LSBs, while [81] proposed to use 8T SRAMs for LSBs and 10T SRAMs for MSBs.

5.3 Non-Precharged SRAM (NPSRAM)

Given the high correlation between adjacent pixels in an image or video, the probability of getting almost equal values with successive access is high. This motivates our implementation, which reduces bitline switching by removing the pre-charge cycle of the SRAM. Thus, we have a non-precharged SRAM (NPSRAM). Our approach is to reduce the switching activity of the bitlines by simply removing the bitline pre-charging phase, unlike in [77], where bitline switching is reduced through prediction of values. This is justifiable because we expect the data to be almost always the same (with probability greater than 60%). For our purpose, we designed a 256x64x2 memory array without column multiplexing. This design was simulated in a 65nm Low-Power (LP) CMOS process.

To reduce energy consumption further, we target a near-threshold voltage operation of 0.5V, as going into near-/sub-threshold voltage reduces power consumption quadratically. However, this requires different additional assist mechanisms to allow robust performance and perform proper read and write operations. The most common way is to separate the read and write port, like that of the 8T transistor [71]. We use the same concept of separating the read from the write port in the 8T. Furthermore, we propose to reduce the energy consumption even further by removing the precharge phase, thereby removing unnecessary 0-1-0 transition in the bitline. To support the removal of the precharging phase, we add an inverter with enough gain to the access transistor to drive the bitline. The circuit is shown in

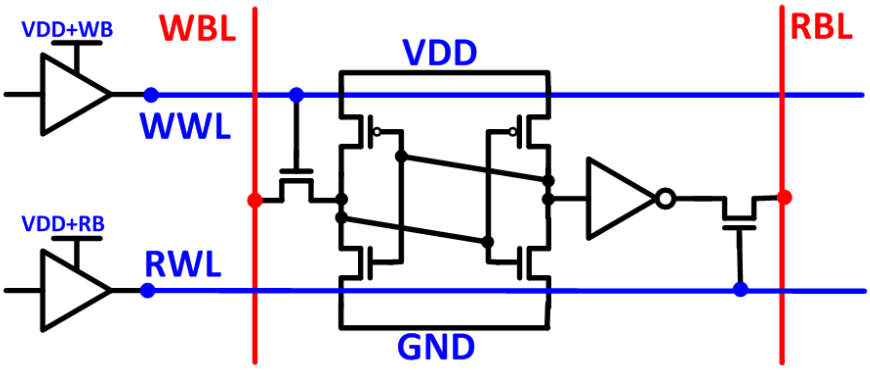


Fig. 5.10 Proposed non-precharged SRAM (NPSRAM)

For additional read and write assist, we investigated three different flavours of our design: (1) Base; (2) Drowsy and (3) Assisted. The base design would have the minimal assist, with just wordline boosting to ensure proper write. The drowsy design is based on the drowsy cache concept [82], where the supply voltage is reduced when not reading. During a read operation, the voltage is returned to 500mV. The assisted version is effectively an opposite approach as the drowsy, where the voltage is reduced during a write operation. For both the drowsy and assisted versions, wordline boosting was still considered. Although other assist techniques [83] have already been proposed and shown to be effective, we limited out investigation to these techniques to have the simplest possible assist circuitry that will ensure proper operation of the bitcell.

We evaluated the performance of the designs using 2000-pt Monte Carlo simulations in Cadence Virtuoso®. Delay and energy were evaluated to choose which scheme to implement. One thing to take note for these three schemes/flavours is that all three can still be tested with the same memory array, provided proper separation of supply voltages and signals is ensured. Thus, in terms of area, both drowsy and assisted versions will have the same area. The base design can be made smaller, since the supply voltage can be routed horizontally or vertically, whichever could give better area. For the case of drowsy and assisted, the supply voltage must be routed horizontally as each row could have a different voltage depending on whether it is doing a read (VDD=500mV) or a write (VDD=350mV). For our design, the drowsy and assisted versions are 1.4x larger than the base.

Table 5.2 shows the delay comparison of the three schemes mentioned above. It should be noted that these values were evaluated without the necessary drivers (some signals are still ideal), decoders and sense amplifiers. Delays are measured as the VDD/2 delay, which should be a pessimistic estimate once sense amplifiers are inserted. For simplicity and for fair comparison, we have set the wordline boosting to 200mV above the supply (500mV). Thus, the supply voltage of the buffer drivers for the read wordline (RWL) and write wordline (WWL) is 700mV. We include the 8T in our comparison. For fairness, we also implement the same assist techniques with the 8T.

Table 5.2 Delay comparisons

Bitcell	**μ Mean Delay (ns)**	**σ Delay Stdev (ns)**	**$\mu + 3\sigma$ (ns)**
8T	15	2.84	23.52
Base	10.8933	1.5	15.39
Drowsy	11.1	1.92	16.9
Assisted	16.72	5.11	32.06

It can be seen from Table 5.2 that the base design could be faster than the 8T, which in turn is faster than the drowsy or assisted schemes. The faster response of the

base design is expected since the supply voltage is constant and no timing penalty for supply voltage switching is incurred.

Another more important parameter we are concerned with is the energy consumption. To approximate the energy of the whole memory array, we evaluate active and idle energy consumption of cells and extrapolate to get the total energy of the whole array for read and write operations as well as when idle. Table 5.3 shows the energy comparison of the bitcell schemes, including that of the 8T cell for comparison.

Table 5.3 Energy comparisons

Bitcell	**Read Energy (fJ)**	**Write Energy (fJ)**	**Idle Energy (fJ)**
8T	796.42	804.49	117.2
Base	1083.05	906.91	597.89
Drowsy	778.92	870.77	256.384
Assisted	713.07	778.5	195.26

From Table 5.3, we can see that although the base design gave the best performance in terms of speed in Table 5.2, it also has the highest energy. This is because the supply voltage for the base design is constantly at 500mV, unlike in other bitcells (including the 8T), where the voltage is lowered to 350mV during some parts of the operation. Looking at the energy consumption of the drowsy and assisted versions, on the other hand, we can say that our proposed bitcell has the potential of offering a low energy alternative SRAM bitcell. Comparing the drowsy and assisted versions, we can also say that the assisted consumes less energy compared to the drowsy version, making it more suitable for our low-energy application requirement.

To approximate the energy consumption of the whole array, we used the energy per operation (including energy when idle) of the SRAM and extrapolate for the whole 256x64 array. Shown in Fig. 5.11 is the energy per operation of an 8T SRAM and our proposed non-precharged SRAM (labelled here as NP2). We can see from the figure that our design can potentially save 50% of read energy. Write energy will essentially be the same for both as we are applying the same write-assist technique to both SRAMs.

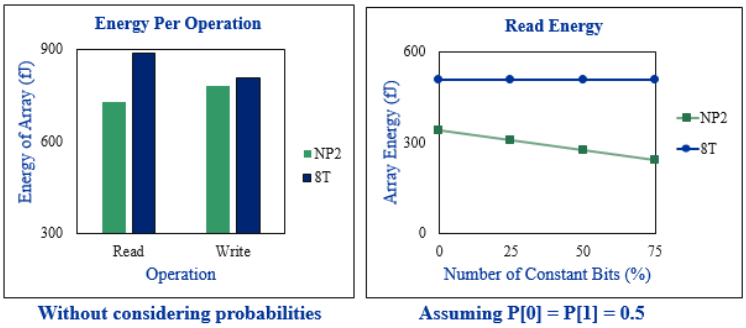


Fig. 5.11 Estimated energy of memory array

After showing the promise of the proposed SRAM, we proceeded with the layout of the bitcells (Fig. 5.12) as well as the peripherals circuitry, such as the drivers and decoders. Shown in Table 5.4 is the area comparison of the SRAM with the 8T SRAM. We can see from the table that the proposed SRAM cell incurs a 15% area overhead relative to the 8T SRAM.

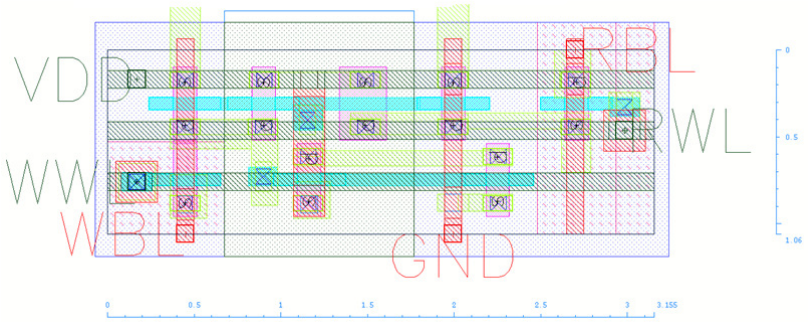


Fig. 5.12 NPSRAM bitcell layout

Table 5.4 Area comparisons

Metric	8T	NP2
Area (F^2)	750	859
Normalized Area	1	1.15
Normalized to 6T	1.3	1.5

Chapter 6

Secure Chip Identification Using PUFs

This chapter covers some background on physically unclonable functions (PUFs) and a discussion on our work on PUFs for secure chip identification. Section 6.1 covers the basic introduction to PUFs. Section 6.2 discusses the properties of PUFs and the metrics used to evaluate them. The state of the art in PUFs is covered in Section 6.3, followed by our proposed class of static monostable PUFs in Section 6.4. Finally, in Section 6.5, we cover some possible future work with PUFs.

6.1 PUF Introduction

The pervasiveness and the prospectively very large number of deployed nodes monitoring the environment, people and goods, makes security a fundamental challenge, especially in IoT applications. Security issues are expected to arise in terms of data authenticity, integrity and confidentiality. Indeed, it is necessary to assure that the data and the sender are legitimate, the data has been sent uncorrupted, and oftentimes data needs to be unreadable from an unintended receiver. Accordingly, security must be assured down to the hardware level, as the authenticity and the integrity need to be assured also in terms of the hardware implementation of each device or node (i.e., each node needs to be confirmed to be authentic and intact, while signalling in case it has been counterfeited or tampered with).

In the recent past, Physically Unclonable Functions (PUFs) have emerged as potentially highly secure and lightweight solution to ensure data and hardware security, assuring trustworthiness down to the chip level [84]–[88]. A PUF is a function that

maps an input (digital) challenge to an output (digital) response in a repeatable but unpredictable manner, leveraging on chip-specific random process variations. PUFs are sometimes referred to as "silicon biometrics", i.e. something equivalent to a "chip fingerprint" that is unique for each die. As such, it eliminates the need to store any key, as the latter is naturally generated and embedded into the chip during its manufacturing. This avoids the need for key programming (e.g., via fuses or e-Flash), and makes devices less prone to the many existing attacks that uncover the content of memories [89], as discussed below.

PUFs are used for chip identification and authentication [86]–[88], [90], [91], secure key storage and lightweight encryption [84], [92], hardware-entangled cryptography [93] and identification of malicious hardware [94]. In this thesis, we focus on PUFs for chip identification and authentication, and cover the other applications towards the end of the chapter. Chip identification and authentication are typically performed by preliminarily storing all challenge-response pairs (CRPs) of the chip PUF in a secure database, during a first enrolment phase. These (or a subset thereof) are used to verify the response of the chip to a given challenge during in-field operation, making sure not to reuse CRPs to reduce susceptibility to cloning, and counteract replay attacks. Fig. 6.1 shows an illustration of the enrolment process and chip authentication.

To keep data secure during transmission, it is typically encrypted using a key that is stored externally, or in an on-chip non-volatile memory (NVM). Unfortunately, storing the key off chip or in an on-chip NVM facilitates the recovery of the key by other parties. Indeed, several studies have shown that NVM are prone to attacks and easy to read out [95], [96]. PUFs replace the conventional key storage, and hence offer superior robustness against invasive attacks, as they do not store information but rather recreate the keys when the chip is being powered on.

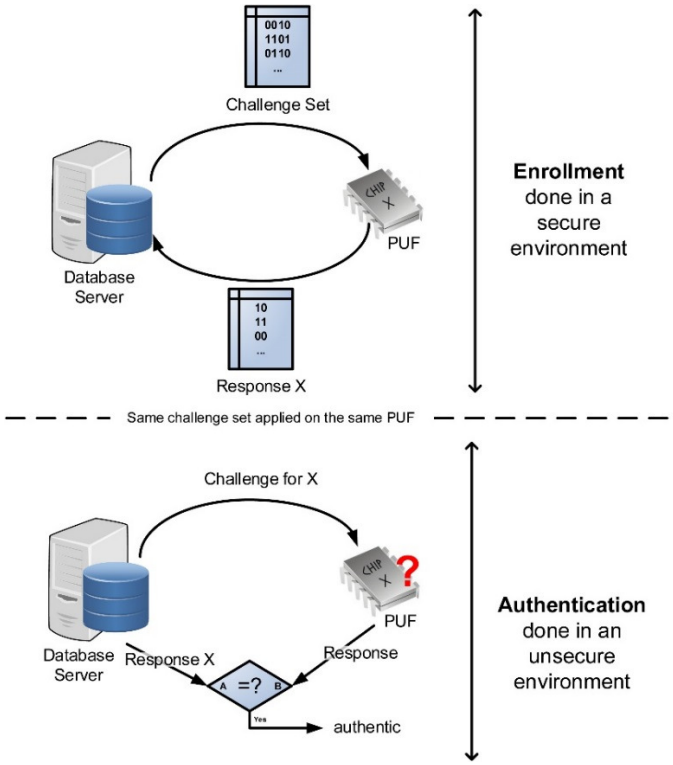


Fig. 6.1 Illustration of typical chip enrolment and subsequent in-field authentication using challenge-response pairs (CRPs) from PUFs.

6.2 PUF Properties and Metrics

Ideally, an array of PUF bitcells generate chip-specific keys that are:

- unpredictable, leveraging on on-chip random process variations
- repeatable, by amplifying random variations, while rejecting global variations and noise [85]
- not directly accessible or measurable externally, once the enrolment phase is completed.

There are two main types of PUFs: weak PUFs and strong PUFs. Weak PUFs have limited number of challenge-response pairs, making them equivalent to random key generators that are typically used for encryption and decryption. Weak PUFs essentially provide chip ID, whereas strong PUFs offer a very large number of

challenge-response pairs (CRPs) for one-time CRPs. Considering the expected long lifespan required by different applications, PUFs with very large number of CRPs (and therefore large area) are very expensive and typically infeasible. As a numerical example, Table 6.1 shows an example of the cost for a PUF with 256-bit key in 65nm [90], [97], assuming 5 cents/mm^2. As can be seen, the cost of the PUF exceeds the typical target of $1/chip.

Table 6.1 Example of SRAM PUF Silicon Cost

(encrypted) data transmitted every	**PUF Capacity (MB)**	**PUF area (mm^2)**	**silicon cost (US$)**
1 hour	5	24	1.2
10 minutes	32	147	7.4
1 minute	320	1,478	74

Table 6.2 PUF Metrics and Typical Values

metric	**measured by**	**typical value**	**ideal value**
stability	Unstable Bits	1 - 60%	0
repeatability	Intra-PUF FHD	0.8 - 15%	0
uniqueness	Inter-PUF FHD	30 - 60 %	50%
identifiability	Inter/Intra HD	5 - 80	∞
randomness	0/1 Bias	40 - 60 %	50%

Given the fundamental PUF properties, such as stability, repeatability, uniqueness and randomness [94], and knowing the statistical nature of process variations, several metrics have been introduced to quantify the quality of PUF bitcells. In the following, such metrics are summarized in Table 6.2, where typical values based on current literature are also reported. In detail, any PUF output should ideally remain the same under fluctuating environmental conditions (e.g., voltage, temperature), and at any process corner. Actual PUFs are not able to provide perfectly stable outputs, due

to non-perfect rejection of noise, global and environmental variations. Stability is measured by counting all bits that become unstable across repeated PUF evaluations and environmental conditions, within the specified range of voltage and temperature of operation.

Repeatability (or reproducibility) and uniqueness are measured from the Hamming Distance (HD) across several measurements of PUF keys. Such measurements are compared to a reference "golden" key [94] that is taken as the first measurement under nominal conditions. Repeatability is the average intra-PUF Hamming Distance (HD) between the golden key and several key evaluations with the same challenge in the same chip, under different environmental conditions. Highly reproducible PUFs should have low intra-PUF Hamming distance (ideally zero). Uniqueness, on the other hand, is taken as the average inter-PUF HD between the golden key and key evaluations from different chips under the same PUF input [84]. The inter-PUF HD should be close to the ideal value equal to half the length of the PUF key (e.g., the ideal inter-PUF HD of a 256-bit key is 128). Alternatively, the fractional Hamming Distance (FHD) can be used to quantify reproducibility and uniqueness [85], where the Hamming distance is simply expressed as a percentage of the key length, or the number of bits N in a PUF key (ideal inter-PUF FHD is 50%). Identifiability quantifies the distinguishability of a PUF instance to other instances, and is loosely taken as the ratio of the inter-PUF and intra-PUF HD (on the assumption that it is both repeatable and unique), where a larger value is desired [84], [94], [98].

Fig. 6.2 shows an example of probability distribution function of reproducibility (intra-PUF FHD) and uniqueness (inter-PUF FHD). A perfectly identifiable PUF ideally has no intersection between the inter-PUF and intra-PUF curves, which means that a single PUF response is enough to determine whether the chip is authentic or not. In practical cases, the two curves in Fig. 6.2 have an intersection, and an optimal decision threshold needs to be chosen to determine whether a given PUF is identifiable.

Such decision threshold is set by the point where Type I and Type II errors are minimized. Type I error is the false positive, where an invalid key is accepted as a valid one. Type II error, on the other hand, is the false negative, where a valid key is discarded as an invalid one.

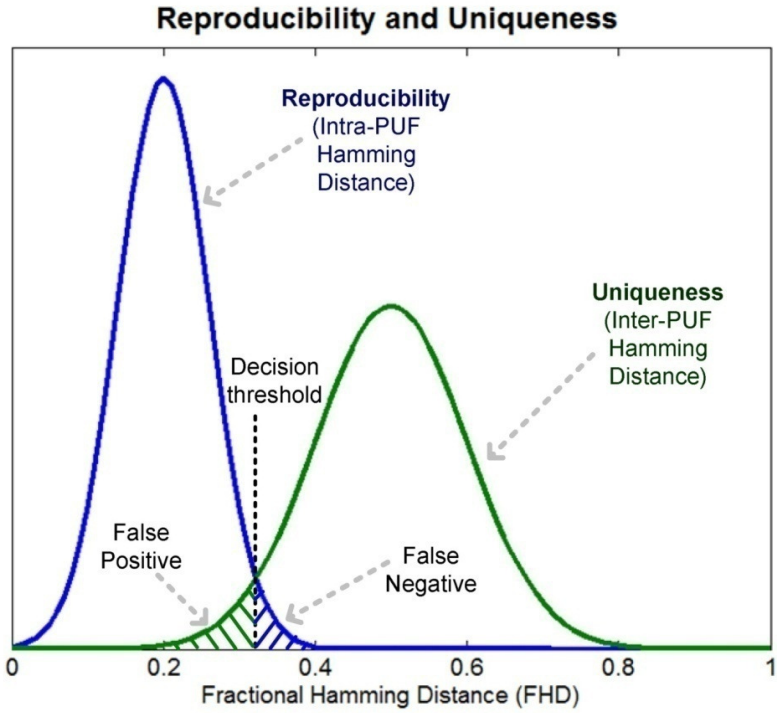


Fig. 6.2 Sample Inter- and Intra-PUF FHD showing decision threshold and Type I (false positive) and Type II (false negative) errors.

Regarding chip authentication, false rejection rate (FRR) and false acceptance rate (FAR) can be used as relevant metrics to assess its quality and security level [99], [100]. Referring to Fig. 6.2, FRR corresponds to the probability of having an output with FHD under the false negative area, whereas FAR corresponds to the area under the false positive area. Accordingly, the PUF yield Y can be defined as the probability of correct authentication, i.e. $Y = 1 - FAR - FRR$. The bit error rate (BER) or the percentage of unstable bits can also be used as a metric of the quality of chip authentication, when the whole array is considered, rather than dividing the array into keys of length N.

Another important property of PUFs is the randomness of its responses, as needed to ensure their unpredictability. Randomness is routinely quantified through the statistical characterization in terms of 0/1 bias (defined as the probability of having a 1 in a PUF output bit [101]), the entropy [84], and more thoroughly through the NIST randomness tests [102]. The NIST statistical test suite [102] is a set of tests to quantify the randomness of a stream of bits. Version 2.1.2 contains 15 tests, each one exercising one property to test randomness. The simplest of these tests is the frequency test, which computes the 0/1 ratio of the whole bitstream. For each of the tests, certain parameters need to be preliminarily set (e.g., length of bitstream n, block size M). Table 6.3 shows the complete list of the tests and parameters to be set.

To quantify the randomness of PUF responses across different positions of PUF bitcell within the die, the autocorrelation function (ACF) is routinely used to detect repeating or correlated patterns among different responses [84], [98]. The correlation between PUF output bits is generally due to layout-dependent variations [90], [97], [103]. Visually, randomness can be represented in the form of the speckle diagram shown in Fig. 6.3, where each pixel represents a PUF bitcell and the PUF output 0's (1's) are represented with black (white) pixels. From the figure, the distribution looks somewhat random (i.e., there are no clear patterns) and the 0/1 bias is also close to ideal value of 0.5.

Fig. 6.3 Sample speckle diagram [97]

Table 6.3 NIST Statistical Test Suite

NIST test	**description**	**minimum stream length *n***	**other parameters**
Frequency Test	takes ratio of number of 1's and 0's	100	--
Frequency test within a block	ratio of 1's and 0's with M-bit block	100	$M \geq 20$ $M > 0.01n$
Runs test	relative oscillation of bit stream	100	--
Longest Run of Ones	length of longest consecutive 1's with a block	128	M (set based on preset n)
Binary Matrix Run	rank of disjoint sub-matrix	$38 \cdot M \cdot Q$	M, Q
DFT	detect periodic features	10^3	--
Non-overlapping Template	detect occurrence of patterns in an m-bit window	10^6	m = [2,10]
Overlapping Template	detects occurrence of patterns, with overlaps included	10^6	m = [2,10]
Universal Statistical Test	number of bits between matching patterns	387,840	L = [6,16]; $Q = 10 * 2^L$
Linear Complexity Test	length of equivalent LFSR	10^6	M
Serial Test	detect frequency of overlapping patterns	--	$m < log_2 n - 2$
Approximate Entropy	detect frequency of overlapping patterns	--	$m < log_2 n - 5$
Cumulative Sums	random walk	100	--
Random Excursions Test	random walk cycle	10^6	--
Random Excursions Variant Test	deviations from a random walk	10^6	--

Most devices nowadays are tightly energy constrained, being either battery operated or energy harvested, hence the energy consumption of the PUF is another important metric. To abstract the energy from the PUF organization and size, the most commonly adopted metric is the energy per bit, obtained by dividing the average energy

per access by the number of bits within the key. The energy per bit of existing PUFs typically ranges from tens of fJ/bit to tens of pJ/bit [90], [97].

Silicon cost in terms of area is another important PUF metric. The effective area per bit (obtained by considering the actual number of available PUF bits obtained after removing unstable bits, and including the area cost of the circuitry performing post-processing on the raw PUF output) may be used as a metric. Robustness to ageing and chip lifetime are assessed through accelerated ageing tests [99], [104], [105]. Modelling complexity, in terms of the number of brute force trials needed to model the PUF, can likewise be used to characterize PUFs [99].

6.3 PUF Topologies and State of the Art

The concept of PUFs have been introduced in the early 2000s, and they have been initially referred to as ICID [106], Physical One-Way Functions (POWF) [107], or Physical Random Functions (PRF) [91], among others. ICID uses an array of MOSFET to generate the random values from random process mismatch, via FET drain current. The physical one-way function was proposed as a solution to the need for a one-way function (easy to evaluate but difficult to invert) for cryptographic applications. The approach uses a laser to scatter light through an inhomogeneous structure (at some precise angle, which serves as the challenge), as shown in Fig. 6.4. The resulting optical speckle diagram is hashed to obtain the key. Most of the literature has then reverted to silicon-based solutions, leveraging the low-cost and high-volume capability of CMOS chips.

Most of the existing silicon PUFs can be classified as either delay-based or memory-based PUFs [91], [108]–[110]. In delay-based PUFs, bits are generated by comparing the delay of two nominally identical paths. The sign of the random delay difference between the two delays determines the output bit. One of the earliest implementation of such a concept is the ring oscillator (RO) PUF [91], [109], whose

digital output is determined by the relative frequency of each selected pair of nominally identical ring oscillators. Fig. 6.5a shows the general diagram of a ring oscillator PUF, where the challenge selects two of the available ring oscillators, and the corresponding response depends on whether the frequency of the first selected oscillator is greater than the second or not. Knowing that these inverter chain ring oscillators tend to be very sensitive to environmental conditions, several techniques have been introduced to improve the high native instability rate, and poor statistical quality of this pair-wise comparison. Some of these techniques include the adoption of k-sum or 1-out-of-k masking techniques [109], [111].

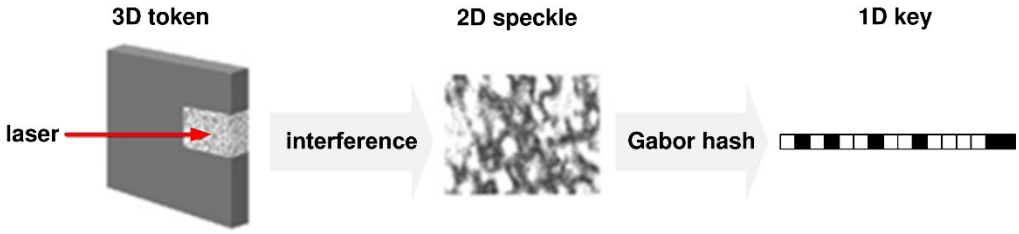


Fig. 6.4 Physical One-Way Function from a non-homogenous material

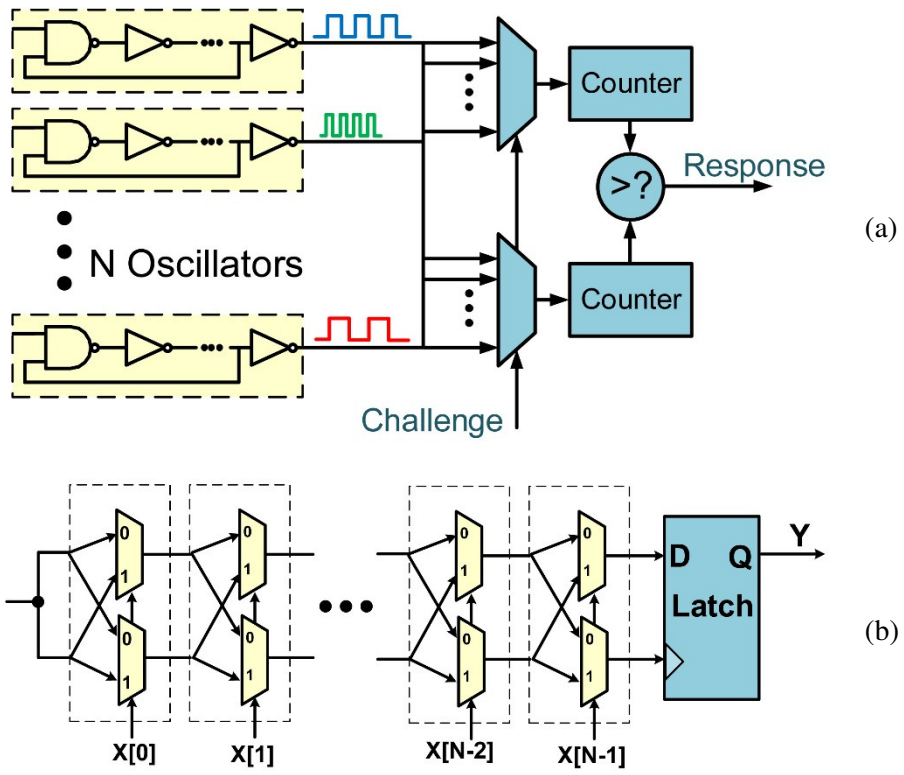


Fig. 6.5 Delay-based PUFs: (a) ring oscillator (RO) PUF [109], (b) arbiter PUF [111]

Another delay-based PUF is the arbiter PUF [100], [109], [111], shown in Fig. 6.5b. It compares the delay of two delay lines, and suffers from the same limitations as the RO PUF [91], [111]. An improved delay-based version was recently proposed [98], based on the oscillation collapse in an even-stage ring of delay-adjustable stages. The delay is set by an applied input (PUF challenge) via inverter replica multiplexing. The native instability of PUF outputs was substantially reduced at the cost of much higher energy and the need for CTAT biasing.

All above delay-based PUFs are also intrinsically vulnerable to PUF modelling attacks, which can capture and clone the content of the entire PUF with very low effort. Indeed, the PUF output is dictated by the overall PUF delay, which is in turn defined by the sum of the delays of cascaded stages. Since each stage delay is fixed (although unpredictable), identifying all stage delays from the analysis of the PUF outputs entails only a linear complexity, making the PUF easy to clone [112].

In memory-based PUFs, a bistable structure of two cross-coupled inverters is used to generate the output bits. They leverage on the natural tendency of cross-coupled inverters to resolve to a preferred state at the power-up, as determined by their asymmetry due to random variations [109]. For example, SRAM PUFs leverage this property in SRAM bitcells [108], [113]. Other similar PUFs are the Latch PUF [88], DFF PUF [114], butterfly PUF [110], and the buskeeper PUF [115], which is similar to the SRAM PUF albeit without the write ability, as access transistors are removed since PUF bitcells are read-only. The butterfly PUF follows the same concept of leveraging on the unstable state of cross-coupled inverters. It was proposed for implementation in an FPGA and uses the available cross-coupled latches instead of inverters, as shown in Fig. 6.6. The operation starts by asserting the *excite* signal, thereby forcing the PUF to be in the unstable state. This signal is then released and after

a few clock cycles, *out* signal settles to its natural stable state determined by the random variations in the related logic gates.

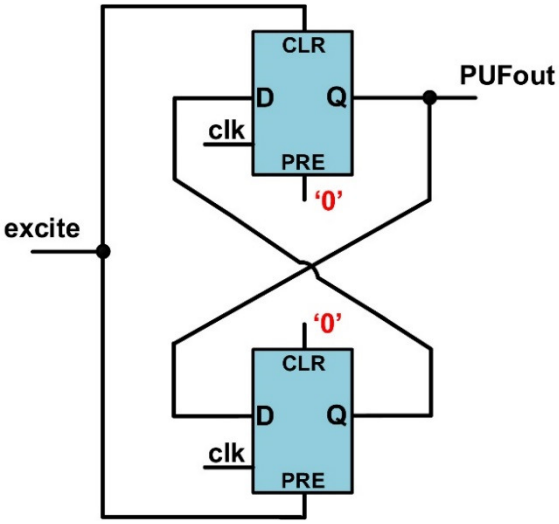


Fig. 6.6 Butterfly PUF [110]

A recent literature on memory-based PUFs and their experimental characterization has shown that PUFs typically have poor stability [116], and are highly vulnerable to semi- invasive attacks such as electrical and optical probing [89]. The same vulnerability to semi-invasive attacks is found in other PUFs that rely on the same principle, such as senseamp [117], [118]. For such PUFs, reasonable levels of stability are typically achieved through substantial temporal redundancy at the expense of energy consumption [119]. Other proposed PUFs are based on:

- the glitch generated in digital paths, although they suffer from high instability rates [120]
- the difference in leakage current generated by nominally identical transistors, although at the cost of large energy due to the necessary circuitry for current/voltage references and opamp [121]
- DRAM errors under different wordline voltage, although such PUFs are highly vulnerable to non-invasive attacks [87]
- the variations in supply network resistance, although this requires the generation of very large currents [119]

- open or short connection in vias [122]
- oxide breakdown in OTP ROMs [123]
- capacitance mismatch [124]–[126]

A hybrid PUF was proposed in [84], [127], [128], combining delay and metastability as sources of randomness. The basic bitcell is shown in Fig. 6.7, where bistability is forced through the pre-charge transistors controlled by *clk0* and *clk1*. The randomness in delay is introduced through the clock skew between *clk0* and *clk1*. To reduce unstable bits, significant temporal majority voting is employed. Soft dark bit masking was also used in [127] by modulating the load in the *bit* and *bit'*, and masking bits that become unstable with the change in the load. Indeed, load modulation simply injects controlled perturbation in the stability of the PUF bitcell, which in turn permits to identify the truly stable bitcells that do not change output even in the presence of such perturbation.

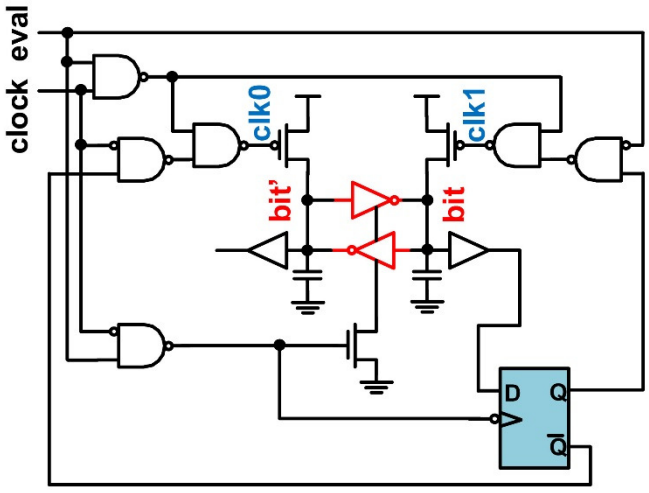


Fig. 6.7 Metastability-based PUF [84]

To achieve adequate native stability despite voltage and temperature fluctuations, authors in [103] proposed to use a proportional-to-absolute-temperature (PTAT) as a bitcell. Fig. 6.8 shows the bitcell and the architecture and principle of operation of the PTAT-based PUF. As seen in the figure, the PUF bitcell output is

determined by the sign of the difference between *Out_l* and *Out_r*, both of which are independent of voltage and temperature. Aside from the high resiliency against voltage and temperature variation, another noteworthy feature of this PUF is its good area efficiency (only $727F^2$/bit, being F the minimum feature size of the adopted process), as enabled by the shared header per column.

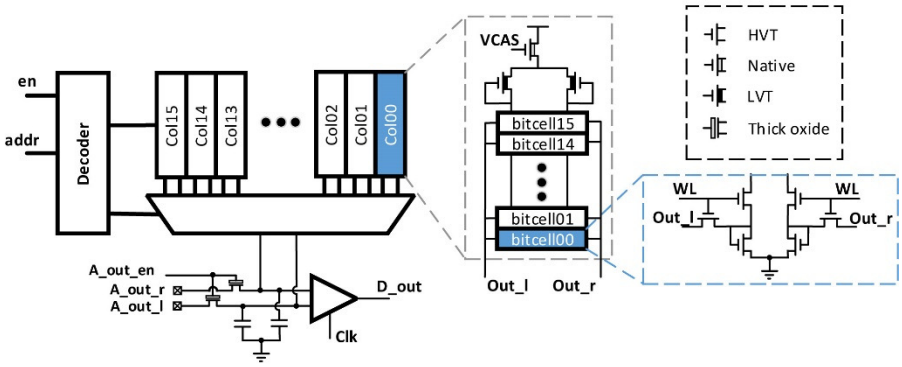


Fig. 6.8 PTAT-based PUF [103]

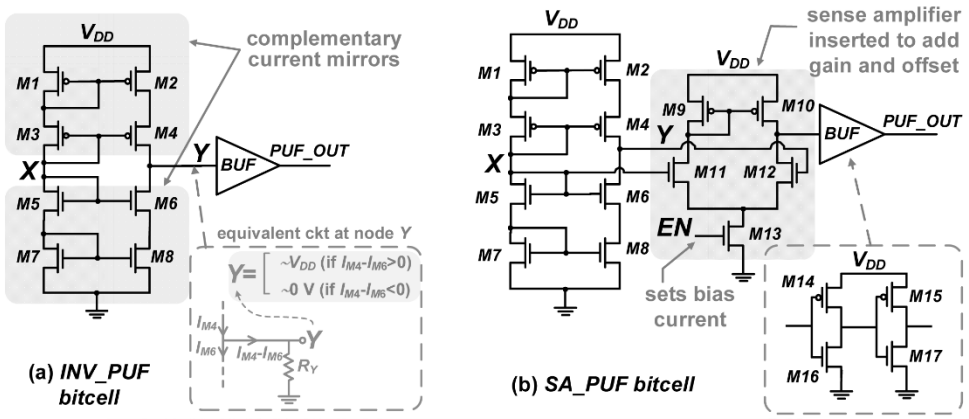
6.4 Static, Monostable PUFs

In this section, we present our proposed novel class of static mono-stable PUFs [90], [97] for extremely low energy operation and low native instability rate, which relies on the amplification of random transistor mismatch through two complementary current mirrors.

6.4.1 Design and Operation

Fig. 6.9 shows two implementations of the same general concept. Fig. 6.9a shows the INV_PUF bitcell implementation of this concept, which comprises the cascode current mirrors M1-M4 and M5-M8. The two 1:1 current mirrors see the same current flowing through their respective input transistors (M3 and M5), and tend to mirror it to their output transistors (M4 and M6, respectively). Without mismatch, M4 and M6 would conduct the same saturation current ($I_{M4,SAT}$ and $I_{M6,SAT}$), and node Y would assume the same voltage as node X (e.g., $V_{DD}/2$), due to the symmetry of the topology in Fig. 6.9a. However, random mismatch between M1-M2 and M7-M8 makes

these currents unpredictably different. The large output small-signal impedance R_Y at node Y (Fig. 6.9a) translates the difference in such currents into a large voltage deviation. Accordingly, the voltage at node Y becomes essentially V_{DD} if $I_{M4,SAT}$- $I_{M6,SAT}$>0, or ground if $I_{M4,SAT}$- $I_{M6,SAT}$<0. Thus, a digital output that is dominantly defined by the random mismatch between the two current mirrors is generated.



Transistor	Width (nm)	Transistor	Width (nm)
M1-M4	550	M13	500 x 2
M5-M8	80	M14	340
M9-M10	80	M15	365
M11-M12	80	M16-M17	80

(c)

Fig. 6.9 Static Mono-stable PUFs [97] (a) INV_PUF and (b) SA_PUF

In statistically infrequent cases with extremely small mismatch, $I_{M4,SAT} - I_{M6,SAT}$ might still be close enough to zero even under random variations, so that the voltage deviation at node Y is not full swing, and the PUF output bit *PUF_OUT* becomes unstable. However, the percentage of such unstable bits will be shown to be very small, as expected from the large impedance at node *Y*. As an example, Fig. 6.*10* shows the statistical distribution of the voltage at node X, Y, and *PUF_OUT* due to the random mismatch, and shows that the voltage in X is rather insensitive to mismatch, whereas Y is very sensitive to it, and its voltages are mostly associated with digital high and low level (as defined by the traditional low- and high-input thresholds of the subsequent buffer). The infrequent intermediate voltages are associated with unstable bits, as discussed above. Similar considerations hold for *PUF_OUT*. In Fig. 6.9b, the

alternative SA_PUF topology adds a sense amplifier (transistors M9-M13) after M1-M8 to further increase the voltage gain (and thus reduce the number of unstable bits) and introduce additional random mismatch through the sense amp offset.

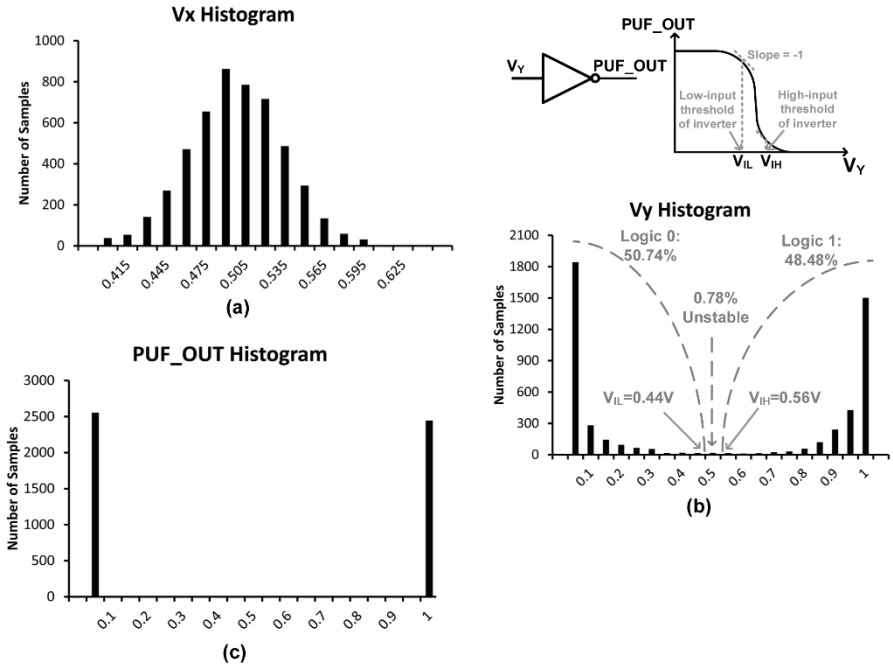


Fig. 6.10 Sample statistical distribution of (a) V_X, (b) V_Y and (c) PUF_OUT under variations (5k-pt Monte Carlo simulation)

To verify the functionality and effectiveness of these PUF bitcells, a PUF array with 3,040 bits was implemented in 65nm CMOS technology. The transistor sizes used are shown in Fig. 6.9c, and the chip photomicrograph and the layout of the bitcell macro are shown in Fig. 6.11. Outputs are scanned out via a latch-based scan chain. The latch scan chain effectively implements a latch PUF, when isolating the INV_PUF and SA_PUF output from the scan chain through a multiplexer between the PUF output and the scan chain input. As in Fig. 6.11, when *LOAD* is set to 1 at the power up, intrinsic power-up state of latches is routed through *INV_SCAN_IN* and *SA_SCAN_IN* to the latch, instead of the PUF output. In this way, the first 95x32=3,040 output bits will correspond to the Latch_PUF output. Setting the signal *LOAD* to 0 uses the output of the PUF as input to the latch. Subsequently changing *LOAD* to 1 forms the scan chain, allowing these data to be scanned out serially.

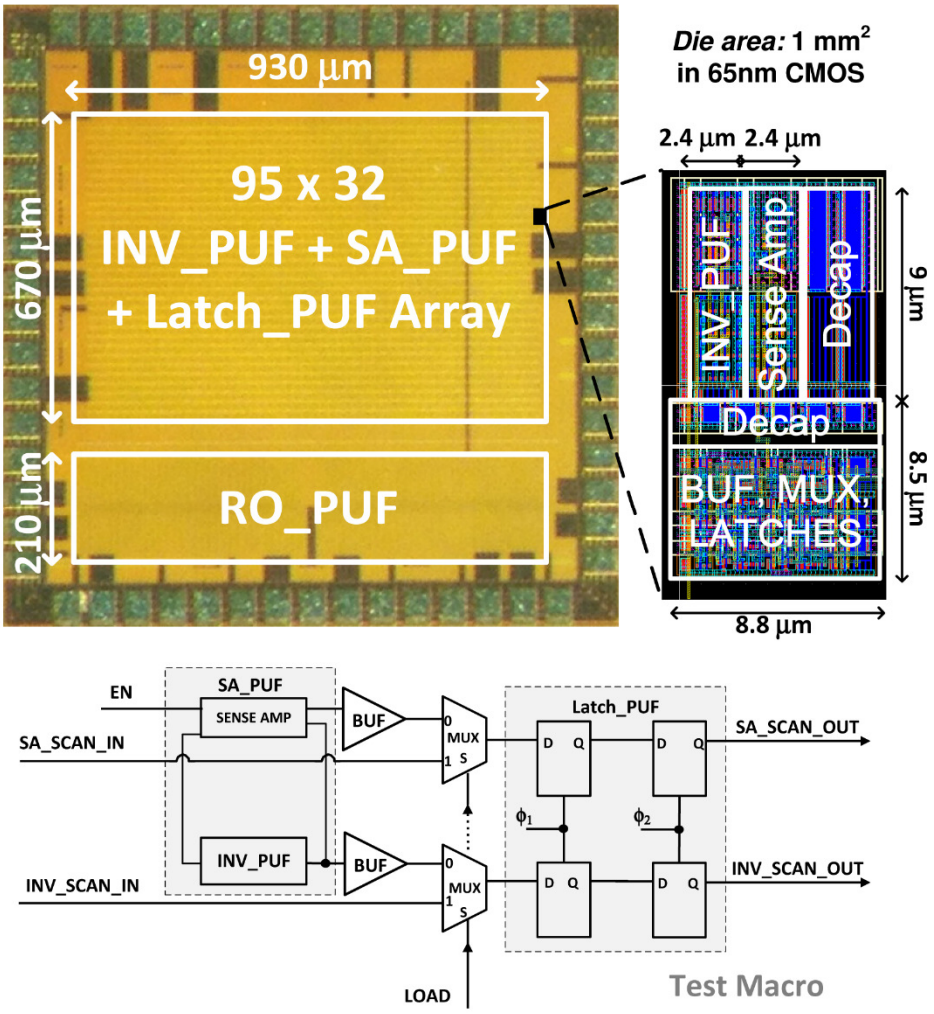


Fig. 6.11 Chip photomicrograph, Bitcell Layout and Test Macro Schematics

The testchip also includes a ring oscillator PUF (RO_PUF) to allow fair comparison at iso-technology. The design of the RO_PUF is similar to [109], and consists of 5-stage ring oscillators. The inverters within the chain are 3-stack HVT transistors, each with a size of 450nm/60nm (300nm/60nm) for the PMOS (NMOS). The driving inverter uses LVT transistors of same size, without stacking. The architecture of the RO_PUF is shown in Fig. 6.12, where the two-hot decoder ensures that 2 of the 1024 ring oscillators are selected at a time. The frequency dividers are programmable (divide by 8/16/32/64) to evaluate the difference between the frequencies of the selected ring oscillators. The selector signals *SA<9:0>* and *SB<9:0>* serve as PUF challenge inputs to choose the specific ring oscillator output to be routed

to *freq_out1* and *freq_out2* outputs, respectively. These outputs in turn are compared to get the RO_PUF output bit. Overall, the testchip permits to compare the two versions of the proposed PUF with latch and ring oscillator PUF sharing the very same die. Fair comparison with SRAM PUF is enabled by the measurements on an available SRAM array [129] implemented in the same technology.

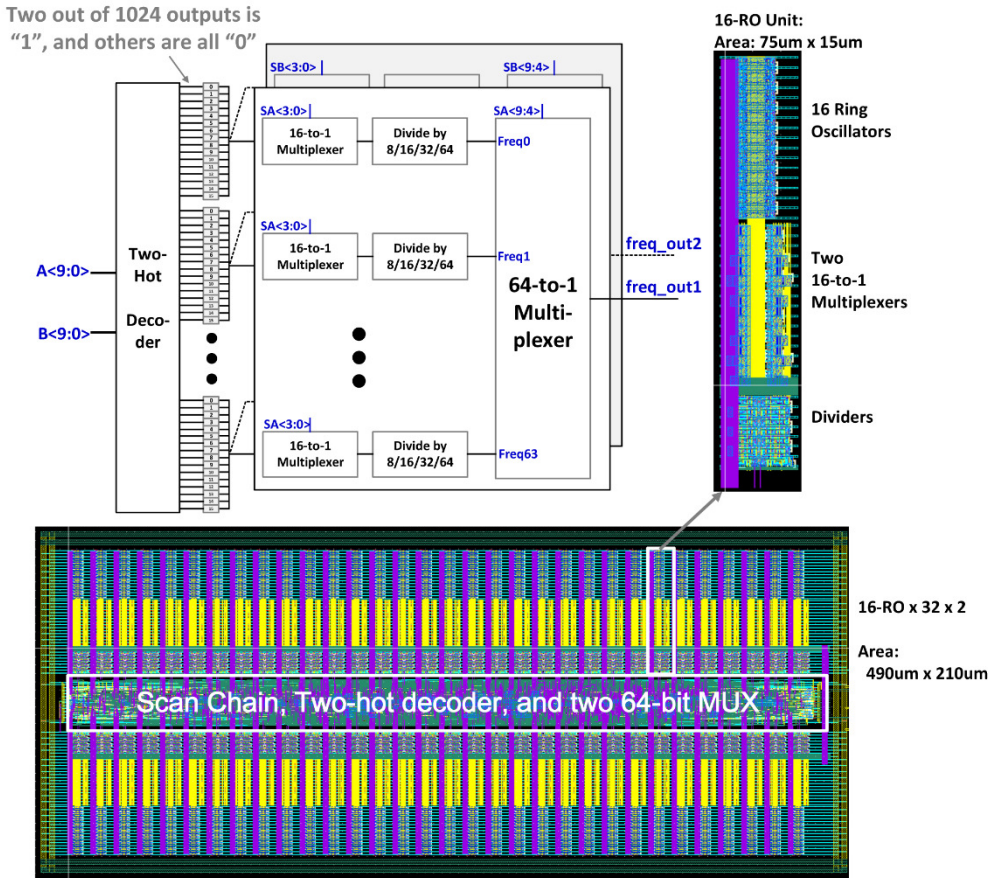


Fig. 6.12 RO-PUF Architecture and Layout

6.4.2 Testchip Measurement and Comparison

For all stability tests, 400 evaluations were performed per configuration, marking bits that change at any evaluation as unstable. The measured percentage of the native unstable bits at nominal condition (1V supply, 25°C temperature) is plotted in Fig. 6.13 versus the number of evaluations. From this figure, INV_PUF exhibits native bit instability of only 2.34% (for the most unstable die), and the SA_PUF offers a further improvement (1.88%), as expected from the voltage gain amplification and the

additional random variations introduced by the senseamp in Fig. 6.9b. As shown in Table I, the native bit instability of the proposed class of PUFs is an order of magnitude lower than all other PUFs, whose instability ranges from 16.66% (SRAM_PUF) to more than 30%, which is just slightly higher, but still consistent with previous reported results [127], [130], [131]. Note that for the RO_PUF was designed for 0.5 V operation and tests were performed at this voltage, as instability becomes even worse above this voltage. Interestingly, the proposed PUF achieves a native stability that is comparable to one of the best techniques [84], although the latter needs the joint adoption of three enhancement techniques that worsen energy efficiency (temporal majority voting), testing cost (burn-in) and area efficiency (unstable bit masking). In other words, the proposed PUF class is intrinsically more robust and trustworthy than all above PUFs, and can be made 100% stable through very limited design/tuning effort (e.g., by masking unstable bits to predefined values as in [84]).

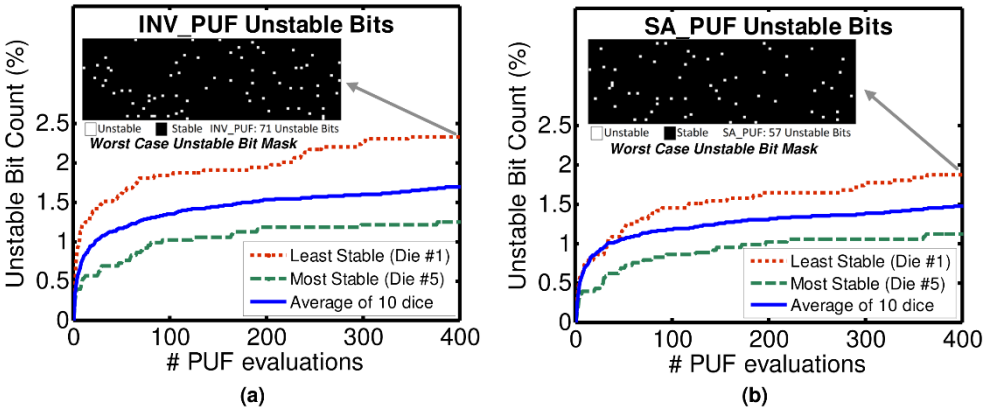


Fig. 6.13 Native Unstable Bit Count at Nominal Conditions for (a) INV_PUF and (b) SA_PUF

Table 6.4 Comparison of Different PUFs

PUF	Latch [88]	Arbiter [100]	ICID [106]	RO_PUF*	SRAM_PUF*	[84]	[98]	PTAT [103]	INV_PUF	SA_PUF
Technology	0.13um	0.18 um	0.35 um	65nm	65nm	22nm	40nm	65nm	65nm	65 nm
Stability (% native unstable bits at nominal condition)	3.04	9.8	**1.3**	18.16	16.66	30	12.5	7.1	2.34	1.88
V-T Variation	0.9-1.2V	1.8V +/-2%, 27-70C	1.1-5V, -25-250C	0.4-0.5V	0.6-1V	0.7-0.9V	0.7-0.9V		0.6-1V, 25-85C	0.6-1V, 25-85C
%error with VT variation	5.46875	4.82	5	53.9	55.73	30	12.5		5.72	5.33
Energy (pJ/bit)	0.93	0.17125	8333.333	0.475	1.1	0.19	17.75	1.1	**0.015**	0.163
Area (F^2/bit)	4369	708403	1708	39000	806	9628	2062	**727**	6000	12000
Randomness				0.5023	0.6141	0.4805		0.4928	0.5016	**0.5005**
Uniqueness (Mean Inter-PUF FHD)	0.5055	0.3800	0.4911	0.4738	0.3321	0.5100	0.5007	**0.5001**	0.5014	0.5015
Repeatability (Mean Intra-PUF FHD)			0.0134	0.0458	0.0602	0.0268	0.0101	0.0057	**0.0034**	**0.0034**
Identifiability (Inter-PUF/Intra-PUF FHD)			37	10	6	19	50	88	**149.05**	148.68
NIST Test				PASS	PASS	PASS	PASS	PASS	PASS	PASS
entropy				0.995	0.99	1		**1**	0.9967	0.9966
Autocorrelation Function @95% confidence				0.088	0.016	**0.01**	0.028	0.019	0.0363	0.0363

* fabricated with the same technology as INV_PUF and SA_PUF

For completeness, the stability for the SA_PUF was also studied as a function of the bias voltage of the senseamp tail *EN* in Fig. 6.9b, which was set at $V_{DD}/2$ for most tests. Fig. 6.*14*a plots the measured and simulated stability versus the *EN* voltage, and shows that this bias voltage does not affect the stability of the PUF, except for extremely low voltages. Indeed, the unstable bit count remains essentially constant for *EN* voltages ranging from 0.25 V to 0.75 V (Fig. 6.*14*a), the spatial distribution of 0's and 1's is random, and bias (Pr[1]) is close to 0.5 (Fig. 6.*14*c). Instead, very low *EN* voltages below 0.25 V degrade the stability (up to 22%) and the distribution of 0's and 1's (bias of 0.94) under the extreme case of grounded *EN* terminal (Fig. 6.*14*d). This is because a very low *EN* voltage (e.g., below 0.25 V) essentially turns the tail transistor off, and the senseamp pulls the output node of SA_PUF towards V_{DD}, effectively introducing an undesirable systematic offset (Fig. 6.*14*b).

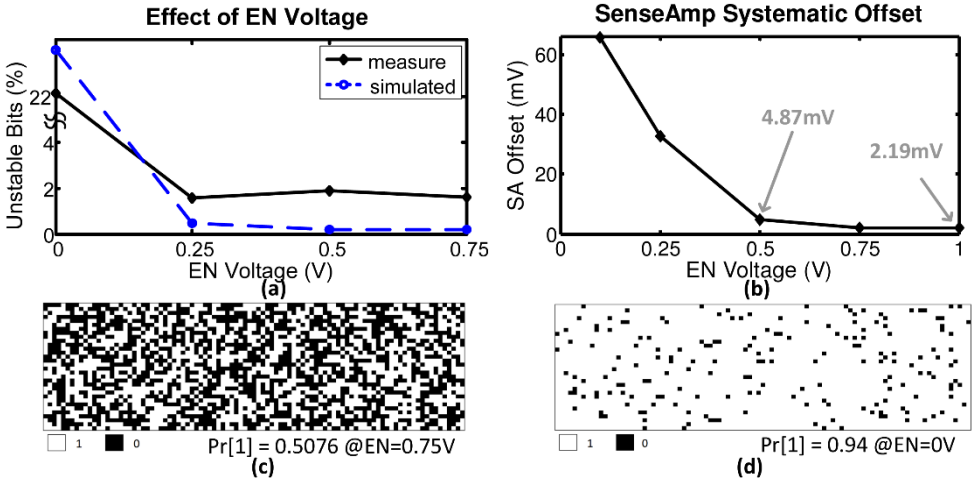


Fig. 6.14 SA_PUF dependence on EN voltage in terms of (a) unstable bit percentage and (b) senseamp systematic offset, and resulting speckle diagram at (c) EN=0.75V and (d) EN=0.

The resilience to PVT systematic and environmental variations was evaluated by measuring stability across different voltages, temperatures and different process corners. Regarding the effect of voltage on the bit stability, Fig. 6.15a shows that the native stability of the proposed class of PUFs is degraded only slightly (3.64% and 3.53% for INV_PUF and SA_PUF) in the wide voltage range from 0.6 to 1 V. On the

other hand, the stability of the other measured PUFs (latch and SRAM) is severely degraded to 50-70% within the same voltage range. Overall, the proposed class of PUFs enables an order of magnitude improvement in stability, compared to most of the other PUFs. Specifically, a native stability improvement from 10.68X (at 1 V) to 14.11X (at 0.6 V) is found compared to latch and SRAM PUF, and a 9.76X (or better) compared to [84] at 0.7 V (at larger voltages). As only exception where the stability improvement is lower than an order of magnitude, the proposed PUFs achieve a stability improvement of 2.5X compared to [98], which is however three orders of magnitude worse than the proposed PUF in terms of energy (see Table 6.4). Regarding the impact of the temperature on stability, Fig. 6.15b shows that native instability at 85 ºC is 4.4% and 4.3%, respectively for INV_PUF and SA_PUF, which is 10.31X (or better) lower than all other PUFs. Comparing INV_PUF and SA_PUF, the latter is slightly more stable than INV_PUF under temperature variations.

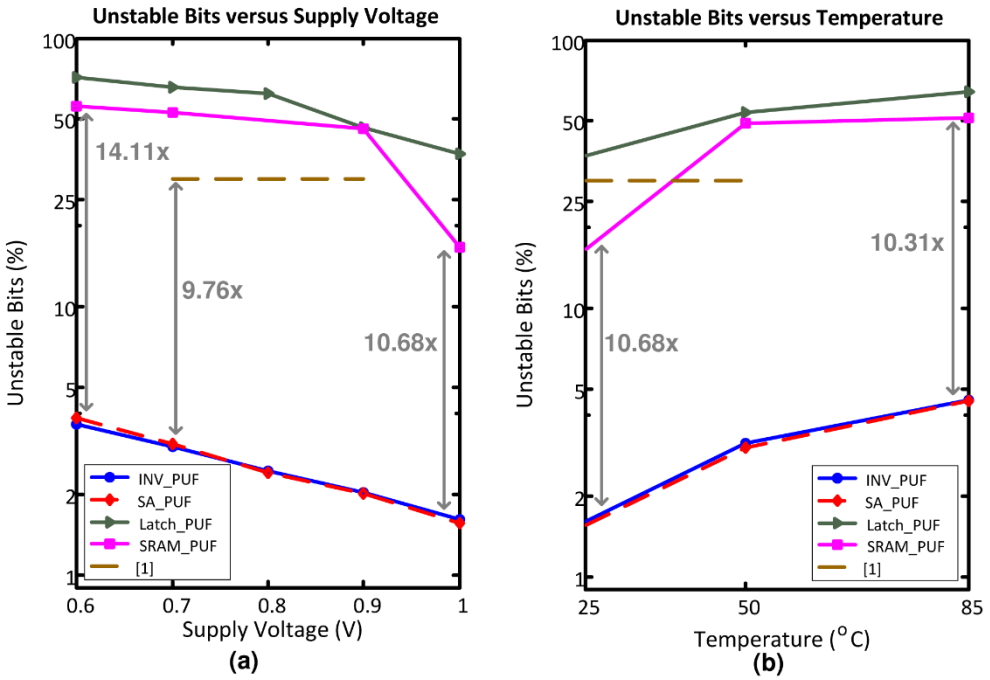


Fig. 6.15 Percentage unstable bit versus (a) supply voltage and (b) temperature for different PUFs

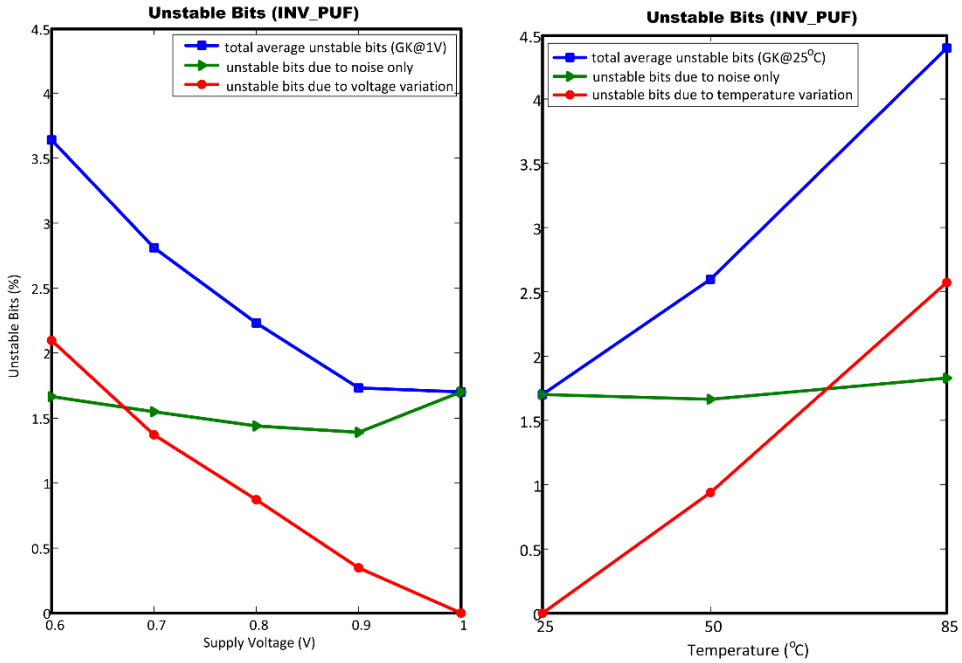


Fig. 6.16 Breakdown of percentage unstable bits in INV_PUF due to supply voltage (left) and temperature (right)

To gain a better understanding of instability, the contributions of on-chip noise, voltage and temperature variations were separately analyzed. In particular, the contribution due only to noise was obtained by evaluating the number of unstable bits across 400 repeated accesses at a given voltage and temperature. The contribution of voltage variations was then obtained by removing the noise contribution (i.e., discarding the unstable bits due to noise alone) and considering the bits that flip when the voltage is reduced below the nominal voltage (1 V). Similar considerations hold for temperature variations. The resulting instability contributions of noise and voltage (temperature) for INV_PUF are shown in Fig. 6.16a (Fig. 6.16b). Results for SA_PUF are just slightly better (~0.033% at 0.6 V and ~0.039% at 85 °C) and were hence omitted. From Fig. 6.16a, the instability due to the noise is approximately constant (~1.7%) across a wide range of voltages and temperatures, and dominates the total instability at relatively high voltages (0.8 V and above). This means that the total unstable bits will be reduced by ~1.7% when applying masking as in [84]. The instability due to voltage variations ranges from 0 to 2.1% when decreasing the voltage

down to 0.6 V, while the contribution of temperature variations ranges from 0 to 2.57% when increasing the temperature to 85 °C.

The same procedure was followed for the other PUFs to mimic the effect of unstable bit masking, as shown in Fig. 6.17 for the Latch_PUF and SRAM_PUF, and Fig. 6.18 for RO_PUF. From Fig. 6.17, the difference between the masked and unmasked data for Latch_PUF ranges from 3.86% to 18.53%, while for SRAM_PUF is 9.11% to 13.82%. For the RO_PUF, the effect of voltage and temperature on the RO_PUF stability, with and without masking, is shown in Fig. 6.18. The data in this figure confirms that the RO_PUF is indeed highly unstable (as is evident in the 100% unstable bits at 0.3 V) and would need additional circuitry to improve its stability. Like the other PUFs, we can also see that the difference between the unstable bit count with and without masking is also high for both change in voltage and change in temperature, further confirming that the RO_PUF is sensitive to voltage and temperature changes. In [84], temporal majority voting was introduced prior to replacing the unstable bits with predefined ones, to reduce the number of unstable bits by 53%.

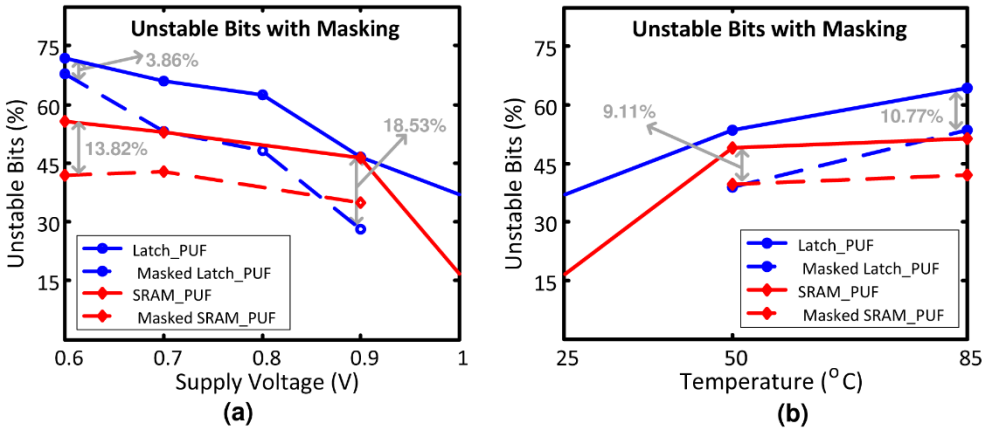


Fig. 6.17 Effect of masking on unstable bits for Latch_PUF and SRAM_PUF with varying (a) supply voltage and (b) tempretaure

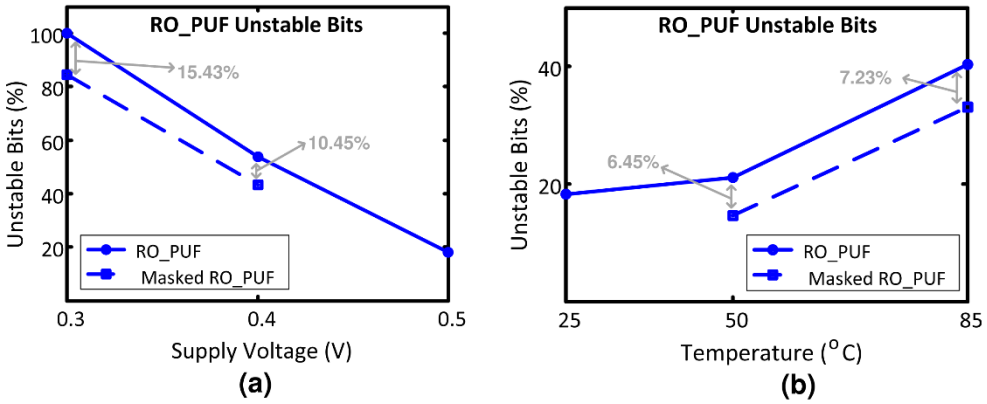


Fig. 6.18 Effect of masking on unstable bits for RO_PUF

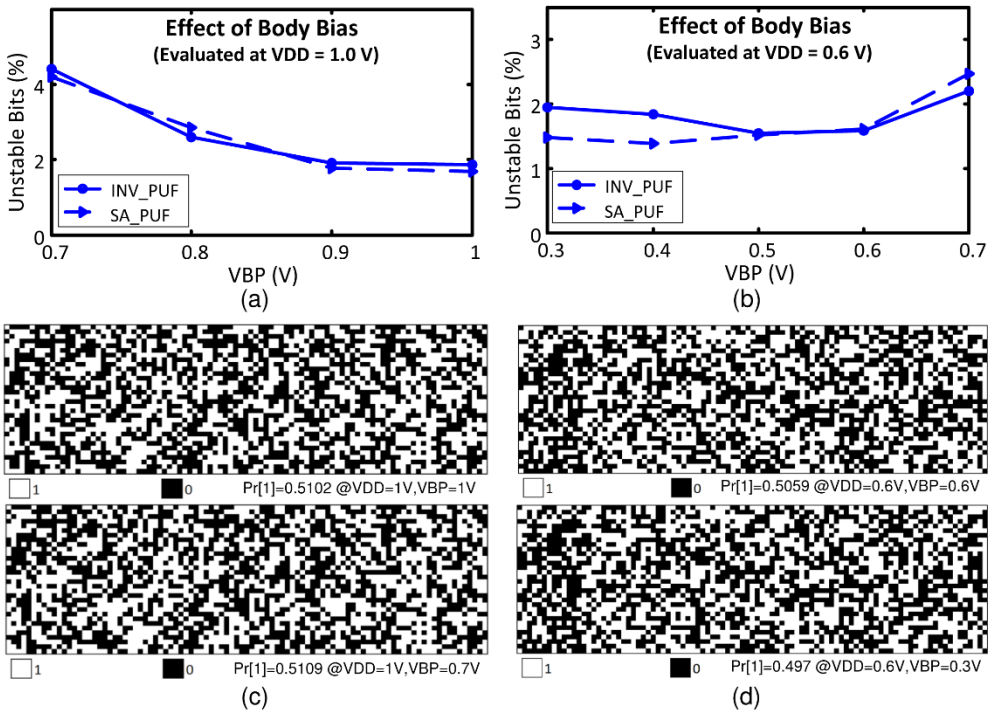


Fig. 6.19 Effect of body bias (to mimic systematic threshold variation) on stability, measured at (a) 1V and (b) 0.6V supply, and the corresponding speckle diagram and bias at (c) 1V and (d) 0.6V supply.

Finally, the impact of systematic transistor variations was studied by modifying the PMOS threshold voltage through body biasing, which permits to introduce a skew between the NMOS and PMOS transistor strength (i.e., emulate different process corners). The instability is plotted in Fig. 6.19 as a function of the PMOS body bias voltage, V_{BP}. From Fig. 6.19a, the percentage of unstable bits is increased to only 4.4% under 300 mV forward body biasing at 1 V supply, which corresponds to a 46-mV

systematic threshold voltage change (i.e., ~3 standard deviations). Similar considerations hold at near-threshold voltages (Fig. 6.19b). The bias is only marginally affected by systematic variations as it changes only by 0.0007 (0.0089) at 1V (0.6V), as shown in Fig. 6.19c (Fig. 6.19d). These results show that the proposed class of PUFs is highly resilient to PVT variations, as stability of a few percentage points is maintained even under 3-σ threshold voltage variations, voltage variations of 0.4 V and temperatures up to 85 °C.

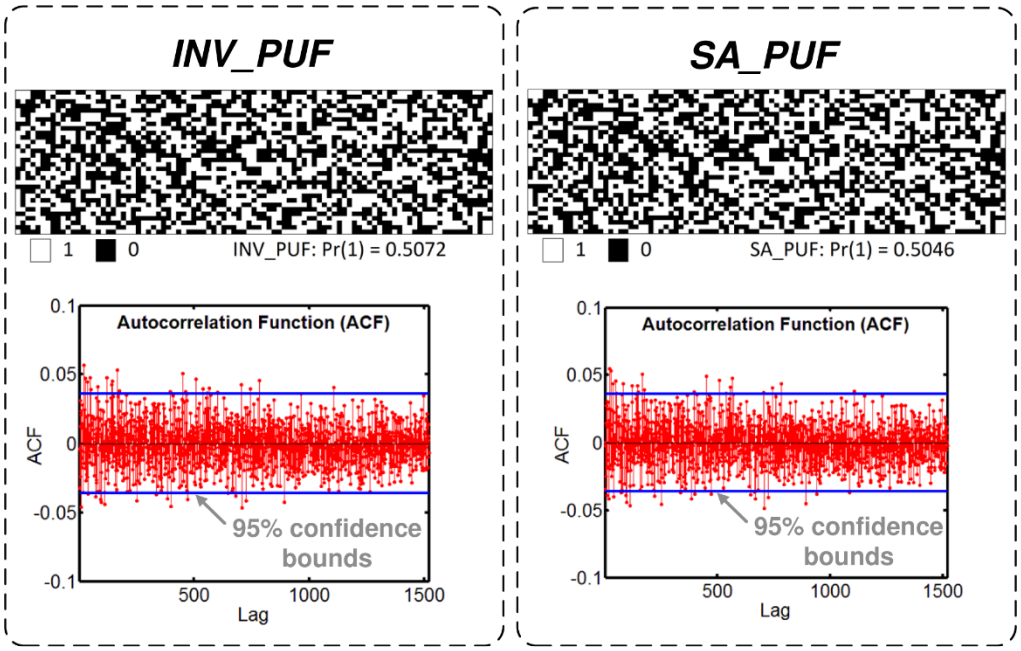


Fig. 6.20 Speckle diagram of the golden key (top) and spatial autocorrelation from die #1 at nominal conditions for INV_PUF (left) and SA_PUF (right)

The measured spatial distribution of INV_PUF and SA_PUF bits is shown in Fig. 6.20, which was obtained from measurements of die #1 of the implemented 65nm PUF array testchip under nominal condition (V_{DD} = 1 V, 25°C). When grouping bitcells in typical 256-bit PUF words [84], the probability of generating a 1 in die #1 is very close to the ideal value of 0.5 (0.5072 for INV_PUF, 0.5046 for SA_PUF). Very similar results are obtained for all other tested dice. The spatial autocorrelation function value at 95% confidence level is close to the ideal value of 0 (less than 0.0363 for both INV_PUF and SA_PUF), showing a fundamental independence of every bit value from

its neighboring bits, and thus confirming effective rejection of layout-dependent variations.

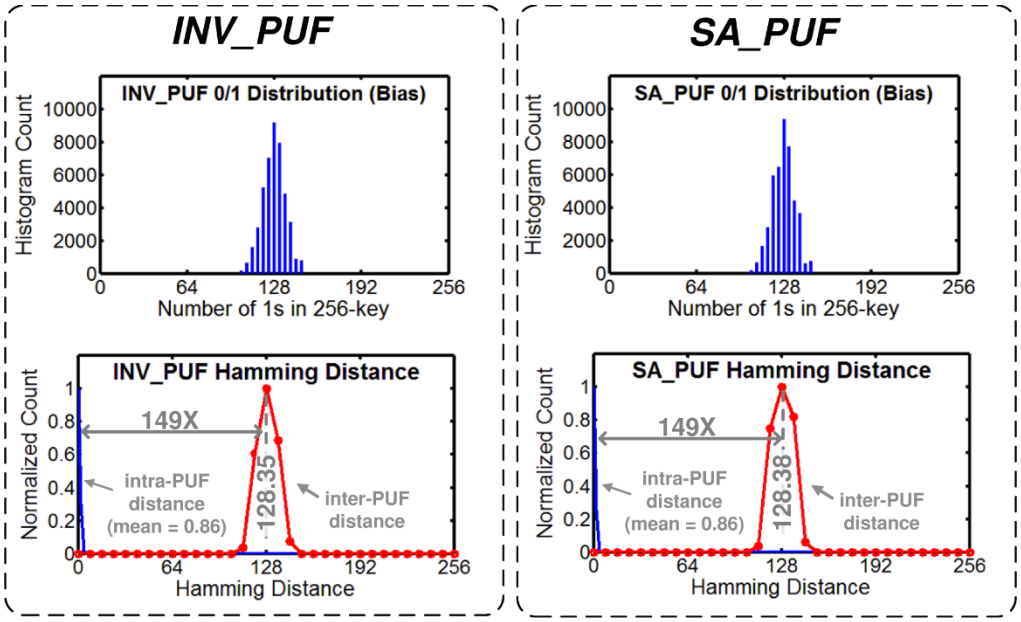


Fig. 6.21 INV_PUF (left) and SA_PUF (right) bias (top) and inter- and intra-PUF HD statistical distribution (bottom).

Regarding uniqueness of the PUF, from Fig. 6.21, the inter-PUF HD across 10 dice has average values of 128.35 for INV_PUF and 128.38 for SA_PUF, which are very close to the ideal value of 128. This confirms that the proposed class of PUFs exhibits very good uniqueness, in the sense that different dice will statistically have largely different values for the same input [85]. As summarized in Table 6.4, the proposed class of PUFs achieves the best uniqueness among most PUFs, being the latch PUF the second best with 126.43 inter-PUF HD. As only exception, the PUF in [98] has a marginally better inter-PUF HD of 128.18.

Regarding the reproducibility, the average intra-PUF HD of the proposed class of PUFs is 0.8611 (0.8635) for INV_PUF (SA_PUF), which is close to the ideal zero value. These values are an order of magnitude better compared to [84], SRAM, latch, and ring oscillator PUFs, and 3X better compared to [98]. Regarding identifiability, the ratio of the inter- and intra-PUF HD of 149X for both INV_PUF and SA_PUF is the

highest to date, and shows 7.8X improvement over [84], 3X improvement over [98] and 20-30X over ring oscillator, latch and SRAM PUFs.

Table 6.5 Summary of NIST Test Results

NIST Test	Stream Length	INV_PUF		SA_PUF	
	n	Average p-value	PASS?	Average p-value	PASS?
Frequency (Freq)	1000	0.56336	YES	0.62002	YES
Block Frequency (BF)	1000	0.38735	YES	0.46658	YES
Runs	1000	0.37511	YES	0.4074	YES
Longest Runs of Ones (LRO)	1000	0.50319	YES	0.28025	YES
FFT	1000	0.70472	YES	0.38435	YES
Non-Overlapping Template (NOT)	1000 (m=5)	0.48719	YES	0.50483	YES
Serial	1000 (m=4)	0.50111	YES	0.50503	YES
Approximate Entropy (AppEn)	1000 (m=4)	0.37291	YES	0.3912	YES
Cumulative Sums (CumSum)	1000	0.38905	YES	0.46648	YES

PUF randomness was quantitatively assessed through statistical NIST tests [102], as more rigorous and systematic approach compared to bias analysis and visual inspection of the golden key mask (Fig. 6.20). For each NIST test, the *p-value* was evaluated to quantify the level of randomness of the PUF. In general, a *p-value* greater than 0.01 is desired to consider an arbitrary source of information random with 99% confidence, and higher values indicate a higher confidence about the source randomness [102]. Table 6.5 shows the average *p-value* for the NIST tests applied on the proposed PUFs. From the table, we can also see that INV_PUF and SA_PUF pass all applicable NIST tests (tests that require n>3,040 were omitted). Also, the proposed PUFs consistently have high *p-value*s in individual NIST tests, with no value below 0.28. Hence, the proposed PUF class has high degree of randomness.

Fig. 6.22 shows the energy consumption per bit of the proposed bitcells as a function of the supply voltage. The minimum energy point for both INV_PUF and SA_PUF lies at around 0.9 V, although the latter has an energy that varies only weakly when reducing the voltage due to the additional (and fairly constant) power contribution of the tail current in the senseamp, SA_PUF has an energy per bit that is 11X higher than INV_PUF, for supply voltages from 0.8 to 1 V and assuming an *EN* voltage of 0.5 V in SA_PUF. This energy gap can be narrowed by lowering *EN* voltage, although INV_PUF is expected to have lower energy in any case.

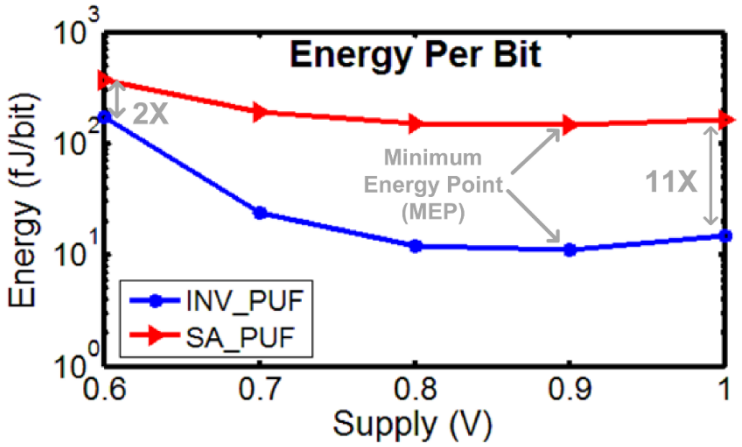


Fig. 6.22 Energy per bit of INV_PUF and SA_PUF for varying voltage supply

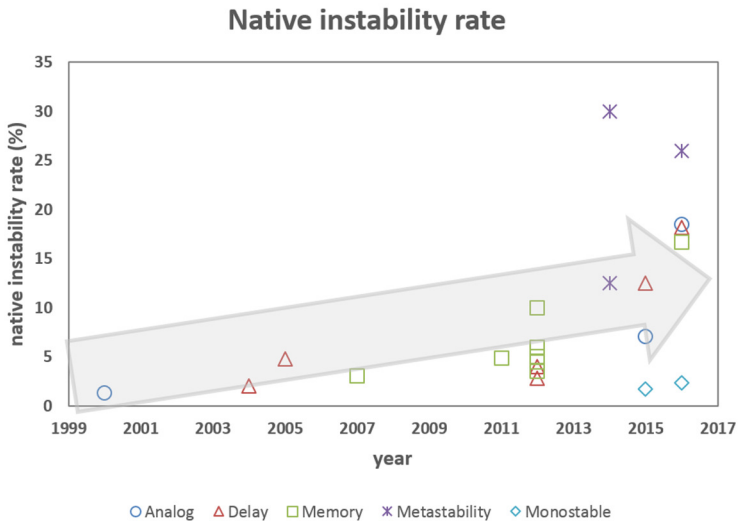
As summarized in Table 6.4, INV_PUF offers a 39.4X energy reduction in the energy per bit at iso-technology, compared to the best previously reported value of 0.19 pJ in 22 nm CMOS [84]. The energy of the few PUFs that were implemented in other technology nodes, such as [84], were scaled to the same 65-nm technology by assuming a 0.7X reduction in capacitance per technology node. Due to its relatively higher consumption compared to INV_PUF, SA_PUF offers only 3.63X improvement over [84]. Observe that the energy advantage obtained by INV_PUF is further enhanced when comparing the PUFs at iso-stability, thanks to the superior stability of the proposed PUF. Indeed, other PUFs requires substantial stability enhancement to reach the same level achieved by INV_PUF, translating into additional consumption that further degrades their energy efficiency. The effective area per bit in Table I was quantified by the ratio of the array area and the actual number of stable (i.e., usable) bitcells. The area per bit of the proposed PUFs is comparable to the most reasonable PUFs, being it 1.5X lower (higher) than [84] for the INV_PUF (SA_PUF). As expected, the SRAM PUF has better area efficiency than any other PUF, with the PUF in [98] being the second best. From Table 6.4, INV_PUF is usually preferable between the proposed PUFs, by virtue of its lower energy and area, and relatively similar statistical characteristics. On the other hand, SA_PUF is preferable only when the requirements in terms of output statistical properties are so stringent that the increased area and energy cost is justified. Neither of the two proposed topologies requires any

calibration or additional reference/biasing circuitry.

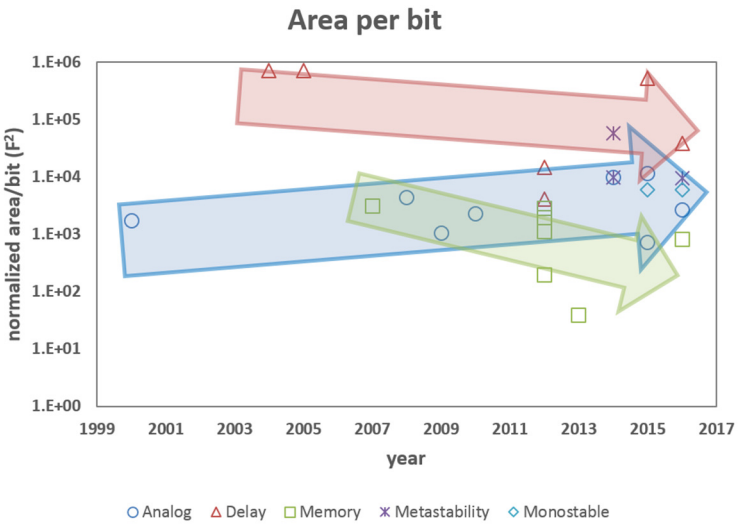
A more complete list of fabricated PUFs can be found in the new public PUF database [132] we compiled. Extracted trends in terms of native instability rate, area, and energy are shown in Fig. 6.23. From Fig. 6.23a, the metastability-based PUFs have the worst native instability rate, while the monostable PUFs exhibit the best native instability rate. The high native instability rate in metastability-based PUFs is reduced through post-processing and other stability enhancement techniques that increase testing time (i.e., cost). For the rest of the PUFs, the native instability rate has slightly increased over the years. From Fig. 6.23b, the area per bit is highest for delay-based PUFs, due to the large number of stages required to 1) limit the oscillation frequency to acceptable values that can be distinguished by the subsequent circuitry, 2) to mitigate the instability rate of individual ring oscillators via k-sum or 1-out-of-k masking [109], [111]. In general, the area efficiency of PUF bitcells has improved over time, especially due to the adoption of more digital approaches that offer better density than analog ones. Analog PUF bitcells have an opposite trend, as their area tends to increase over time, when area is normalized to the square of the minimum feature size of the technology. This is mostly because of their analog nature, which does not enable shrinking with finer technologies.

From Fig. 6.23c, the energy per bit is improving, thanks to the adoption of more energy-aware PUFs. The circuit improvements in terms of energy dominate the benefits of mere technology scaling. This is shown by Fig. 6.23c, which plots the energy normalized to the energy consumed by a minimum-sized inverter in the same technology, and hence represents a technology-independent metric. Interestingly, from Fig. 6.23c delay-based PUFs are an exception, as they tend to have larger energy per bit over the years. This is due to the need for a larger number of ring oscillators or oscillations to maintain acceptable stability, in spite of the progressively worse native stability in Fig. 6.23a.

(a)



(b)



(c)

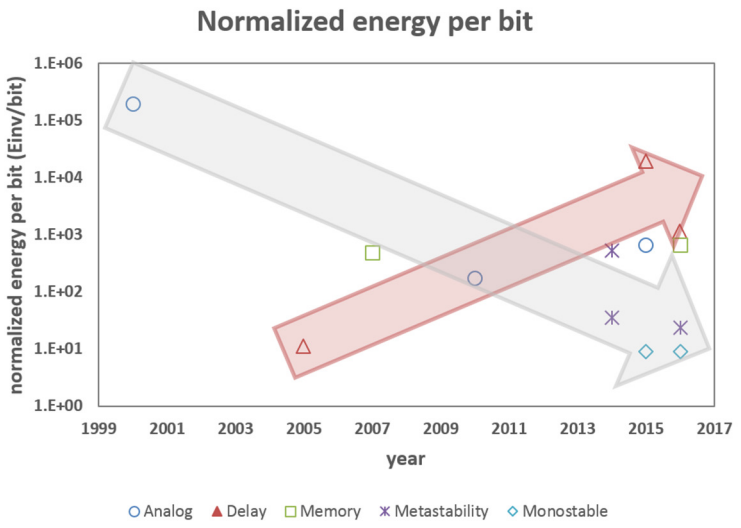


Fig. 6.23 Trend of (a) native instability rate, (b) normalized area per bit, (c) normalized energy per bit for different PUFs [132]

6.5 Possible Future Work on PUFs

Some prior work enables the capability to assure a well-defined stability safety margin at the output word level [101], as a form of robustness assurance against individual bit instability. Other prior work focuses on improving the stability of PUF bitcells without quantitative stability assurance at run-time. For example, introducing burn-in hardening in [84] improves stability at the expense of significantly longer testing time. Another way to improve the statistical quality and suppress a limited number of unstable bits is through digital post-processing, at the expense of substantially larger silicon area and energy. The post-processing block can be a mixture of the following techniques:

- Error Correcting Code (ECC), which introduces a large area/energy overhead especially for high levels of targeted security, as its complexity grows exponentially in applications requiring wider PUF outputs; post-processing also leaks information and makes the PUF more vulnerable to physical attacks [118]. Various ECCs were used [101], such as 2D Hamming [91], BCH [128], [133], two-stage ECC [134], soft-decision ECC [135], [136], Index-Based Syndrome [137], Code-Offset Syndrome [91], [109], [138]–[141], pattern matching techniques [140], and fuzzy extractors [139]
- temporal majority voting across repeated PUF readings, which typically slow down and increase the energy per access by more than an order of magnitude [84], [128], [142]
- on-the-fly PUF bitcell masking [127], and PUF redundancy [98], [109], which skips the bitcells that are found to be unstable at testing time by storing the bit error map in an additional volatile memory array [84], [118], [143]; this approach may introduce significant area/energy overhead, and considerably widens the opportunities to perform successful invasive attacks (e.g., interfering with PUF operation by writing on the additional memory).

Fig. 6.24 shows an example where ECC is used to improve the reliability of the PUF [GCD02]. In this implementation, redundant information is generated for each challenge-response pair, to allow the correction of the PUF output. The ECC overhead is ~14 kgates, which is about an order of magnitude bigger than the PUF array itself. Similarly, in [144], ECC encoder was shown to have an area of ~3-12 kgates, with the ECC decoder requiring an even larger area of ~20-75 kgates.

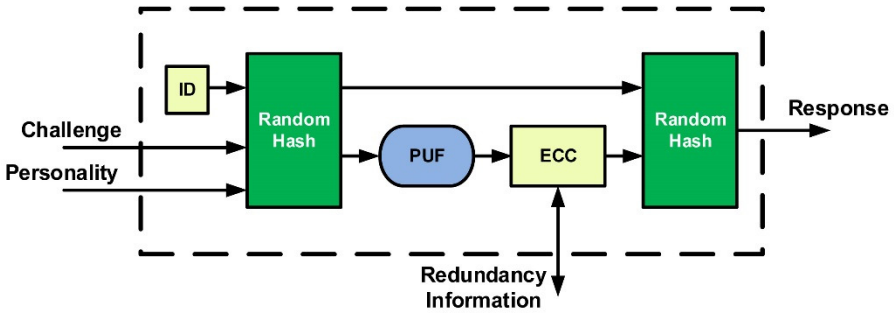


Fig. 6.24 Block diagram of an improved PUF that utilizes ECC to improve the PUF reliability.

Detection of instability was proposed in [143] during the PUF response generation, as shown in Fig. 6.25a. In this circuit, an unwanted 1-0 (0-1) transition results in a rising edge of the clock at the top (bottom) flipflop, which in turn sets *VALID* low. Similarly, in Fig. 6.25b, error is detected at boot time in [127], and is thereafter masked to a predefined value. In this circuit, error is indicated by a difference in the outputs of the latches, which in turn disables the latches, thereby keeping the error signal despite possible switches in *PUFbit*. In Fig. 6.25c, similar error detection in Fig. 6.25b is done using an XOR gate. Like Fig. 6.25a, this error produces a clock rising edge in the flipflop, thereby latching out the fixed logic 1 output to denote that an error has occurred.

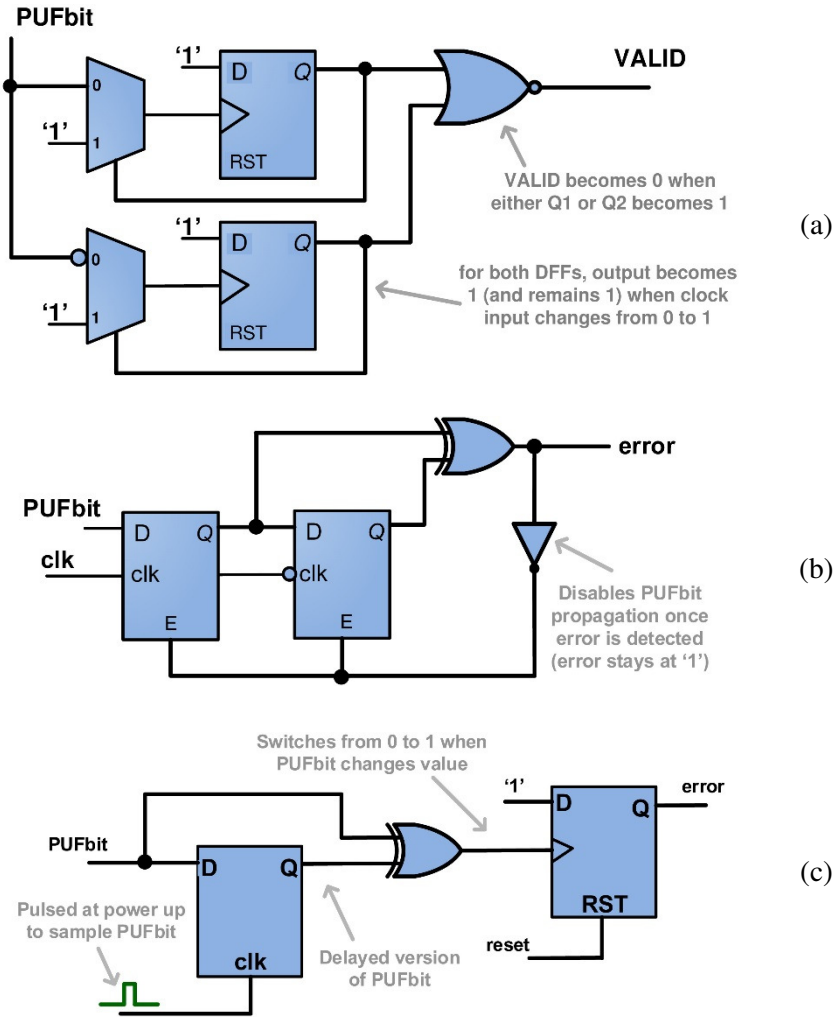


Fig. 6.25 Possible circuits for runtime error detection: (a) glitch detector from [143]; (b) dark bit masking from [127]; and (c) canary-type detection

Having a stable secret key embedded within the chip allows for proper chip authentication [91], [145]. Such keys can also be used as cryptographic keys [85] to encrypt data sent over wireless channel [146], or to establish a trusted communication between nodes in the network [147]. For node-to-node communications, the concept of combining a PUF with a crypto-core can also be used to reduce the circuit complexity and energy required for continuous authentication, thereby reducing the required PUF capacity at a given level of security. Conventional node-to-node communication is illustrated in Fig. 6.26, where CRPs are used to authenticate both nodes each time data is transferred between them. Instead, a more efficient security scheme is introduced in Fig. 6.27. In this "PUF-enabled node-to-node communication" scheme, secure PUF

key exchange is enabled at the authentication phase through cryptography. After one-time authentication, both nodes can communicate with each other securely through encryption and decryption using the exchanged keys, and without server assistance (therefore not needing a large CRP database). This makes communication over complex networks scalable, as the database is involved only at the first communication between nodes. As can be seen in the figure, node-to-node communication is simplified through the joint use of PUF and cryptography, which permit to securely exchange keys over an insecure channel, and avoiding the very energy- and area-hungry public-key cryptography. Such interesting and synergistic use of PUFs and cryptography is here introduced and named “PUF-enhanced cryptography”.

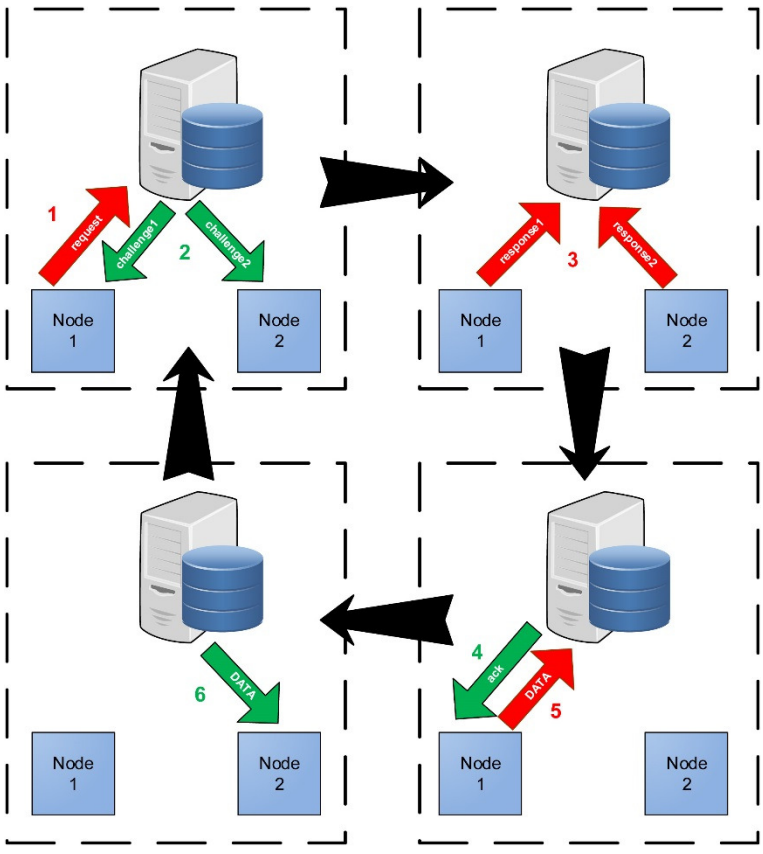


Fig. 6.26 Conventional node-to-node data transfer through server, which needs to constantly assist the two nodes during their communications.

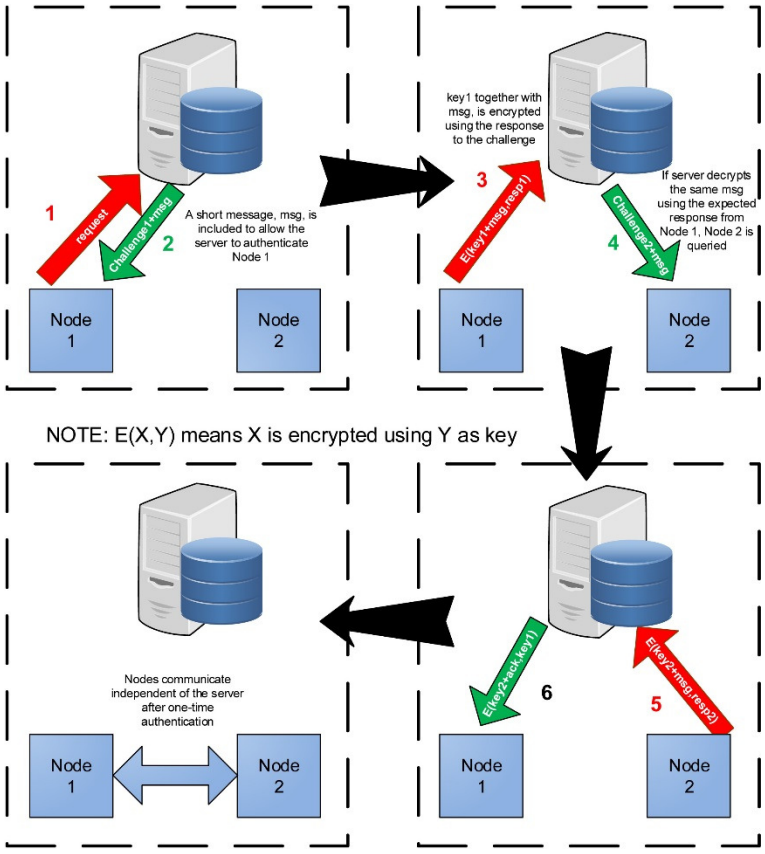


Fig. 6.27 PUF-enabled key exchange and node-to-node communication.

Another interesting ramification of PUF-enhanced cryptography is the ability to substantially strengthen the security of a crypto-core against cryptanalytic attacks, by appropriately embedding a PUF into it. As illustrated in Fig. 6.28, PUF-enhanced cryptography goes beyond the traditional scheme of securely storing a single crypto-key, and permits to extend the crypto-key compared to the size imposed by the crypto-algorithm, thus making it stronger against cryptanalytic attacks. Traditionally, key extension is not possible since its length is dictated by the encryption standard. However, in PUF-enhanced cryptography, a PUF with capacity larger than the key is used to generate repeatable but unpredictable new keys that are combined with the conventional user key to generate the fixed length enhanced key used by the on-chip crypto-core. To this aim, the key enhancer in Fig. 6.28 is introduced to dynamically concatenate the user and PUF keys, and then compress them into the pre-defined length. Although the key enhancer in Fig. 6.28 is shown to be outside the crypto-core

(i.e., without interfering with conventional operation), it can also extend to the inside of the latter, and operate across several blocks of plaintext. The encryption sequence is initialized by the user key, and then managed by a key enhancer. The key enhancer can likewise be a simple finite state machine, which generates time-varying challenges to a PUF, or a lightweight cipher itself [148]. As a result, as opposed to the traditional scheme that uses a single private key, the PUF-enhanced cryptography scheme in Fig. 6.28 actually uses a larger set of keys, whose number is basically limited by the desired PUF capacity.

From an attacker point of view, guessing the private crypto-key of a typical cryptography system requires an effort that is (exponentially) defined by the size of the single key size. Instead, in the PUF-enhanced cryptography scheme in Fig. 6.28, the search space for the crypto-key is enlarged by the capacity of the PUF, thus easily making the key search unfeasible even under very powerful equipment and computing resources. In practical cases, the PUF-enhanced cryptography permits to drastically strengthen the security of an existing algorithm with (1) limited area cost, thanks to the exponential increase of the size of the key search space, under PUF capacity extension, and (2) no throughput penalty, since the generation of the PUF output is generally much faster than encryption. When using PUFs like in [97], the latter property is enabled by the intrinsically high speed of the PUF architecture, since PUF bits are always available at the output and only need to be routed to the circuitry that consumes them.

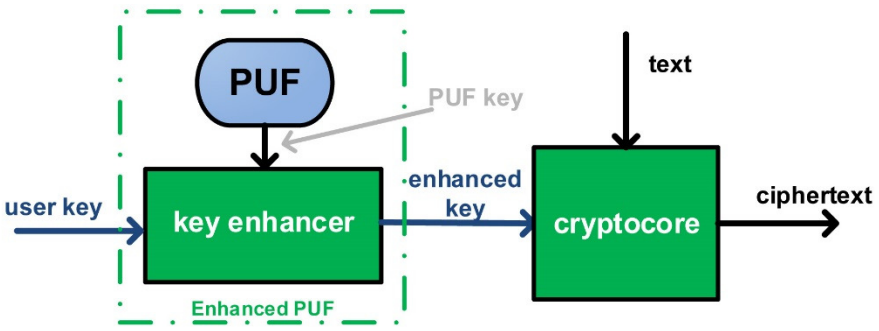


Fig. 6.28 PUF-enhanced enhanced cryptography

The above mentioned dynamic change of the key over time is a tool to improve the strength of PUF-enhanced cryptography against cryptanalytic attacks. In the case of IoT devices relying on energy harvesting, changing keys becomes a necessity as dictated by the availability of supply. For example, in [149] key generation is divided into several phases and precomputation is done whenever supply available, and intermediate results are stored, for use in the next phase.

In summary, PUF-enhanced cryptography permits to drastically enhance the security of a crypto-core by leveraging its synergy with a PUF, to generate time-varying crypto-keys instead of having a fixed one. In addition, the adoption of such PUF to enhance the crypto-algorithm also permits to easily scale up the level of security on demand. Indeed, the level of security defines the number of PUF words that are needed, and hence it only affects the periodicity of the key enhancer for a given PUF capacity. Also, the PUF unambiguously authenticates the die that the crypto-core runs on. In addition, the addition of a PUF to a crypto-core generally entails a very small energy overhead, as the energy per bit of a PUF is typically 2-3 orders of magnitude smaller than a crypto-core. Very similar considerations hold for the area efficiency. These features are particularly interesting in the context of the Internet of Things, as they make crypto-algorithms and crypto-cores affordable in terms of area and energy, thus enabling continuous and ubiquitous security. When a much higher level of security is occasionally needed, the PUF enhancement permits to further scale it up at a very low area/energy cost.

Chapter 7

Energy-Efficient Microcontroller for Wireless Sensor Nodes

Microcontrollers have been the enablers for embedded and emerging applications like wearable electronics and environment sensors [150]–[152]. They control system operations such as data transfer, external communications and power management, which is critical for battery-operated and energy-autonomous systems. To promote sustainable energy, several world-wide green initiatives have been set for 2020, such as, but not limited to, reducing energy use by 20% and increasing the share of renewable energy to 20% [153]. In line with this aim, several researches focus on using power from harvested energy [76], [152], [154]. Correspondingly, the trend in MCUs is towards power efficiencies of 10uW/MHz or less, as shown in Fig. 7.*1* [153].

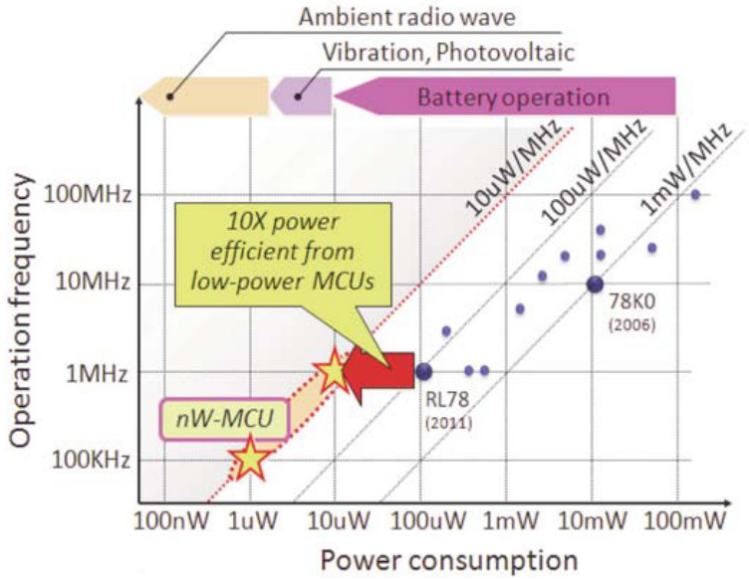


Fig. 7.1 Trends for Low-Power MCUs

One way to reduce power consumption is to aggressively scale the supply voltage to sub- or near-threshold operation [3], [4]. In this chapter, a near-threshold energy-efficient microcontroller SoC for WSN application is presented, to investigate some techniques for energy-efficient design methodology. The design features the following techniques: (1) instruction set architecture (ISA) extension for reduced number of cycles per operation; (2) near-threshold operation of MCU core; and (3) a customized standard cell to ensure proper operation at sub/near-threshold supply voltage [155].

The block diagram of the microcontroller is shown in Fig. 7.2. The main block is named as *Core*. Being a Harvard architecture core, the instruction memory (*IMEM*) and data memory (*DMEM*) are separate. *IMEM* is a 2kB memory, with 16-bit word access. The *DMEM* is a 4kB memory, also with 16-bit word access. The list of input and output pins of the chip is given in Table 7.1. The core was designed using ASIPMeister®, a processor designer with the Brownie [156], [157] core as the base processor.

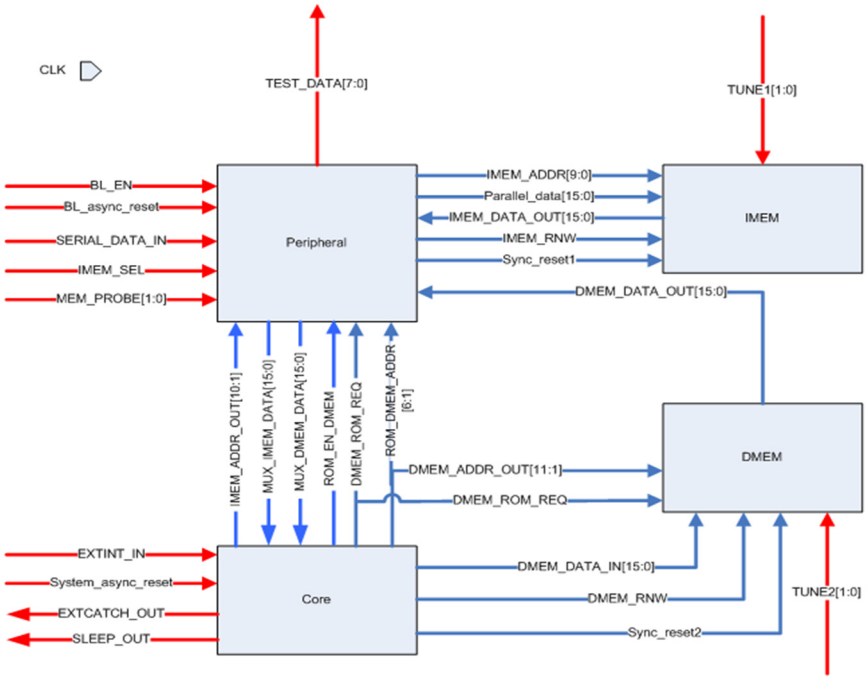


Fig. 7.2 Microcontroller Block Diagram

Table 7.1 I/O Pins of Fabricated Chip

PIN	Direction	Description	Value
CLK	IN	Synchronous system	10MHz freq
BL_async_reset	IN	Asynchronous reset for bit loading	'1' for reset
BL_EN	IN	Bit loading enable signal	'1' to enable data input
System_async_reset	IN	Asynchronous reset	'1' for reset
IMEM_SEL	IN	Select between	'0' for eDRAM
MEM_PROBE[1:0]	IN	Choose byte to output to TEST_DATA	"00" for IMEM LSB; "01" for IMEM MSB; "10" for DMEM LSB; "11" for DMEM MSB
EXTINT_IN	IN	External interrupt to	'1' for interrupt
SERIAL_DATA_IN	IN	Serial input to be	
EXTCATCH_OUT	OUT	Signals recognition	
TEST_DATA[7:0]	OUT	Multiplexed byte from DMEM or IMEM, as indicated by MEM_PROBE	
SLEEP_OUT	OUT	Sleep	'1' for sleep

ASIPMeister® was used to generate the HDL codes for a 32-bit RISC processor. The generated codes use IP from Synopsys Designware® and were synthesized using Synopsys Design Vision®. Compiler and assembler tools for the processor were also provided with the ASIPMeister tool. Area, power and speed estimates are shown in Table 7.2 and Table 7.3, for 100MHz and 100kHz clock frequency, respectively. The 1.2V library is the standard cell that came with the design kit, while the 0.5V and 0.3V libraries are custom libraries for subthreshold operation [155]. Synthesis with 0.3V was done only with 100kHz clock frequency constraint as it is cannot run at 100MHz.

Table 7.2 Processor Parameters at 100MHz

Parameter	1.2V Library	0.5V Library
Area (sq. um)	42447.24	62210.88
Dynamic Power (uW)	433.73	25.48
Leakage Power (nW)	3390.9	347.11
Slack (ns)	+ 93.13	+ 0.03

Table 7.3 Processor Parameters at 100kHz

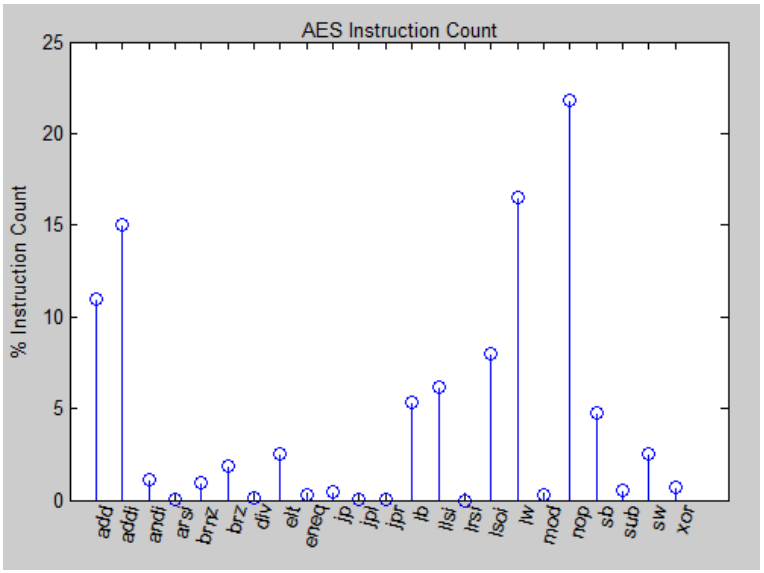
Parameter	1.2V Library	0.5V Library	0.3V Library
Area (sq. um)	44694.72	57614.40	65144.88
Dynamic Power	254.62 uW	951.38 nW	120.69 nW
Leakage Power	3.59 uW	318.91 nW	161.57 nW
Slack (ns)	+ 9759.79	+ 9403.86	+ 92.03

Table 7.4 Benchmark code sizes

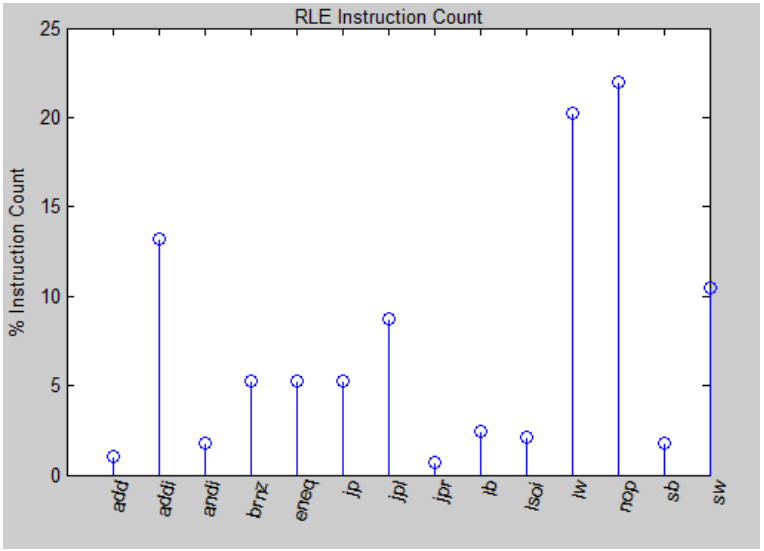
Benchmark	Code length (32-bit words)	Longest loop (32-bit words)
Crypt3	1098	678
AES	1179	571
RLE	287	--

Three benchmarks, namely crypt3, AES and RLE, were used to profile the processor. Both crypt3 and AES are encryption algorithms, while RLE (run length encoding) is a compression algorithm. The compiler was used to obtain the assembly codes from the benchmark C codes. Table 7.4 shows the instruction count and longest loop for the three benchmarks. From the table, we can say that we would need at least 1179x32 bits of program memory if we want to be able to perform AES, or 1098x32 bits for crypt3.

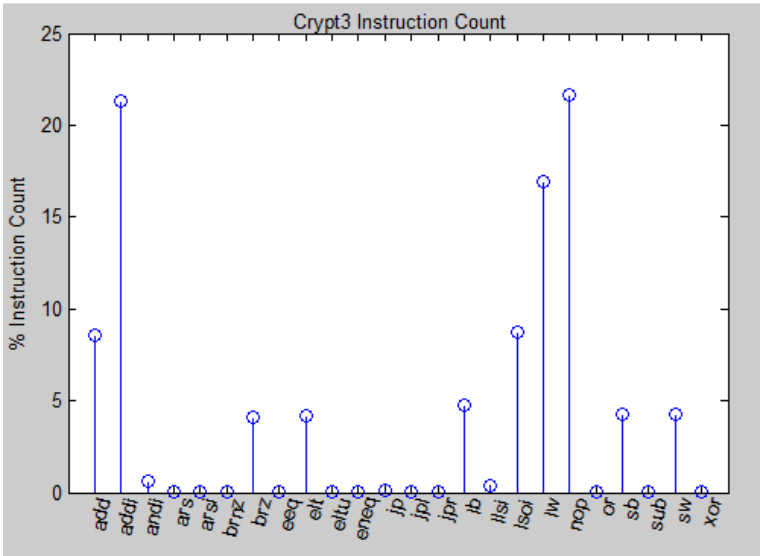
In the absence of a debugger for the processor, loops from the assembly code generated from the compiler were manually unrolled to determine the percentage usage of the different instructions and identify possible instructions to optimize. The corresponding percentages per benchmark is shown in Fig. 7.3. For all three benchmarks, the instructions with the highest percentages are NOP, ADDI, LW, LSOI and ADD.



(a)



(b)



(c)

Fig. 7.3 Instruction count per unrolled benchmark code

Careful examination of the assembly code shows that the NOP instructions can be reduced through compiler improvements, and thus were excluded from the analysis for instructions for ISA extension. One such extension is the combination of ADD and LSOI, as illustrated in Fig. 7.4.

Fig. 7.4 LSAI instruction extension

With the initial extensions (LSAI and branches) as well as improvements to reduce NOP, the total program memory needed for crypt3 was reduced by around 26%. By doing this, leakage power of the system can potentially be reduced through the use of a smaller program memory.

The Brownie core comes in 16-bit (BrownieMicro16) and 32-bit (BrownieSTD32) versions. Table 7.5 shows the basic architecture of these cores. The two cores were compared in terms of area, power consumption and IMEM size requirement. Synopsys Design Vision® was used to synthesize the designs using a limited (generic gates only, with no adder or multiplier cell) customized standard cell for 500mV supply [10], [155] and a target frequency of 100 kHz. Table 7.6 shows the result of this comparison.

Table 7.5 Brownie Core Architecture

Parameter	BrownieSTD32	BrownieMicro16
Base Architecture	RISC	RISC
Memory Architecture	Harvard	Harvard
Data length	32	16
Addressing	Byte	Byte
# of GPRs	32	16
# of pipeline stages	4	3
# Floating Point Unit	0	0
Forwarding	Full forwarding	Full forwarding

Table 7.6 Comparison of Brownie 32-bit and 16-bit Cores

Parameter	BrownieSTD32[19]	BrownieMicro16 [20]
Area (sq. um)	57614.88	12052.08
Dynamic Power (nW)	951.38	197.81
Leakage Power (nW)	318.91	61.46
Total Power (uW)	1.27	0.30
Slack (ns)	9403.86	9866.65

As expected, we can see from Table 7.6 that the 16-bit core, BrownieMicro16, is smaller and consumes less power than its 32-bit counterpart, BrownieSTD32. This is because the BrownieMicro16 is a 16-bit processor, with 16 16-bit registers in the register file while the BrownieSTD32 is a 32-bit processor with 32 32-bit registers in the register file. What was not expected, however, was their more than 4x difference in area. In terms of power consumption, there is a 4x difference in total power consumption of BrownieMicro16 compared to BrownieSTD32. This can still be due to the more complex design of the BrownieSTD32 compared to that of BrownieMicro16. One thing to note, however, is that since the target application is that of WSNs, the cores are expected to be idle most of the time, and therefore consume mostly leakage power. Thus, in this respect, the BrownieMicro16 is still a better choice because of its low leakage power. The speed of the two cores are comparable, showing that both can reach upto 1MHz operating frequency.

One possible benefit of a 32-bit processor over a 16-bit one could be the length of code needed to implement a certain task. The length of code would determine the size of the IMEM needed. And, knowing that memory units are usually leaky, we want to minimize the IMEM size needed. We determine the size requirement by taking the C code of representative WSN applications. These C codes were then compiled and assembled using the compiler and assembler that came with the core. The produced assembly code was not optimal, so we modified the assembly code a little, removing unnecessary NOPs. For the case of the BrownieMicro16, manual conversion from the original BrownieSTD32 had to be done since no compiler and assembler were provided

with the BrownieMicro16. After conversion, unnecessary NOPs were likewise removed, to have a fair comparison with the 32-bit version. We then determine the minimum size of IMEM needed by getting the nearest power of two size that could contain the largest code. Table 7.7 shows the resulting code lengths with three benchmarks used. The BrownieSTD32_mod represents the modified assembly code after removal of the unnecessary NOPs.

Table 7.7 Code Size Comparison with 16- and 32-bit Cores

Benchmark	BrownieSTD32	BrownieSTD32_mod	BrownieMicro16
Crypt3	697	609	657
RLE	168	142	142
FIR	30	30	39

We can see from Table 7.7 that except BrownieSTD32_mod indeed has shorter code length compared to BrownieMicro16. The longer code length of the FIR benchmark in the BrownieMicro16 core is because of the absence of a multiplier in the BrownieMicro16. The FIR benchmark is composed mostly of multiply and add commands, while the other two benchmarks are mostly compare and add. Thus, the absence of a multiplier unit in the BrownieMicro16 core forces the core to perform multiplication using a series of additions, thereby producing a lengthy code. Comparing all three benchmarks, however, the longest one is still the Crypt3. Thus, it is the Crypt3 benchmark that will dictate the size of the IMEM. Based on the data from Table 7.7, we get a minimum IMEM size of 32kb for the BrownieSTD32 and 16kb for the BrownieMicro16. Thus, the BrownieMicro16 core is still a better choice compared to BrownieSTD32. Seeing the effect of the absence of a multiplier in the BrownieMicro16 core, we then proceeded to extending its instruction to include multiplication in order to reduce code length, and more importantly, the number of cycles.

A 16x16 multiplier unit in the BrownieMicro16 core was added, to go with the *mult* ISA extension. The new core with the additional multiplier unit (BrownieMult16) was synthesized using Synopsys Design Compiler® with the complete customized

500mV standard cell library [158].The comparison of the original BrownieMicro16 with the new BrownieMult16, for a 7.5MHz operation at 0.5V is shown in Table 7.8.

Table 7.8 Comparison of Cores at 0.5V, 7.5MHz

Parameter	BrownieMicro16	BrownieMult16
Area (sq. um)	12639.60	18026.28
Dynamic Power (uW)	3.27	3.33
Leakage Power (nW)	60.38	94.05
Total Power (uW)	3.33	3.42
Slack (ns)	0.03	0

Because of the addition of the multiplier unit, the increase in area and power is expected. It should be noted, though, that although the area increase is almost 50%, the increase in total power is less than 3%. This results in less than 0.5uW/MHz energy efficiency of the processor core. By adding the multiplier unit, the FIR code length is now 30, same as that of 32-bit version, which can be translated to an even better performance in terms of number of clock cycles for the operation.

After verifying the functionality of the design, it was automatically placed and routed using Cadence SoC Encouter® and the resulting area of the core was approximately 180um x 180um. The area breakdown is shown in Fig. 7.5.

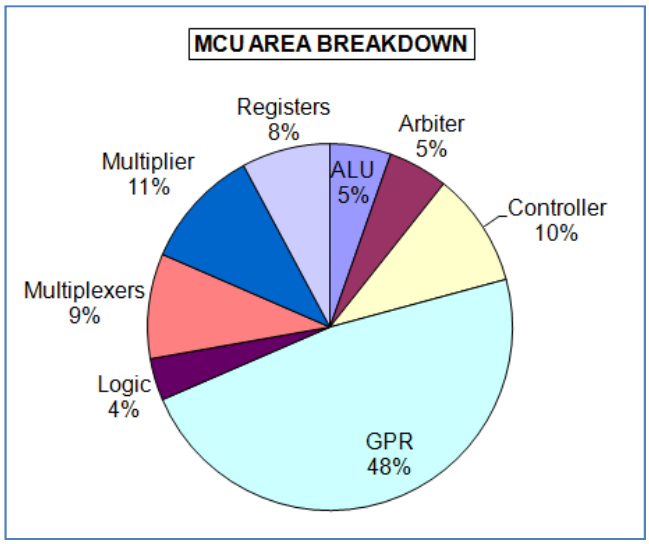


Fig. 7.5 Area breakdown of BrownieMult16 core

The peripheral block in Fig. 7.2 was mainly added for interfacing and testing. It is composed of the bitloading circuit (for serial input to parallel output), a 128x16-bit memory array to act as alternate IMEM, a 64x16-bit ROM containing randomized numbers for key generation, and some multiplexers as well as synchronizing circuits for reset. The resulting area estimate for the peripheral block is 30657 sq.um, of which, almost 95% is from the alternate IMEM.

A sample short pseudocode for testing (labelled as *TEST*) is shown in Fig. 7.6. Based on timing reports from synthesis, it was determined that the critical path would involve addition with register operands (*ADD*). Because of the complexity of the multiplier which is expected to be completed within 1 cycle, the *MUL* instruction was also included.

1. LD data from ROM (uses ADDI, LD and LHI instructions
2. Operate on the data (uses ADD, ADDI, LLI and MUL instructions)
3. Check if result is correct (uses ADDI, BRNZ, LLI and SEQ instructions)
4. Go back to 1 if results is correct
5. NOP then SLEEP
6. JPR to top

Fig. 7.6 TEST pseudocode

Fig. 7.7 shows the power consumption per operation of the core when ran at 0.5V supply and 10MHz clock frequency. We can see from the figure that the power consumption of a *NOP* operation is only slightly less than a *SLEEP* operation. Also, the power consumption of *TEST*, which should include the critical path, is just slightly lower than a *LD* operation. Comparing the *NOP* and *SLEEP* with the *LD* and *TEST*, we can see that the *NOP*/*SLEEP* consumes just slightly above half of that consumed by a *LD* operation. It should be noted that these data are only for the core (IMEM, DMEM and peripheral block not included).

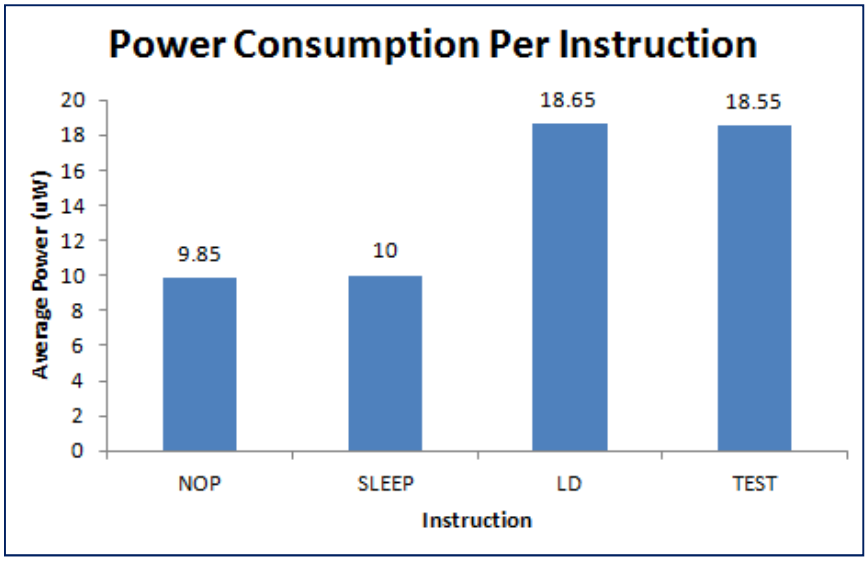


Fig. 7.7 Power consumption per instruction at 0.5V, 10MHz

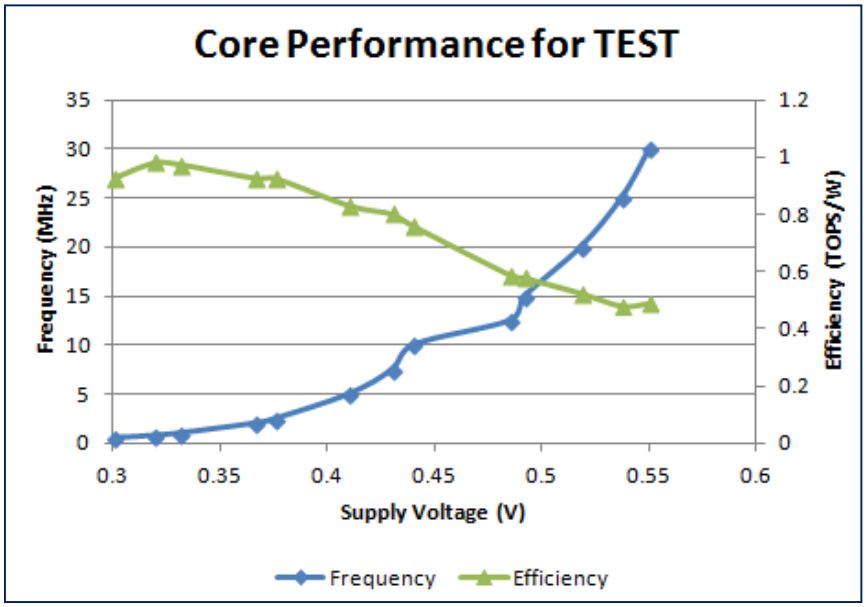


Fig. 7.8 Performance of the core using TEST code

Fig. 7.8 shows the performance of the core for different supply voltage and clock frequency. It can be seen that the maximum efficiency point of the core is at around 320mV with a frequency of around 750kHz. The operating point near the intersection of the efficiency and frequency plot is around 500mV with a frequency of 16.5MHz. Table 7.9 summarizes the design parameters of the core. It should be noted that the data in Table 7.6 refers only to the core. In this experiment, we were able to show the effectiveness of the custom library for 500mV. Our MCU's efficiency of 1.86 uW/MHz

is also comparable with that of the SleepWalker MCU in [159], which has 2uW/MHz using also a 65nm CMOS technology.

Table 7.9 BrownieMult16 Design Parameters

Parameter	Value
Technology	65nm CMOS
Supply Voltage	500 mV
Frequency	10 MHZ
Average Power	18.55 uW
Efficiency	1.86 uW/MHz

Chapter 8

Conclusion

This thesis presents a base framework for the design of ubiquitous computer vision systems. In today's era of internet of things and ubiquitous computing, the demand for high connectivity, especially with battery operated systems or those with embedded energy harvesting units, continuous to increase. Ubiquitous surveillance systems are not an exception. Scene analysis and object detection and classification through feature extraction requires continuous processing and therefore energy consumption. This poses more stringent constraints in current designs, compelling designers to design for sub- or near-threshold voltage operation, or propose energy-efficient techniques.

Energy-quality scalability adds another dimension to the existing energy-performance trade-off. In EQSCALE (Chapters 3 and 4), we introduce tuneable knobs to allow balance between energy and quality (in terms of image resolution and matching percentage). We showed that we are able to provide an order improvement in energy consumption with a feature extraction hardware with similar area and complexity. Compared with a feature extraction hardware with similar energy consumption, the area of EQSCALE is an order smaller.

With the data re-use in EQSCALE, an external DRAM (frame buffer) is no longer necessary. Instead, an SRAM that serves as a sub-frame (less than 1/3 of a frame) buffer can be used. Leveraging on the correlation between adjacent pixels in an image, a non-precharge SRAM (Chapter 5) was proposed. By removing the precharge stage, 50% of the energy consumption compared to 8T SRAM can potentially be saved. The resulting speed is comparable to 8T, with 15% area overhead.

To ensure the authenticity and security of data, a light weight physically unclonable function was proposed for use as chip identification (Chapter 6). A class of static, monostable PUFs was presented, and results have shown that it has the best repeatability, identifiability and randomness compared with other PUFs. It consumes 15fJ/bit of energy, which is at least an order lower compared to other PUFs.

Low yield due to variations is another key limitation when designing in sub- or near-threshold voltages. This can be improved through upsizing, or by adding more margin in the design to allot for the variations. Using a customized standard cell designed for near-threshold operation, a 16-bit MCU with modified instruction set was designed (Chapter 7). A maximum energy efficiency point at around 320mV (sub-threshold) with a frequency of 750kHz was achieved using a commercial 65nm CMOS technology. Although ISA extension results in slightly larger area and power, it reduces energy consumption by decreasing the number of cycles required to finish one task.

References

[1] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011.

[2] Y. Pu *et al.*, "From Xetal-II to Xetal-Pro : On the Road Towards An Ultra Low-Energy and High Throughput SIMD Processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 4, pp. 472–484, 2011.

[3] Y. Pu, J. Pineda de Gyvez, H. Corporaal, and Y. Ha, "An Ultra-Low-Energy Multi-Standard JPEG Co-Processor in 65 nm CMOS With Sub/Near Threshold Supply Voltage," *IEEE J. Solid-State Circuits*, vol. 45, no. 3, pp. 668–680, Mar. 2010.

[4] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[5] M. Pedram and J. M. Rabaey, *Power Aware Design Methodologies*. KluwerAcademic Publishers, 2002.

[6] D. M. Markovic, "A Power / Area Optimal Approach to VLSI Signal Processing," University of California, Berkeley, 2006.

[7] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-Power Design in Near-Threshold Region," *Proc. IEEE*, vol. 98, no. 2, pp. 237–252, Feb. 2010.

[8] J. Burr and A. Peterson, "Ultra Low Power CMOS Technology," in *NASA Symposium on VLSI Design*, 1991, p. 11.1.1-11.1.12.

[9] B. Zhai *et al.*, "Energy-Efficient Subthreshold Processor Design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 8, pp. 1127–1137, 2009.

[10] Z. Wenfeng, "Ultra Energy-Efficient Sub-/Near-Threshold Computing: Platform and Methodology," National University of Singapore, 2013.

[11] S. Mysore, B. Agrawal, F. T. Chong, and T. Sherwood, "Exploring the Processor and ISA Design for Wireless Sensor Network Applications," in *International Conference on VLSI Design (VLSID)*, 2008, pp. 59–64.

[12] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[13] A. Wang and A. Chandrakasan, "A 180mV FFT Processor Using Subthreshold Circuit Techniques," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2004, vol. 34, no. 3, pp. 380–387.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, 1999, pp. 1150–1157 vol.2.

[15] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” 2003.

[16] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF : Speeded Up Robust Features,” in *European Conference on Computer Vision (ECCV)*, 2006, pp. 404–417.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346--359, 2008.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *International Conference on Computer Vision*, 2011, pp. 2564–2571.

[20] S. Park *et al.*, “An Energy-Efficient and Scalable Deep Learning/Inference Processor with Tetra-Parallel MIMD Architecture for Big Data Applications,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 6, pp. 838–848, 2015.

[21] A. Ardakani, C. Condo, and W. J. Gross, “Sparsely-Connected Neural Networks: Towards Efficient VLSI Implementation of Deep Neural Networks,” in *International Conference on Learning Representations (ICLR)*, 2017, pp. 1–14.

[22] H. B. Barlow, “Summation and Inhibition in the Frog’s Retina,” *J. Physiol.*, vol. 119, pp. 69–88, 1953.

[23] K. A. C. Martin, “A Brief History of the ‘Feature Detector,’” *Cereb. Cortex*, vol. 4, no. 1, pp. 1–7, 1994.

[24] S. Papert, “The Summer Vision Project.” pp. 1–6, 1966.

[25] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[26] T. Lindeberg, “Scale-space theory : A basic tool for analysing structures at different scales,” *J. Appied Stat.*, vol. 21, no. 2, pp. 225–270, 1994.

[27] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” in *Alvey Vision Conference*, 1988, pp. 147–151.

[28] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *EEE International Conference on Computer Vision (ICCV’)*, 2005, pp. 1508–1515.

[29] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” *Lect. Notes Comput. Sci.*, vol. 3951, pp. 430–443, 2006.

[30] M. Calonder, V. Lepetit, and P. Fua, “BRIEF : Binary Robust Independent Elementary Features,” in *Lecture Notes in Computer Science*, 2010, pp. 778–792.

[31] K. Mikolajczyk and C. Schmid, “Performance evaluation of local descriptors.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–30, Oct. 2005.

[32] L. Juan and O. Gwun, “A Comparison of SIFT , PCA-SIFT and SURF,” *Int. J. Image Process.*, vol. 3, no. 4, pp. 143–152, 2009.

[33] Katholieke Universiteit Leuven, Inria Rhone-Alpes, and Center for Machine Perception, “Affine Covariant Features.” [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/.

[34] I. Khvedchenia, “Feature Descriptor Comparison Report,” *Computer Vision Talks*, 2011. [Online]. Available: http://computer-vision-talks.com/2011/08/feature-descriptor-comparison-report/.

[35] I. Khvedchenia, “Comparison of OpenCV’s Feature Detection Algorithms II,” *Computer Vision Talks*, 2011. [Online]. Available: http://computer-vision-talks.com/2011/07/comparison-of-the-opencvs-feature-detection-algorithms-ii/.

[36] D. Kim, K. Kim, J. Kim, S. Lee, S. Lee, and H. Yoo, “81.6 GOPS Object Recognition Processor Based on a Memory-Centric NoC,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 3, pp. 370–383, 2009.

[37] K. Kim, S. Lee, J. Kim, M. Kim, and H. Yoo, “A 125 GOPS 583 mW Network-on-Chip Based Parallel Processor With Bio-Inspired Visual Attention Engine,” *IEEE J. Solid State Circuits*, vol. 44, no. 1, pp. 136–147, 2009.

[38] J. Kim *et al.*, “A 201.4 GOPS 496mW Real-Time Multi-Object Recognition Processor with Bio-Inspired Nueral Perception Engine,” in *IEEE Journal of Solid State Circuits (JSSC)*, 2009, vol. 45, no. 1, pp. 150–152.

[39] S. Lee, J. Oh, J. Park, J. Kwon, M. Kim, and H. Yoo, “A 345 mW Heterogeneous Many-Core Processor With an Intelligent Inference Engine for Robust Object Recognition,” *IEEE J. Solid State Circuits*, vol. 46, no. 1, pp. 42–51, 2011.

[40] S. Lee, J. Oh, M. Kim, J. Park, J. Kwon, and H.-J. Yoo, “A 345mW Heterogenous Many-Core Processor with an Intelligent Inference Engine for robust object recognition,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2010, vol. 43, no. 3, pp. 1116–1128.

[41] J.-Y. Kim, S. Oh, S. Lee, M. Kim, J. Oh, and H.-J. Yoo, “An attention controlled multi-core architecture for energy efficient object recognition,” *Signal Process. Image Commun.*, vol. 25, no. 5, pp. 363–376, Jun. 2010.

[42] Y. Su, K. Huang, T. Chen, Y. Tsai, S. Chien, and L. Chen, “A 52mW Full HD 160-Degree Object Viewpoint Recognition SoC with Visual Vocabulary Processor for Wearable Vision Applications,” in *Symposium on VLSI Circuits*, 2011, pp. 258–259.

[43] J. Oh *et al.*, “Low-Power Real-time Object-Recognition Processors for Mobile Vision Systems,” *IEEE Computer*, pp. 38–50, 2012.

[44] D. Jeon *et al.*, “An Energy Efficient Full-Frame Feature Extraction Accelerator With Shift-Latch FIFO in 28 nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 49, no. 5, pp. 1271–1284, 2014.

[45] D. Jeon, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, “A Low-Power VGA Full-Frame Feature Extraction Processor,” in *International*

Conference on Acoustics Speech and Signal Processing (ICASSP), 2013, pp. 2726–2730.

[46] J. Park, H. Kim, and L. Kim, "A 182 mW 94.3 f/s in Full HD Pattern-Matching Based Image Recognition Accelerator for an Embedded Vision System in 0.13-μm CMOS Technology," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 5, pp. 832–845, 2013.

[47] K. Lee and K. Byun, "A hardware design of optimized ORB algorithm with reduced hardware cost," *Adv. Sci. Technol. Lett.*, vol. 43, no. Multimedia, pp. 58–62, 2013.

[48] K. Lee, "A Design of an Optimized ORB Accelerator for Real-Time Feature Detection," *Int. J. Control Autom.*, vol. 7, no. 3, pp. 213–218, 2014.

[49] P. Viswanath, P. Swami, K. Desappan, and A. Jain, "ORB in 5ms : An efficient SIMD friendly implementation," in *Asian Conference on Computer Vision (ACCV)*, 2014.

[50] J. Weberruss, L. Kleeman, and T. Drummond, "ORB Feature Extraction and Matching in Hardware," in *Australasian Conference Robotics and Automation (ACRA)*, 2015.

[51] G. Bradski, "OpenCV," *Dr. Dobb's Journal of Software Tools*. 2000.

[52] O. Chum and J. Matas, "Matching with PROSAC-Progressive Sample Consensus," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 220–226.

[53] M. a Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[54] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Midwest Symposium on Circuits and Systems*, 2010, pp. 129–132.

[55] P. Meinerzhagen, O. Andersson, B. Mohammadi, Y. Sherazi, A. Burg, and J. N. Rodrigues, "A 500 fW/bit 14fj/bit-access 4kb Standard-Cell Based Sub-VT Memory in 65nm CMOS," in *European Solid-State Circuits Conference (ESSCIRC)*, 2012, pp. 321–324.

[56] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014, vol. 57, pp. 10–14.

[57] J. Oh *et al.*, "A 320 mW 342 GOPS Real-Time Dynamic Object Recognition Processor for HD 720p Video Streams," *IEEE J. Solid State Circuits*, vol. 48, no. 1, pp. 33–45, 2013.

[58] J. M. Aman, J. Yao, and R. M. Summers, "Content-Based Image Retrieval on CT Colonography Using Rotation and Scale Invariant Features and Bag-Of-Words Model," in *IEEE International Symposium on Biomedical Imageing (ISBI)*, 2010, pp. 1357–1360.

[59] R. G. J. Wijnhoven and P. H. N. De With, "Comparing Feature Matching for Object Categorization in Video Surveillance," *Lect. Notes Comput. Sci.*, vol. 5807, pp. 410–421, 2009.

[60] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," in *European Conference on Computer Vision (ECCV)*, 2004, pp. 1–22.

[61] J. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Prentice Hall, 2002.

[62] A. Raychowdhury, S. Mukhopadhyay, and K. Roy, "A Feasibility Study of Subthreshold SRAM Across Technology Generations *," in *International Conference on Computer Design (ICCD)*, 2005.

[63] B. H. Calhoun and A. P. Chandrakasan, "Static Noise Margin Variation for Sub-threshold SRAM in 65-nm CMOS," *IEEE J. Solid State Circuits*, vol. 41, no. 7, pp. 1673–1679, 2006.

[64] M. Qazi, M. E. Sinangil, and A. P. Chandrakasan, "Challenges and Directions for Low-Voltage SRAM," *IEEE Design and Test of Computers*, pp. 32–43, 2011.

[65] K. Takeda *et al.*, "A Read-Static-Noise-Margin-Free SRAM Cell for Low-VDD and High-Speed Applications," in *IEEE Journal of Solid State Circuits (JSSC)*, 2005, vol. 41, no. 1, pp. 478–480.

[66] J. P. Kulkarni, K. Kim, and K. Roy, "A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM," *IEEE J. Solid State Circuits*, vol. 42, no. 10, pp. 2303–2313, 2007.

[67] J. P. Kulkarni, K. Kim, S. P. Park, and K. Roy, "Process variation tolerant SRAM array for ultra low voltage applications," in *IEEE Design Automation Conference (DAC)*, 2008, pp. 108–113.

[68] J. P. Kulkarni and K. Roy, "Ultralow-Voltage Process-Variation-Tolerant Schmitt-Trigger-Based SRAM Design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 2, pp. 319–332, 2012.

[69] B. Zhai, S. Hanson, D. Blaauw, and D. Sylvester, "A Variation-Tolerant Sub-200 mV 6-T Subthreshold SRAM," *IEEE J. Solid State Circuits*, vol. 43, no. 10, pp. 2338–2348, 2008.

[70] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson, "A Sub-200mV 6T SRAM in 0.13um CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2007, pp. 8–10.

[71] L. Chang *et al.*, "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches," *IEEE J. Solid State Circuits*, vol. 43, no. 4, pp. 956–963, 2008.

[72] N. Verma and A. P. Chandrakasan, "A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2007, pp. 328–330.

[73] T. Kim, S. Member, J. Liu, J. Keane, C. H. Kim, and A. D. S. Cell, "A 0.2 V , 480 kb Subthreshold SRAM With 1 k Cells Per Bitline for Ultra-Low-Voltage Computing," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 518–529, 2008.

[74] B. H. Calhoun and A. P. Chandrakasan, "A 256-kb 65-nm Sub-threshold SRAM Design for Ultra-Low-Voltage Operation," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 680–688, Mar. 2007.

[75] M. Fojtik *et al.*, “A Millimeter-Scale Energy-Autonomous Sensor System With Stacked Battery and Solar Cells,” *IEEE J. Solid State Circuits*, vol. 48, no. 3, pp. 801–813, 2013.

[76] G. Chen *et al.*, “Millimeter-Scale Nearly Perpetual Sensor System with Stacked Battery and Solar Cells,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2010, pp. 166–167.

[77] M. E. Sinangil and A. P. Chandrakasan, “Application-Specific SRAM Design Using Output Prediction to Reduce Bit-Line Switching Activity and Statistically Gated Sense Amplifiers for Up to 1.9x Lower Energy / Access,” *IEEE J. Solid State Circuits*, vol. 49, no. 1, pp. 1–11, 2014.

[78] M. E. Sinangil and A. P. Chandrakasan, “An SRAM Using Output Prediction to Reduce BL-Switching Activity and Statistically-Gated SA for up to 1.9x Reduction in Energy / Access,” in *IEEE International Solid-State Circuits Conference (ISSC)*, 2013, pp. 318–320.

[79] J. Kwon, I. J. Chang, I. Lee, H. Park, and J. Park, “Heterogeneous SRAM Cell Sizing for Low-Power H.264 Applications,” *IEEE Trans. Circuits Syst. I*, vol. 59, no. 10, pp. 2275–2284, 2012.

[80] I. J. Chang, D. Mohapatra, and K. Roy, “A Priority-Based 6T/8T Hybrid SRAM Architecture for Aggressive Voltage Scaling in Video Applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 101–112, Feb. 2011.

[81] N. Gong, S. Jiang, A. Challapalli, S. Fernandes, R. Sridhar, and S. Member, “Ultra-Low Voltage Split-Data-Aware Embedded SRAM for Mobile Video Applications,” *IEEE Trans. Circuits Syst. II*, vol. 59, no. 12, pp. 883–887, 2012.

[82] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, “Drowsy caches: simple techniques for reducing leakage power,” in *International Symposium on Computer Architecture (ISCA)*, 2002, pp. 148–157.

[83] B. Zimmer *et al.*, “SRAM Assist Techniques for Operation in a Wide Voltage Range in 28-nm CMOS,” *IEEE Trans. Circuits Syst. II*, vol. 59, no. 12, pp. 853–857, 2013.

[84] S. K. Mathew *et al.*, “A 0.19pJ/b PVT-Variation-Tolerant Hybrid Physically Unclonable Function Circuit for 100% Stable Secure Key Generation in 22nm CMOS,” in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference (ISSCC)*, 2014, vol. 2, no. c, pp. 278–280.

[85] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. can der Leest, “Experimental Evaluation of Physically Unclonable Functions in 65 nm CMOS,” in *European Solid State Circuit Conference (ESSCIRC)*, 2012, pp. 486–489.

[86] R. Maes, “Physically Unclonable Functions : Constructions , Properties and Applications,” Katholieke Universiteit Leuven, 2012.

[87] S. Rosenblatt *et al.*, “Field Tolerant Dynamic Intrinsic Chip ID Using 32 nm High-K/Metal Gate SOI Embedded DRAM,” *IEEE J. Solid State Circuits*, vol. 48, no. 4, pp. 940–947, 2013.

[88] Y. Su, J. Holleman, and B. Otis, “A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations,” in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference (ISSCC)*, 2007, pp. 406–408.

[89] D. Nedospasov, J. P. Seifert, C. Helfmeier, and C. Boit, “Invasive PUF analysis,” in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2013, pp. 30–38.

[90] A. Alvarez, W. Zhao, and M. Alioto, “15fJ/b Static Physically Unclonable Functions for Secure Chip Identification with < 2% Native Bit Instability and 140x Inter/Intra PUF Hamming Distance Separation in 65nm,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2015, vol. 5, pp. 256–258.

[91] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon Physical Random Functions,” in *ACM Conference on Computer and communications security (CCS)*, 2002, p. 148.

[92] T. Xu, J. B. Wendt, and M. Potkonjak, “Matched Digital PUFs for Low Power Security in Implantable Medical Devices,” *2014 IEEE Int. Conf. Healthc. Informatics*, pp. 33–38, 2014.

[93] A.-R. Sadeghi and D. Naccache, Eds., *Towards Hardware-Intrinsic Security: Foundations and Practice*. Springer, 2010.

[94] R. Maes, *Physically Unclonable Functions: Construction, Properties and Applications*. London: Springer, 2013.

[95] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater, “On a New Way to Read Data from Memory,” in *International IEEE Security in Storage Workshop*, 2002, pp. 65–69.

[96] O. Kömmerling and M. G. Kuhn, “Design Principles for Tamper-Resistant Smartcard Processors,” in *USENIX Workshop on Smartcard Technology*, 1999, pp. 9–20.

[97] A. B. Alvarez, W. Zhao, and M. Alioto, “Static Physically Unclonable Functions for Secure Chip Identification at 0.6-1 V and 15fJ/bit in 65nm,” *IEEE J. Solid State Circuits*, vol. 51, no. 3, pp. 763–775, 2016.

[98] K. Yang, Q. Dong, D. Blaauw, and D. Sylvester, “A Physically Unclonable Function with BER < 10^-8 for Robust Chip Authentication Using Oscillator Collapse in 40nm CMOS,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2015, pp. 254–256.

[99] S. Stanzione and G. Iannaccone, “Silicon Physical Unclonable Function Resistant to a 10^25-trial brute force Attack in 90 nm CMOS,” in *Symposium on VLSI Circuits*, 2009, pp. 116–117.

[100] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, “Extracting Secret Keys from Integrated Circuits,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 10, pp. 1200–1205, 2005.

[101] M. M. Yu, R. Sowell, A. Singh, D. M. Raihi, and S. Devadas, “Performance Metrics and Empirical Results of a PUF Cryptographic Key Generation ASIC,” in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 108–115.

[102] A. Rukhin *et al.*, “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” *Natl. Inst. Stand. Technol.*, vol. 800–22, no. Rev 1a, p. 131, 2010.

[103] J. Li and M. Seok, “A 3.07µm{\^{}}2/Bitcell Physically Unclonable Function with 3.5{\%} and 1{\%} Bit-Instability across 0 to 80°C and 0.6 to 1.2V in a 65nm CMOS,” in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2015, pp. 250–251.

[104] D. Puntin, S. Stanzione, and G. Iannaccone, “CMOS Unclonable System for Secure Authentication Based on Device Variability,” in *European Solid State Circuit Conference (ESSCIRC)*, 2008, pp. 130–133.

[105] G. Selimis *et al.*, “Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for secure key generation in wireless sensor nodes,” *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 567–570, 2011.

[106] K. Lofstrom, W. R. Daasch, and D. Taylor, “IC Identification Circuit using Device Mismatch,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2000, vol. 46, no. 8, pp. 372–373.

[107] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical One-Way Functions,” *Science*, vol. 297, no. September, pp. 2026–2030, 2002.

[108] J. Guajardo, S. S. Kumar, G. Schrijen, and P. Tuyls, “FPGA Intrinsic PUFs and Their Use for IP Protection,” in *Lecture Notes in Computer Science*, P. Paillier and I. Verbauwhede, Eds. Springer, Heidelberg, 2007, pp. 63–80.

[109] G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in *ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[110] S. S. Kumar, J. Guajardo, R. Maes, G. Schrijen, and P. Tuyls, “The Butterfly PUF Protecting IP on Every FPGA,” in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008, no. 71369, pp. 67–70.

[111] J. W. W. Lee, B. Gassend, G. E. E. Suh, M. van Dijk, and S. Devadas, “A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications,” in *Symposium on VLSI Circuits*, 2004, pp. 176–179.

[112] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. ürgen Schmidhuber, “Modeling Attacks on Physical Unclonable Functions,” in *Proceedings of ACM conference on Computer and Communications Security*, 2010, pp. 237–249.

[113] D. E. Holcomb, W. P. Burleson, and K. Fu, “Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers,” *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, 2009.

[114] R. Maes, P. Tuyls, and I. Verbauwhede, “Intrinsic PUFs From Flip-Flops on Reconfigurable Devices,” in *Workshop on Information and System Security*, 2008, no. 71369, pp. 1–17.

[115] P. Simons, E. Van Der Sluis, and V. Van Der Leest, “Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs,” in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 7–12.

[116] G.-J. Schrijen and V. Van Der Leest, "Comparative Analysis of SRAM Memories used as PUF Primitives," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 1319–1324.

[117] M. Bhargava, C. Cakir, and K. Mai, "Attack Resistant Sense Amplifier Based PUFs (SA-PUF) with Deterministic and Controllable Reliability of PUF Responses," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 106–111.

[118] M. Bhargava and K. Mai, "An efficient reliable PUF-based cryptographic key generator in 65nm CMOS," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, vol. 1, pp. 1–6.

[119] R. Helinski, D. Acharyya, and J. Plusquellic, "A Physical Unclonable Function Defined Using Power Distribution System Equivalent Resistance Variations," in *ACM/IEEE Design Automation Conference*, 2009, pp. 676–681.

[120] D. Suzuki and K. Shimizu, "The Glitch PUF: A new Delay-PUF Architecture Exploiting Glitch Shapes," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2010, pp. 366–382.

[121] D. Ganta, V. Vivekraja, K. Priya, and L. Nazhandali, "A Highly Stable Leakage-Based Silicon Physical Unclonable Functions," in *International Conference on VLSI Design*, 2011, pp. 135–140.

[122] B. D. Choi, T. W. Kim, and D. K. Kim, "Zero bit error rate ID generation circuit using via formation probability in 0.18 μm CMOS process," *IET Journals Mag.*, vol. 50, no. 12, pp. 876–877, 2014.

[123] N. Liu, S. Hanson, D. Sylvester, and D. Blaauw, "OxID: On-chip one-time random ID generation using oxide breakdown," *Symp. VLSI Circuits*, pp. 231–232, Jun. 2010.

[124] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-Proof Hardware from Protective Coatings," in *Cryptographic Hardware and Embedded Systems (CHES)*, 2006, pp. 369–383.

[125] D. Roy, J. H. Klootwijk, N. A. M. Verhaegh, H. H. A. J. Roosen, and R. A. M. Wolters, "Comb Capacitor Structures for On-Chip Physical Uncloneable Function," *IEEE Trans. Semicond. Manuf.*, vol. 22, no. 1, pp. 96–102, 2009.

[126] M. Wan, Z. He, S. Han, K. Dai, and X. Zou, "An Invasive-Attack-Resistant PUF Based On Switched-Capacitor Circuit," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 8, pp. 2024–2034, 2015.

[127] S. Satpathy *et al.*, "13fJ/bit Probing-resilient 250K PUF Array with Soft Dark-bit Masking for 1.94 % Bit-error in 22nm Tri-gate CMOS," in *European Solid State Circuit Conference (ESSCIRC)*, 2014, pp. 239–242.

[128] S. Mathew *et al.*, "A 4fJ/bit Delay-Hardened Physically Unclonable Function Circuit with Selective Bit Destabilization in 14nm Tri-gate CMOS," in *Symposium on VLSI Circuits*, 2016, pp. 248–249.

[129] M. Khayatzadeh and Y. Lian, "A 4 . 28 pJ / access High-Density Average-8T Sub - Threshold SRAM with Reverse Narrow-Width Effect (RNWE) -Aware Sizing," in *IEEE International Conference on Solid-State and Integrated*

Circuit Technology (ICSICT), 2014, pp. 8–11.

[130] M. Bhargava, C. Cakir, and K. Mai, "Comparison of Bi-stable and Delay-Based Physical Unclonable Functions from Measurements in 65nm bulk CMOS," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2012, pp. 1–4.

[131] A. Maiti and P. Schaumont, "Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive," *J. Cryptol.*, vol. 24, no. 2, pp. 375–397, 2011.

[132] M. Alioto and A. Alvarez, "Physically Unclonable Function Database." [Online]. Available: http://www.green-ic.org/pufdb.

[133] G. E. Suh, C. W. O'Donnell, and S. Devadas, "Aegis: A Single-Chip Secure Processor," *IEEE Des. Test Comput.*, vol. 24, no. 6, pp. 570–580, Nov. 2007.

[134] C. Bösch, J. Guajardo, A. R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5154 LNCS, pp. 181–197, 2008.

[135] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *IEEE International Symposium on Information Theory*, 2009, pp. 2101–2105.

[136] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs," in *IEEE International Symposium on Information Theory*, 2009, pp. 1–15.

[137] M. M. Yu, D. M. Raihi, R. Sowell, and S. Devadas, "Lightweight and Secure PUF Key Storage Using Limits of Machine Learning," in *Workshop on Cryptographic Hardware and Embedded Systems*, 2011, pp. 358–373.

[138] M.-D. (Mandel) Yu and S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Functions," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan. 2010.

[139] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, 2008.

[140] Z. S. Paral and S. Devadas, "Reliable and Efficient PUF-Based Key Generation Using Pattern Matching," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2011, no. 978, pp. 128–133.

[141] S. Eiroa, J. Castro, M. C. Martínez-Rodríguez, E. Tena, P. Brox, and I. Baturone, "Reducing bit flipping problems in SRAM physical unclonable functions for chip identification," in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2012, pp. 392–395.

[142] J. Li and M. Seok, "A 3.07µm^2/Bitcell Physically Unclonable Function with 3.5% and 1% Bit-Instability across 0 to 80°C and 0.6 to 1.2V in a 65nm CMOS," in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2015, pp. 250–251.

[143] B. Karpinskyy, Y. Lee, Y. Choi, Y. Kim, M. Noh, and S. Lee, "Physically Unclonable Function for Secure Key Generation with a Key Error Rate of 2E-38 in 45nm Smart-Card Chips," in *IEEE International Solid-State Circuits*

Conference (ISSCC), 2016, pp. 158–160.

[144] M. T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "ARO-PUF: An aging-resistant ring oscillator PUF design," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.

[145] J. Bolus, B. Calhoun, and T. Blalock, "39 fJ/bit On-Chip Identification ofWireless Sensors Based on Manufacturing Variation," *J. Low Power Electron. Appl.*, vol. 4, no. 3, pp. 252–267, 2014.

[146] V. Jathar and L. Colaco, "Wireless Sensor Network - Authentication Protocol for Military Surveillance," *Int. J. Sci. Res. Dev.*, vol. 4, no. 4, pp. 1467–1470, 2016.

[147] Y. M. Yussoff, H. Hashim, and M. D. Baba, "Identity-based Trusted Authentication in Wireless Sensor Networks," *Comput. Res. Repos.*, vol. 9, no. 3, pp. 230–239, 2012.

[148] M. Shiozaki, T. Kubota, T. Nakai, A. Takeuchi, T. Nishimura, and T. Fujino, "Tamper-resistant authentication system with side-channel attack resistant AES and PUF using MDR-ROM," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, vol. 2015–July, pp. 1462–1465.

[149] A. Aysu and P. Schaumont, "Precomputation Methods for Hash-Based Signatures on Energy-Harvesting Platforms," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2925–2931, 2016.

[150] N. Ickes, Y. Sinangil, F. Pappalardo, E. Guidetti, and A. P. Chandrakasan, "A 10 pJ / cycle ultra-low-voltage 32-bit microprocessor system-on-chip," in *European Solid-State Circuits Conference (ESSCIRC)*, 2011, pp. 159–162.

[151] W. Jin, S. Lu, W. He, and Z. Mao, "A 230mV 8-bit Sub-threshold Microprocessor for Wireless Sensor Network," in *IEEE International Conference on VLSI and System-on-Chip*, 2011, no. 2009, pp. 126–129.

[152] B. A. Warneke and K. S. J. Pister, "An Ultra-Low Energy Microcontroller for Smart Dust Wireless Sensor Networks," in *IEEE International Solid-State Circuits Conference (ISSC)*, 2004, pp. 316–317.

[153] Y. Yano, "Take the Expressway to go Greener," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2012, pp. 24–30.

[154] G. Chen, S. Hanson, D. Blaauw, and D. Sylvester, "Circuit Design Advances for Wireless Sensing Applications," *Proc. IEEE*, vol. 98, no. 11, pp. 1808–1827, 2010.

[155] W. Zhao, Y. Ha, C. Hau, and A. B. Alvarez, "Robustness-Driven Energy-Efficient Ultra-Low Voltage Standard Cell Design with Intra-Cell Mixed-V t Methodology," in *International Symposium on Low Power Electronic Devices (ISLPED)*, 2013, pp. 323–328.

[156] ASIP Solutions Inc., "Brownie STD 32 Reference Manual, v1.1," 2008.

[157] ASIP Solutions Inc., "Brownie Micro 16 Reference Manual, v.2.0," 2010.

[158] W. Zhao, Y. Ha, C. H. Hoo, and A. B. Alvarez, "Robustness-driven energy-efficient ultra-low voltage standard cell design with intra-cell mixed-Vt methodology," in *Proceedings of the International Symposium on Low Power*

Electronics and Design, 2013.

[159] D. Bol *et al.*, "SleepWalker: A 25-MHz 0.4-V Sub-mm 7- W/MHz Microcontroller in 65-nm LP/GP CMOS for Low-CarbonWireless Sensor Nodes," *IEEE J. Solid State Circuits*, vol. 48, no. 1, pp. 20–32, 2013.

List of Publications

1. A. Alvarez and M. Alioto, “Security down to the Hardware Level,” in *Enabling the Internet of Things - from Circuits to Networks*, Springer, 2017.

2. A. B. Alvarez, W. Zhao, and M. Alioto, “Static Physically Unclonable Functions for Secure Chip Identification at 0.6-1 V and 15fJ/bit in 65nm,” *IEEE J. Solid State Circuits*, vol. 51, no. 3, pp. 763–775, 2016.

3. A. Alvarez, W. Zhao, and M. Alioto, “15fJ/b Static Physically Unclonable Functions for Secure Chip Identification with < 2% Native Bit Instability and 140x Inter/Intra PUF Hamming Distance Separation in 65nm,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2015, pp. 256–258.

4. W. Zhao, A. B. Alvarez, and Y. Ha, “A 65-nm 25.1-ns 30.7-fJ Robust Subthreshold Level Shifter With Wide Conversion Range,” *IEEE Trans. Circuits Syst. II*, vol. 62, no. 7, pp. 671–675, 2015.

5. W. Zhao, Y. Ha, C. Hau, and A. B. Alvarez, “Robustness-Driven Energy-Efficient Ultra-Low Voltage Standard Cell Design with Intra-Cell Mixed-Vt Methodology,” in *International Symposium on Low Power Electronic Devices (ISLPED)*, 2013, pp. 323–328.