A Power-Aware Study of Iris Matching Algorithms on Intel’s SCC

Gildo Torres, Jed Kao-Tung Chang, Fang Hua, Chen Liu, Stephanie Schuckers
Department of Electrical and Computer Engineering
Clarkson University
Potsdam, New York 13699
Email: {torresg, jchang, huaf, cliu, sschucke}@clarkson.edu

Abstract—Biometric application is paramount in private business, industry, and government. However, as large amounts of data are collected from many different sources, managing such volume of data and developing efficient and effective large-scale operation solutions has become a concern. For example, real-time identification of individuals with the purpose of allowing or denying them access to a specific system or resource is challenging from the performance point of view. In addition, processing large amounts of data would definitely consume lots of energy. In this paper we employ a many-core architecture, the Intel’s Single-chip Cloud Computer (SCC) which supports dynamic frequency and voltage scaling (DVFS), to investigate the power-aware computing and performance enhancement of an iris matching algorithm. This application contains a large degree of parallelism that we exploited by porting it to the SCC. Results of the performance, power, energy, energy delay product (EDP), and power per speedup (PPS) metrics of executing the iris match application under different number of cores, frequency, and voltage settings of the SCC platform are presented. We also analyze how the results for these metrics vary as we change these parameters.

I. INTRODUCTION

Biometrics employ pattern recognition and image processing techniques for the purpose of secure identification and verification. It can be used to allow or deny a person’s access privilege to protected information or physical locations, based on one’s physiological or behavioral characteristics such as face, iris, fingerprint, etc. The increasing security concerns from industry and government lead to the wide adoption of biometrics technology. Large amounts of biometric samples are being collected every day via remote sensing, cameras, radio-frequency identification readers by the health care system, employment, criminal justice, and various security systems. As a result, huge amounts of biometric data are stored in government and private databases, while the number keeps growing at a very fast pace.

It is of great interest and importance to manage and analyze large-scale biometric information in an efficient and effective manner. For example, identifying individual(s) of interest among large datasets at ideally real-time can be a challenging task even with supercomputing power. This fact contrasts with a reality in which most existing biometric systems are reaching their limit of operational capabilities [1]. In addition, running biometric applications with large datasets definitely consumes lots of power and energy. Thus, efficient power management is another challenge. With this in mind, we propose to execute biometrics recognition applications on a many-core architecture in a power-aware/energy-aware manner, at the same time achieving superb performance as well. There are two ultimate goals in our research. First, there is large degree of data parallelism inherited in biometric application. Our research will contribute the work of parallel processing of biometric applications on many-core platform for future cloud computing. Second, we will utilize high-speed on-chip network, advanced power management, and message-based programming features of the many-core platform for the biometrics processing to achieve energy-efficiency.

Our target application in this paper is iris pattern recognition. Iris pattern is considered to be one of the most accurate and stable features for biometric systems [2]. Due to different types of muscles and pigment cells growing to control pupil dilation in early age of human beings, the patterns and color textures shown on iris are unique for each person and each eye. Daugman’s iris recognition algorithm [3] is the most famous one and has been proven to be the most reliable method. In Daugman’s method, there are four steps for an iris matching system: segmentation, normalization, feature encoding, and matching. This research focuses on the matching stage. The stages prior to the iris matching stage acquire the iris image, isolate the noise and eyelid, transform the iris region into a 2-dimension image, and extract the discerning information from iris pattern as a template. Each iris sample is presented by two 2D bit matrices: template matrix and noise mask matrix. We compare two iris samples by calculating the Hamming Distance on the two matrices of each sample and use that as the matching score. Low matching score means two samples are similar and if it is below the threshold, very probably they are from the same person. The Hamming Distance function is compution-intensive, containing mainly simple logical and arithmetic operations.

In our research, we use Intel’s Single-chip Cloud Computer (SCC) as the many-core architecture for executing the iris matching algorithm. The SCC is a 48-core experimental processor created by Intel Labs as a 'concept vehicle' for advancing many-core software research [4]. It consists of 48 Pentium class IA-32 cores put on a 6 × 4 2D-mesh network of tiles, with each tile containing 2 cores. Figure 1 shows the internal architecture of the SCC and the tiles’ structure. Each core in the SCC has 16KB of each separated L1 data and instruction caches, and 256KB of a unified L2 cache. The SCC also supports up to 64GB (only 32GB in our case) of off-chip DRAM accessed through four on-die memory controllers (MC). Each tile in the SCC has an internal block of 16KB SRAM called Message Passing Buffer (MPB), which is
shared by both cores within the tile, and is optimized to support message-passing programming model for communication among all the cores.

For many-core processors’ designs there comes the “coherence wall” problem: as the number of cores increases, the effort invested in maintaining cache coherence will generate considerable overhead that will deteriorate performance and increase power consumption [5]. Bewaring of this problem, the SCC does not offer any built-in support for cache coherency among the cores. If coherency is required, it has to be maintained by software.

Another promising feature of the Single-chip Cloud Computer is that, as it was designed with fine-grain power management in mind, it comes with advanced power management technologies. The programmer has the capability of independently changing the voltage across the entire chip [6]. The cores in the SCC are divided into six voltage domains, each containing a 2x2 array of tiles (a total of eight cores) as shown in Figure 1. Clock may be adjusted at an even finer granularity with each tile on the SCC able to have its own operating frequency. The voltage and frequency islands, together called power domains, enable the programmer to adjust the power/performance settings of different parts of the SCC to particular levels. The frequency may be adjusted (by a frequency divider) to any of the fifteen available values ranging from 100MHz up to 800MHz. There are also seven possible voltage levels in the range of 0.7-1.3Volts at a resolution of 0.1Volts [6]. These features allow the implementation of dynamic voltage and frequency scaling (DVFS) techniques [7] that may continuously adapt to using minimum power at a given moment, resulting in energy savings whenever possible.

In addition to the physical platform, Intel also provides the RCCE library. The RCCE is a many-core communication environment particularly developed for the SCC. This customized message-passing library based on MPI provides several functions, including properly handling the voltage level and operating frequency requests originated at runtime. It also includes support for core numbering in the SCC system, managing communication among the cores, and many other features that can be explored in [6].

There are several reasons for saying that the SCC, as an example of many-core architectures, is a good platform for studying the parallel computation of biometric applications. First, SCC’s high core count may be exploited by the high degree of data-parallelism inherent to biometric applications. Biometric data can be distributed among the cores and processed in parallel. Second, the message-passing model supported by the SCC allows us to have explicit control for intercore communication and therefore address any dependencies that may exist among the data. Third, the DVFS capabilities offered by the SCC allow conducting power-aware computing research. The later is possible by observing the performance and power-related metrics’ results of different voltage and frequency settings, while adapting to minimum power or maximum performance at any given time. Thus, the many-core architecture, message passing programming model, and DVFS features of the SCC, make it an ideal platform for us to investigate the power-aware computing and performance enhancements for biometric application’s execution.

The rest of this paper is organized as follows: we will discuss the related work in Sec.II. The iris matching algorithm and proposed scheme is explained in detail in Sec.III, and the experiments and results are presented in Sec. IV. Finally, the conclusions are drawn in Sec.V along with an outlook for future work.

II. RELATED WORK

As stated previously, the iris recognition algorithm can be classified as computation-intensive having real-time access and processing requirements. If purely implemented into software and executed on a general-purpose processor, the performance will be limited, since it is not efficient as a dedicated hardware. In addition, the program structure of iris matching algorithms shows its code can be highly parallelized. However, the number of cores on a general-purpose processor is normally small and can achieve only a limited degree of parallel execution.

Recently, VLSI technology has advanced consistently, and researchers prefer implementing software algorithms into hardware which has strong computation capability and can process large amount of data in parallel. This approach can meet the real-time performance requirement of iris-matching algorithm. Two representative types of hardware are field-programmable gate array (FPGA) and digital signal processor (DSP).

There are many works for FPGA prototyping of iris recognition. The straightforward way is to implement the time-consuming parts of iris recognition algorithm into hardware for accelerating its execution. For example, we can extract the iris matching code and implement it into hardware using hardware description language. The iris template database can be stored in ROM while the live templates (incoming samples) are stored in RAM. In this way the performance can be improved. In [8], Rakvic et al. did code parallelization for the iris segmentation, template creation, and template matching and implemented on FPGA. They compared the performance of FPGA with that of the state-of-the-art CPU and showed speedups of 9.6, 324, and 19 times, respectively, in the three iris matching parts. This was achieved with moderate hardware usage.

Profiling work helps to identify the time-consuming part of the iris recognition. Raida et al. [9] did computation workload...
profiling and identified Gabor filter, Gaussian mask, canny, and hough transform are the most time-consuming components of the iris recognition algorithm. These parts were then implemented directly into hardware IP or by using tool generator such as C2H compiler. The power consumption, hardware cost (area), design time, and performance for different implementation ways were also compared.

Neural network mechanism is adopted in FPGA implementation to enhance the iris recognition accuracy. Mohd-Yasin et al. [10] employed neural network concept to implement iris recognition on Altera FPGA board to improve the efficiency and accuracy of iris recognition. The neural network architecture has three layers: input, hidden, and output. Based on the back propagation and training process by feeding the iris vector into the network iteratively, the accuracy rate to recognize the iris samples could be lifted up to 88.6%.

Besides FPGA, DSP is used for hardware implementation of iris recognition as well. Miyazawa et al. [11] applied the phase-based image matching [12], [13] based on technique using phase components in 2D Discrete Fourier Transforms (DFT) to two parts of the iris recognition algorithm: image alignment and matching score calculation. Then, they implemented the modified iris recognition algorithm into DSP as the prototype. The proposed implementation has low error rate and meets real-time requirement with low hardware cost.

Different from previous work, in our research the platform is an innovative many-core architecture with abundant hardware resources for parallel computation such as 48 cores, the message-passing buffer (MPB), large memory bandwidth, etc.. The iris samples for comparison can be distributed to many cores and processed in a parallel fashion. In addition, thanks to the provided RCCE library, researchers can directly control the voltage, frequency, and inter-core communication in software. Based on this, our research conducts the power-aware computing for iris matching. We can observe the behavior and trend of performance, power, and energy consumption more thoroughly than the previous research.

III. PROPOSED SCHEME

As mentioned in the Introduction section, this paper focuses on the iris matching algorithm which is the last stage of the iris recognition method proposed by Daugman [5]. It has been proved to be the most reliable method for iris recognition.

A. Iris Matching Algorithm

The iris matching algorithm consists of comparing two samples by calculating their Hamming Distance, which result is known as the matching score and is usually between 0 and 1. The lower the score, the more similar the samples are.

Each iris sample is represented with two 2-dimension matrices of bits (20 rows and 480 columns). The first matrix is the template representing the iris pattern. After sampling the iris region (pupil and limbus), it is transformed into a 2-dimension image. Then Gabor filtering will extract the most discriminating information from iris pattern as a biometric template (matrix). The second one, is the mask matrix representing aspects such as eyelid, eyelash, and noise (e.g. spectacular reflections) which may occlude parts of the iris. When doing

<table>
<thead>
<tr>
<th>Algorithm 1 Iris_Matching Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>template1, mask1, template2, mask2;</td>
</tr>
<tr>
<td>{Input 2-D Matrix with ROW rows and COL columns}</td>
</tr>
<tr>
<td>Hamming_Distance; {Return value}</td>
</tr>
<tr>
<td>template1s, mask1s, temp, mask, result;</td>
</tr>
<tr>
<td>{Intermediate 2D Matrix}</td>
</tr>
<tr>
<td>hd=0, hd_min=1; {Intermediate variable}</td>
</tr>
<tr>
<td>for shift = -8 to 8 do</td>
</tr>
<tr>
<td>template1s ← Rotate_Column(template1, shift)</td>
</tr>
<tr>
<td>mask1s ← Rotate_Column(mask1, shift)</td>
</tr>
<tr>
<td>temp ← Matrix_XOR(template1s, template2)</td>
</tr>
<tr>
<td>mask → Matrix_OR(mask1s, mask2)</td>
</tr>
<tr>
<td>result ← Matrix_AND(temp, mask)</td>
</tr>
<tr>
<td>hd ← number of 1s in Matrix result</td>
</tr>
<tr>
<td>if hd &lt; hd_min then</td>
</tr>
<tr>
<td>hd_min ← hd</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>Hamming_Distance ← hd_min</td>
</tr>
<tr>
<td>return Hamming_Distance</td>
</tr>
</tbody>
</table>

Hamming Distance calculation, both matrices need to be used for each sample.

Algorithm 1 shows the flow of Daugman’s iris matching algorithm. The main body of Hamming Distance calculation is a 17 round for-loop structure with the index shift from -8 to 8. Suppose the matrices of the first sample are template1 and mask1, and the matrices of the second are template2 and mask2. In each iteration, first, template1 and mask1 will be left-rotated (if shift<0) or right-rotated (if shift>0) for 2 * abs(shift) columns as template1s and mask1s. Next, template1s will be XORed with template2 and mask1s will be ORed with mask2, and the result matrices are temp and mask, respectively. Finally, the matrix temp will be ANDED with matrix mask into the matrix result. So the Hamming Distance for this round (variable hd) is calculated as follows: suppose the number of 1s of the result matrix is bitsdiff, and totalbits is the total number of the 2D matrix entries (20 * 480) minus the number of 1s in matrix temp, hd is calculated as dividing totalbits by bitsdiff. The minimum value of all 17 iterations hd is the calculated Hamming Distance on two samples as matching scores, showing their similarity. This algorithm calls Rotate_Column function (described in Algorithm 2). Matrix_XOR, Matrix_OR, and Matrix_AND functions (bit-wise operations of two matrix).

B. Proposed Scheme

We conducted the power-aware computing of iris matching algorithm on Intel’s SCC because firstly, the iris matching algorithm contains large amount of data parallelism, enabling the SCC’s cores to process the biometric data in parallel to enhance the performance. Secondly, the SCC DVFS’s features help the researcher to dynamically control the voltage and frequency while watching the power/energy consumptions.

Next, we show the steps we followed for porting this algorithm to Intel’s SCC architecture.
Algorithm 2 Rotate_Column

{Two Input Parameters: Matrix MAT and variable shift} 
{Return Matrix MAT} 
if shift ≤ 0 then 
   rotate left MAT for 2 × abs(shift) columns 
else 
   rotate right MAT for 2 × abs(shift) columns 
end if 
return MAT 

1) Porting Work: We converted the original Matlab code into C. Based on the RCCE message-passing programming model, we modified the code to port on SCC. We also tested its correctness and validity against the original results obtained with Matlab.

2) Parallelization Work: In the original version, each iris sample is compared against the entire dataset, following an N vs N approach. The objective is identifying the pair of samples having the lowest Hamming Distance (matching score) within the dataset. When porting this application to the SCC, we tried exploiting the most possible parallelization from the many-core platform. Our approach consisted on distributing the total number of comparisons among all the participating cores, in a way that each core can independently complete all the comparisons assigned to them without incurring in too much communication overhead. As the cores complete all their assigned comparisons, they will send their best local matching score to core 0 (master core). Upon completing its own set of comparisons, and after receiving the results from all the participating cores, core 0 will determine the absolute best match among all.

At the beginning of the execution, each core will individually create the list (only by name) of all the samples included in the dataset. Then, each core, based on the indices of the samples pairs and their own core IDs, will determine which comparisons to complete. This approach guarantees each core having an even workload, also guarantees that not pair of samples is compared more than once, and the communication overhead is kept to a single message being sent from each core to the master core at the end of the computation.

3) Performance and Energy Evaluation: By testing different settings, such as varying the number of cores, memory usage, frequency, and voltage, we studied the performance, power, and energy behavior of the iris matching application in the SCC. In our analysis we included five different metrics which are detailed in the next section.

We designed two experiments. Within the first one, we tried to observe how increasing the number of cores, with the default values of frequency and voltage, affects the performance and power behavior of the execution in what we named the Baseline approach. In this first experiment, we also wanted to study the impact in performance due to a possible bottleneck that might occur in the memory controllers (MC) when several samples are loaded into the cores. We named this as the MC aware approach.

The second experiment, called the Voltage Domain (VD) aware experiment, aimed to studying the impact in performance and power of applying different voltages and frequencies settings while also varying the number of cores. In this experiment, we followed the goal of trying to minimize the number of power domains involved in the computation. Thus, while the participating power domains were adjusted to mid or high power states, the left out domains were kept at the minimum power state so that the total power consumption can be reduced.

IV. Experiment

We conducted the power-aware computing experiments of the iris matching program by exploiting SCC’s many-core and DVFS features. The results presented in this section correspond to executing the iris matching application over a database containing over 700 samples. Each iris sample is compared against the entire dataset following an N vs N approach, resulting in a total of 164451 comparisons.

The first experiment is about the data parallelism. We can see how different metrics (performance, power, energy, energy-delay-product (EDP), and power per speedup (PPS)) vary as we increase the number of cores. Also in this experiment we study the impact of memory bandwidth on the performance of the iris matching algorithm. The second experiment is similar to the first one, but in addition to varying the number of cores, we also vary the frequency and voltage parameters in our study. We would like to investigate how different metrics are affected by different combinations of frequency, voltage, and the number of cores.

The first metric is the performance: the execution time under different configurations. Nevertheless, considering that running biometric applications (and many other scientific applications) with large datasets inevitably will consume considerable amounts of power and energy, the scope of this research is not only to target performance but also the behavior of power, energy, energy-delay product (EDP), and Power per Speedup (PPS) metrics. Usually different configurations would impact different metrics, so it is very uncommon to have one configuration that best satisfies all the metrics at the same time. In order to achieve higher performance, executing at a higher frequency (and consequently voltage) for the SCC would imply higher power consumption. CMOS circuits often have the ability to trade energy for performance, and it is quite difficult to improve both energy and performance simultaneously [14]. Therefore, power and energy consumptions are two important metrics to observe by applying different voltage/frequency/number of cores settings.

Aside from the above metrics, we will also measure the energy-delay product (EDP). From the system point of view, minimizing the execution time, the power or the energy consumption may be the first priority. However, even if the total energy is minimized, the user may not be satisfied with extended system response time and vice versa [15][16]. Some researchers have argued that the process normalized EDP, expressed as Joule × Second, is a relative implementation-neutral metric [14]. Because of this, the EDP is also considered.

Joule × Second, is a relative implementation-neutral metric [14]. Because of this, the EDP is also considered...
as a metric to evaluate the overall implementation of our application on the SCC platform.

The last metric we used in our study is the Power per Speedup (PPS). It was proposed by [17] as an indicator to measure energy efficiency. Different from the common energy metric, PPS is more intuitive for watching energy savings when performance is not the first priority. As mentioned by [17], this metric tells us how much power the system needs for each performance unit, such as the speedup.

There are two key aspects of the SCC platform that allowed us to accomplish that. The first one is the capability of measuring the power consumption of the SCC throughout the execution of the application (at the granularity of the entire chip). The second aspect is the advanced power-management capabilities that come with SCC which allows the programmer to dynamically control the frequency and voltage (and therefore power) state of different parts of the chip.

We describe the different experiments completed in the following two subsections.

A. Baseline and Memory Controller Aware (MC aware):

1) Experiment: We know that the iris match application is an embarrassingly parallel algorithm, where each comparison can be completed independently and in parallel. Increasing the number of cores employed for execution would positively impact performance. Having all the cores on the SCC configured with voltage and frequency of [533MHz - 1.1V] (default configuration), we experimented varying the number of cores from 1 to 48 to complete the execution in what we called the Baseline experiment.

The second part of this first experiment, referred as the MC aware, was based on our supposition that a bottleneck could occur in each memory controller (MC) when several cores at the same time try accessing memory for loading their data files. We suspected that due to saturation of the memory bandwidth of the MCs, the performance results may be downgraded.

The SCC has four memory controllers (see Figure 1) that divide the 6x4 2D-mesh of tiles into four memory domains, where each controller serves to the cores inside its domain. The method for specifying which cores are to be allocated for executing an application is determined by a host file passed to the application when it is launched. The default method selects the cores to be used based on their IDs (see Figure 2(a)), not on their physical location within the 6x4 array of tiles, which is not efficient respecting to the memory controller’s location.

In this setup, the voltage and frequency settings of the cores were configured similarly to the Baseline with [533MHz - 1.1V]. We also experimented varying the number of cores from 1 to 48 to complete the execution. The only difference is in the way the cores were allocated. This time, the cores involved on each run from 1 to 48 cores, were evenly distributed amongst the four memory controllers as shown in Figure 2(b).

2) Results: In this subsection we present and analyze the results obtained from running the iris matching algorithm by using different strategies and configurations for the SCC.

Although the Single-chip Cloud Computer has many features supporting high performance computing, advanced power-management, and message-passing model, this does not apply to the microarchitecture of each core, which is based on a very simple architecture that was first introduced by Intel in March 1994 as the Pentium P54C version of the Pentium class. Although a few modifications were done for this platform, it continues to be a simple architecture, lacking out-of-order execution among a few other aspects incorporated within modern processors. In the Baseline experiment, we present and analyze the scaling behavior that SCC can offer to iris matching algorithm respecting to performance, power, energy, EDP, and PPS compared to single SCC core. We incorporate the results of the MC aware into the Baseline experiment to show the impacts of memory bandwidth. The data presented in Figure 3 to 7 provide insight into the effectiveness of the different configurations.

Figure 3 presents the performance results obtained from running both the Baseline and MC aware approaches. It shows how effectively the iris matching algorithm scales for the SCC, showing an almost linear speedup as the number of cores increases from 1 to 48. Both cases appear to have very similar behaviors, showing that there is no performance penalty associated with the way the cores are allocated with respect to the memory controllers, which disproves our initial suspicions. The results included in Figure 3 demonstrate that this particular application does not saturate the bandwidth offered by the memory controllers of the SCC.

The average power consumption for each execution is represented in Figure 4. It can be observed how it follows a linear increase as the number of cores grows. For both cases the entire platform is configured with the default values of voltage
and frequency [533MHz - 1.1V]. There are no differences between the cores that are executing the iris match application and the cores sitting idle. The power increase is purely due to the computing intensity of the cores involved in the execution. The average power consumption value ranges from 46 watts using one core up to 78 watts when all the 48 cores are used.

After observing almost identical results for the performance and power of the Baseline approach and the MC aware approach, we can expect similar behaviors for the energy, EDP, and PPS metrics, due to their direct dependency on power and performance. These results are presented in Figures 5, 6, and 7.

The energy consumption and the Energy Delay Product metrics results show the trend in which using more cores would result in decreasing both energy consumed and EDP values. The reductions in energy and EDP for increasing the core count come as a result of the improvement in performance being more significant that the power increase incurred from Figure 3 and 4. This trend reaffirms how the energy consumed by a system is dependent on the balance between performance and power, not simply processor speed or power dissipation alone. The energy consumption goes from around 725457 joules when only one core is used, down to 27720 joules for full chip utilization. In Figure 7, we see that the PPS decreases from around 46 to 2 as maximum number of cores are used, which shows that the power needed for each performance unit decreases as the number of cores is increased.

B. Power-aware/Energy-aware configuration:

1) Experiment: In this experiment we wanted to explore how different configurations of the SCC platform may result in different behavior of the metrics mentioned above. We mainly changed three parameters for completing the simulations. These are total number of cores, frequency, and voltage. For the total number of cores utilized, similarly to the previous approaches, we executed the application using from 1 up to 48 cores. Frequency and voltage were modified at the granularity of a power domain (8 cores) and two possible combinations were explored [533MHz - 0.8V] and [800MHz - 1.2V]. We also used a different method for selecting which cores are to be allocated for a given number of cores. This time the cores are mapped in a way that minimizes the number of power domains hosting cores involved in the computation, as shown in Figure 2(c). This allows the power domains that are left out of the computation to be adjusted at their minimum possible power state [100MHz - 0.7V] following the objective of reducing the power consumption of the SCC while executing the iris match application.

2) Results: Figures 8 to 12 include the performance, speedup, power, energy, EDP, and PPS results obtained from running the third approach using two different sets of frequency and voltage [533MHz - 0.8V] and [800MHz - 1.2V]. In Figure 8, it can be seen how the application performance also scales very well for both configurations. The speedup increases almost at a 1:1 ratio with respect to the number of cores being used, having some fluctuation when reaching the maximum number of SCC cores. The setting running at higher frequency
Figure 8: Performance Results for Different Frequency and Voltage Settings

Figure 9: Power Results for Different Frequency and Voltage Settings

Figure 10: Energy Results for Different Frequency and Voltage Settings

Figure 11: EDP Results for Different Frequency and Voltage Settings

Figure 12: PPS Results for Different Frequency and Voltage Settings

(800MHz) shows the best performance of both for all cases.

The contrast for the average power consumption is represented in Figure 9. Such a significant difference is due to the distinct power states between the active power domains hosting the cores involved in the iris match computation. This trend mainly comes from the differences in frequency and voltage between both settings. Also an interesting thing is the steps-like patterns that both cases show (more noticeable for the high power case). If close attention is paid, the reader will notice that the steps happen to be at core counts 8-9, 16-17, 24-25, 32-33, and 40-41. This can be attributed to the activation of new power domains (hosting eight cores each) joining the computation and adjusting their voltage and frequency from [100MHz - 0.7V] to the operating values of [533MHz - 0.8V] or [800MHz - 1.2V], and therefore increasing their power consumption. Remember that the power is controlled at the granularity of an entire power domain, which contains eight cores, and the domains that are not active are adjusted at the minimum values of voltage and frequency. The average power consumption ranges from 20 to 50 watts and 26 to 117 watts for the [533MHz - 0.8V] and the [800MHz - 1.2V] configurations, respectively.

Results for energy consumption are presented in Figure 10. It illustrates both schemes experiencing a fast drop from 1 to around 8 cores. After this point, each configuration maintains a slightly different pattern from there up to 48 cores. This results in different values between both configurations at opposite ends of the spectrum. The [800MHz - 1.2V] setting is more energy efficient for low core utilization, while it is outperformed by the [533MHz - 0.8V] scheme at the high core end. The reason for this behavior is that energy is equally affected by both power and performance: Figure 8 shows how the total difference in performance between both settings decreases while Figure 9 demonstrate the power gap between them getting larger as the number of cores grows. It can be seen how in the x-axis of Figure 10 (Energy), starting from around 8 cores and above, the energy consumption of [533MHz - 0.8V] is lower than the [800MHz - 1.2V] setting. Another observation that can be made is that after 16 cores, the energy consumed by the [800MHz - 1.2V] setting fluctuates within a constant range without suffering any significant improvement or worsening. This is due to the little performance improvement added by each core contrasted to the steady power rise of high core counts, especially when a new power domain is added to the computation. Overall, total energy consumption for this experiments oscillates from around 332998 joules when only one core is used at [533MHz - 0.8V], down to 18479 joules for the same voltage and frequency values when using 47 cores.

EDP results shown in Figure 11 demonstrate a trend in which utilizing more cores would result beneficial for this metric. As we increase the number of cores, the energy consumption of both settings decreases as well. It can also be observed how the [800MHz - 1.2V] configuration offers better EDP values than [533MHz - 0.8V] for low number of cores, while the [533MHz - 0.8V] performs better for higher numbers. The reason is that the value of the EDP metric is
Table I: SCC Best Configuration for Different Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Frequency</th>
<th>Voltage</th>
<th>Number of Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>800MHz</td>
<td>1.2V</td>
<td>46</td>
</tr>
<tr>
<td>Power</td>
<td>533MHz</td>
<td>0.8V</td>
<td>47</td>
</tr>
<tr>
<td>Energy</td>
<td>533MHz</td>
<td>0.8V</td>
<td>47</td>
</tr>
<tr>
<td>EDP</td>
<td>533MHz</td>
<td>0.8V</td>
<td>47</td>
</tr>
<tr>
<td>PPS</td>
<td>533MHz</td>
<td>0.8V</td>
<td>47</td>
</tr>
</tbody>
</table>

given by $\text{Energy} \times \text{delay}$. If we go back to analyze Figure 8 and Figure 10 we can see that as the number of cores increases, the differences in delay between both settings is reduced, while at the same time the energy consumption of the [800MHz - 1.2V] setting surpasses the [533MHz - 0.8V] when the core count is higher than 24. Thus, starting from around 24 cores, the [533MHz - 0.8V] starts to have lower EDP values than the [800MHz - 1.2V] setting. When looking at these results in numbers, the best option for minimizing both Energy and EDP values is configuring the system to run at [533MHz - 0.8V] and use 47 cores to complete the computation.

Figure 12 presents the behavior for the PPS metric. As explained above, this metric measures how much power is required by the system for each speedup unit achieved, whereas smaller means better. As it can be seen, the biggest reductions occur for low core counts in both cases, nevertheless it is remarkable that adding more cores close to full chip utilization still adds some improvements. The [533MHz - 0.8V] configuration offers best PPS results for all core counts. Overall, full chip utilization represents the best case for both configurations if minimizing PPS is the primary goal.

Table I shows the best configurations of the SCC for each metric. As mentioned above, it is very unusual to find one configuration that best satisfies all the metrics simultaneously.

V. CONCLUSION

This research focused on studying the performance and power-aware/energy-aware computing aspects of a biometric application. In particular, we selected an iris matching algorithm as a case study, which is the final stage of the iris recognition process. We modified and ported this application to Intel’s SCC many-core platform. The high core count and DVFS features of the SCC allowed us to achieve solid performance enhancements while studying the variations on power, energy, EDP, and PPS metrics, under different settings.

For all cases, the speedup increased almost linearly with the number of cores involved in the computation. Average power consumption also increased steadily when more cores were used, ranging from 20 to 50 watts and 26 to 117 watts for the [533MHz - 0.8V] and the [800MHz - 1.2V] configurations, respectively. The total energy consumption and EDP metrics followed a decreasing trend while more cores are added to the computation mainly because the performance improvement outweighs the power consumption increase. It proved the [800MHz - 1.2V] setting to be the most energy and EDP efficient for low core utilization, while it is outperformed by the [533MHz - 0.8V] scheme at the high core end. The PPS metric also followed a decreasing trend as the core count increases where the [533MHz - 0.8V] setting offered the best of both settings. From the conducted experiments, we also demonstrated that the memory bandwidth offered by the SCC memory controllers is not saturated by this particular application. Therefore, different core allocation criteria should not impact the performance results. The results as a whole, showed the ability of the iris matching application to efficiently scale and fully exploit the capabilities offered by Intel’s Single-chip Cloud Computer.

As for future work, we will apply our methodology to other stages of the iris pattern recognition algorithm. We will also explore more efficient methods for finding the best configurations of the SCC for different metrics. In addition, we are also planning to incorporate other biometric applications, such as fingerprint and facial recognition to further take advantage of the SCC capabilities.

REFERENCES