Chapter 2

Review of Literature
- Work published so far

2.1 Introduction

The overall goal of National Institute of Standards and Technology (NIST) was to develop a Federal Information Processing Standard (FIPS) that specifies an encryption algorithm capable of protecting sensitive government information well into the twenty-first century. Five candidate algorithms that were finalized were MARS, RC6, Rijndael, Serpent and Twofish. All these five finalists had following features in common: they are iterative, uses a symmetric block cipher that mixes secret sub keys in each of the iteration round. Whereas these sub keys were derived from the initial input cipher key along with the data block.

The overall assessments of the finalists were based on the following analysis [1] (James Nechvatal, 2000):

1. General Security
2. Software Implementation
3. Restricted Space Environment
4. Hardware Implementations
5. Attacks on Implementations
6. Encryption Vs. Decryption
7. Key agility
8. Versatility and Flexibility
9. Potential for Instruction level Parallelism

Rijndael proved itself to be consistently a good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback mode. The NIST finally proposed Rijndael as the Advanced Encryption Standard.
The publications based on AES can be found in various Journals, Proceedings of Conference and Symposia. Relevant journals and other resources include Springer Journal of Cryptographic Engineering, IEEE Transactions on Very Large Scale Integration jointly published by IEEE Circuits and Systems Society and IEEE Computer Society, Springer Lecture Notes in Computer Science (LNCS), Journal of Microprocessor and Microsystems, Integration: The VLSI Journal and IET Information Security etc. There are many conferences and symposiums conducted regularly, wherein AES implementation is considered as central theme. Proceedings of some of the important conferences like Cryptographic Hardware and Embedded Systems, CRYPTO, ASIACRYPT, EuroCrypt, International Conference on Applied Cryptography and Network Security (ACNS), and INDOCRYPT, have featured AES implementations and architecture designs.

2.2 Hardware Implementation of AES

Many researchers have implemented AES on hardware platforms and published their work through various articles in Journals and Conference Proceedings. After reviewing these publications, we have attempted to classify them into different categories. The discriminators used for this classification were: architectural style, strategy of implementation and platform on which implemented.

Various architectural styles consist of Rolled architecture, Pipelined architecture, Partial Sub-Pipelined architecture and Systolic architecture. A rolled architecture implementation normally results into a low area design. This is because the hardware of a round is iteratively reused for all other rounds. The drawback of this kind of implementation is that they have large system latency. The pipelined architecture provides larger throughput by implementing independent hardware blocks for each round separated by pipelining registers. Such architecture generates cipher text after each clock cycle, except with an initial latency, equal to the number of pipelining stages. The drawback of this kind of implementation is that they employ large amount of hardware resources and hence larger area. The midway solution to rolled and fully pipelined architecture is partial sub-pipelining. The partial sub-pipelined architecture is similar to rolled architecture, except that it employs sub-pipelined registers after every
process stage within a round. This results into a higher throughput than the rolled by as many times as the number of sub-pipelined stages within a round. In Systolic architecture the whole implementation is looked upon as network of processing elements (PEs) that rhythmically compute and pass data though the system. As a result, systolic architectures feature modularity and regularity, which are important properties for any VLSI design.

The AES algorithm involves few computations while performing SubstituteByte, MixColumn and Key Expansion operations, which is dealt in detail in next section. These computations can be performed on the fly (OTF) every time or, a Look-up-Table (LUT), consisting of pre-computed values for each of the sub-processes, can be employed. The OTF computational strategy is more or less a memoryless implementation but with higher complexity. Whereas the LUT based design requires lots of memory to store the pre-computed values. In particular, the SubstituteByte operation involves computation of multiplicative inverse and affine transformation of a byte and is the most complex out of all sub-processes. A pre-computed table of multiplicative inverse and affine transformation of all possible 256 bytes combinations is called S-Box. Obviously the computation time required for OTF is quite higher than LUT, because in LUT based approach, the memory access time is the only critical time. Either of these two strategies, OTF or LUT, can be followed, while implementing AES using any of the architectural styles, mentioned in above paragraph.

The AES evaluation process involved the efficient implementation of hardware platforms. The researchers from Academic groups first implemented AES on FPGAs [4] [5] [6] [7] and the researchers from National Security Agency and Industry, provided the first implementations targeting ASICs [8] [9]. Based upon the outcomes of these evaluation processes, AES has been declared suitable for FPGA as well as ASIC platforms.

The following sections review various hardware architectures that are potentially suitable for AES implementations. These reviews are subdivided into categories such as small area and high speed implementations. The review of some selected few publications are presented in the next sub-section, each one representing a different strategy of implementations and are the ones which are more relevant to our research statement.
2.2.1 Low Area Implementations of AES

Hernandez et al., [10] in 2005, presented novel minimum cost architecture for AES. The architecture employed a bit-serial approach, such that it can be used as a coprocessor integrated with any inexpensive microcontroller in a System on Chip (SoC) platform. The design was prototyped on FPGA and achieved 0.37Mbps at 510MHz with only 10K gate count. Later on, in 2008 [11], they modified their earlier design with a more effective controller based design, with a dedicated microinstruction command format.

Kosaraju et al., [12] in 2006, proposed a high throughput and area efficient architecture for 128 bit AES. In their work they have achieved area efficient implementation using rolled architecture, wherein they have performed one round of encryption/decryption in each clock cycle, getting throughput of 232.7 Mbps at 20MHz clock frequency using 350nm CMOS technology. A similar attempt by Mangard et al., [13] involves the rolled architecture using multipliers while performing MixColumn operations. The architecture of Kosaraju et al., has replaced this multiplier by XOR gates and multiplexers.

Huang et al., [14], implemented 128-bit AES on multiple platforms. They proposed two architectures, namely, as AES1 and AES2. AES1 was regular rolled architecture performing encryption and decryption both without functional integration. While AES2 integrated the S-Box and InvS-Box, also the MixColumn and preprocess InvMixColumn were integrated together to perform encryption and decryption. The designs were implemented on 350nm, 180nm and VirtexE FPGA. A maximum of 641Mbps was achieved at 55MHz clock frequency.

Hamalainen et al., [15], proposed an only encryption, AES-128 hardware core implemented on 130nm CMOS technology, area optimized and consuming only 3.1K gates. The architecture consists of five components: byte permutation unit, MixColumn multiplier, parallel to serial converter, S-Box and key expansion unit. All these components were using 8-bit data width; hence each round consumed 16 clock cycles. The throughput of the design at the maximum clock frequency of 153MHz was 121Mbps.

Burns et al.[16] in their proposed architecture based on normal basis rather than polynomial basis in Galois Field (GF) showed the results in terms of lesser power
consumption. A look up technique was presented to incorporate register files to contain the intermediate AES state. A unique use of squaring operation and lookup, instead of multiplicative approach resulted in more efficient architecture. A throughput of 156Mbps achieved at 132MHz clock frequency, with 4.8K gates, employed on 350nm CMOS technology, are some of their achievements.

Po-Chun Liu et al., [17] in 2009 proposed a combined encryption and decryption AES core for all key sizes. The use of OTF key expansion unit and highly integrated encryption and decryption data path have resulted into a throughput of 1.69Gbps at 131.8MHz on a 90nm CMOS technology. The area of the chip was 15,577 equivalent gates. The AES crypto core consisted of control unit, key expansion unit and integrated data process unit.

Qingfu Cao et al., [18] also proposed a high throughput cost effective implementation of AES supporting both encryption and decryption for all key sizes. The performance of this architecture was evaluated on 180nm CMOS technology and was also verified on FPGA. Composite field arithmetic was employed for SubstituteByte and InvSubstituteByte operations to reduce the area requirement. OTF key expansion capable to support 128, 192 and 256-bit key sizes generated round keys of 128 bits per cycle. A similar OTF key expansion was suggested by Monjur Alam et al. [19], but have generating the 128-bit round key in four cycles of 32-bit each.

Wang et al., [20] in 2010, implemented rolled architecture for 128-bit AES using 180nm CMOS standard cell technology from Taiwan Semiconductor Manufacturing Company (TSMC). The design featured a 4-stage pipelined S-box, which increased the speed by 2.1 times the encryption only design of Hamalainen et al.[15]. The design achieved throughput of 117.9Mbps at a maximum frequency of 253MHz, employing 2730 equivalent gates.

A low power 128-bit AES was implemented by Tim Good et al., [21]. The architecture is based on 8-bit data path and concentrated on low power implementation. Even though throughput of the design was not impressive, it was the first of its kind on 130nm technology consuming 692nW at 100 KHz clock frequency. The major drawback of the design was large system latency of 356 cycles. Hence, authors preferred power-latency-area product to compare the performance with other published designs.
A good number of publicized literatures are available on FPGA based implementations using Rolled architecture. As we were interested in the architecture for implementation and its performance, even these literatures were reviewed and summarized below.

Rouvroy et al.,[22], in 2004, suggested a compact and efficient encryptor and decryptor modules for Spartan-III and Vertex-II FPGA implementation. In their design the low area was achieved by employing T-Boxes and hosting them on the Block RAMs available on board of FPGA. T-Boxes are the pre-computed look-up-tables which represent the combined operation of SubstituteByte and MixColumn transformations. The design generates the cipher text at a throughput of 208Mbps on Spartan-III device using 163 slices and 358Mbps on Vertex-II device using 146 slices. Two Block RAMs were used for T-Boxes and one Block RAM for storing round keys while key expansion.

Brokalakis et al.[23], proposed an area efficient high throughput architecture implemented on Vertex-II FPGA device. The architecture was designed to be a good candidate for integration in SoCs, having 128-bit data access and other control signals. A processing core performing encryption of 128-plain text was designed in conjunction with Key logic unit for key expansion and system control unit to generate various control signals internally. The design achieved 1.94Gbps at 159.2MHz clock and using 1122 CLB slices and 8 Block RAMs. Compared to other implementations, the design demonstrated the highest throughput and smallest area to performance ratios.

A bit level and byte level structures were proposed for MixColumn and InvMixColumn operations by Hsiao et al. [24]. These structures when used in AES implementation, shared maximum possible XOR gates between MixColumn and InvMixColumn operations. Similarly the SubstituteByte and InvSubstituteByte operation also shares common hardware, achieving 685Mbps on Vertex-II FPGA and 1.16Gbps on 180nm CMOS technology.

Viktor Fischer et al.[25], analyzed the basic operation of MixColumn and InvMixColumn operations and proposed possibilities for resource sharing at Bit level, Byte level and Word level by parallel and serial decomposition of the polynomials. This work initiated an altogether new aspect of area efficient AES implementations.

Liberatori et al. [26], presented a novel 64-bit architecture to implement 128-bit AES on Spartan-III FPGA. The architecture was developed using a processor design approach,
wherein a FSM based Control Unit was employed to manage the clock cycles for a 64-bit data path. The throughput achieved was 224.12Mbps using only 822 CLB slices of Spartan-III FPGA device.

Chih-Peng Fan et al. [27], implemented 128-bit AES using sequential architecture, wherein Content-Addressable Memory (CAM) was employed for realization of SubstituteByte operation and novel hardware sharing architecture for MixColumn operation. This sequential design achieved an 876Mbps at a clock frequency of 75.3MHz when implemented on Vertex-II FPGA. The concept of CAM based SubstituteByte was introduced by H. Li et al.,[28] in 2005.

Monjur Alam et al.[29], proposed an architecture which can process all 9 possible combinations of key and data lengths. The computations of SubstituteByte and InvSubstituteByte are done in GF(((2^3)^3)^3) rather than GF(2^n). The use of composite field arithmetic reduces the complexity. The design was clocked at 135MHz and achieved a throughput of 432Mbps

In 2010 Opritoiu et al.,[30], proposed architecture, based on 128-bit parallel data path for 128-bit AES, encryption and decryption both. In this architecture the SubstituteByte and InvSubstituteByte were performed using Galois Field multiplicative inverter, which was commonly shared between Encryption and Decryption data paths. The key generation unit avoided the pre-computation of entire key set by delivering the last encryption key as the initial key, while decryption. The design was implemented on Altera’s Cyclone II EP2C35 FPGA and achieved throughput of 805.24Mbps at 70MHz clock frequency, using 6608 Logic Elements.

Krukowski et al., [31], have designed a cryptographic unit and implemented on a Spartan-3 XC3S1000 device. The design pre-generated all the 11 round keys in 44 clock cycles with 32-bits generated in each cycle and stored them in the CLB registers of FPGA.
2.2.2 High Speed Implementation of AES

The high speed implementations of AES may involve the use of concurrency in the algorithm, widening of data path, reducing latency, reducing inter path delay differences and other adhoc solutions like reducing fan-out and fan-in of the gates etc. Primarily in this category of implementations, the preference is given to the throughput rather than area of implementation. Reviewing of the literature based on high speed implementations can be helpful in further enhancing the throughput of the designed architectures. This section presents the literature review for those designs which have achieved high throughput.

A novel architecture named as Twisted Binary Decision Diagram (TBDD) was suggested by Sumio Morioko and Akashi Satoh [32]. In this architecture, to reduce the propagation delay of the S-Box, TBDD was employed, wherein the fan-out of signals is decreased. The resultant S-Box is two times faster than the conventional S-Box implementations. The design was clocked at 880MHz and achieved 11Gbps throughput when implemented on 130nm CMOS technology.

Akashi Satoh [33], proposed a sequential Galois Counter Mode (GCM) architecture with fully pipelined AES implementation. A fully pipelined architecture un-rolls all the required rounds and inserts register in between the rounds. The data path width of the architecture was 128-bit and supported all key sizes. He also suggested a pipelined loop architecture, wherein a block consisting of the four pipelined rounds was rolled over iteratively. The fully pipelined design achieved highest throughput of 42.7Gbps with 297,000 gates and a pipeline-loop (Rolled Pipeline) achieved, 6.4Gbps with only 73,000 gates on 130nm CMOS technology using standard cell libraries.

Xinmiao et al. [34], in their implementation employed sub-pipelining of the unrolled design of AES-128. In sub-pipelining, the pipelining registers are inserted in between the sub-processes within the round and as well as between the rounds of the loop-unrolled version of the AES algorithm. The authors introduced 7 substages in each round including the OTF computational block of SubstituteByte operation. The design was implemented on Vertex architecture with a frequency of 168.4MHz and achieved a throughput of 21.56Gbps. The performance of this implementation was claimed to be 79% more efficient than the previous implementation by Saggese et al. [35] in terms of
throughput per slice. The similar kind of implementation was done by Hodjat et al., [36] in the same year 2004, on Vertex-II device and achieved even better results than Xinmiao et al. They achieved the same throughput as in [32] but employing less number of slices. The throughput per slice of Hodjat et al. is 4.2Mbps/slice.

Zambreno et al.,[37], through their experimentation, demonstrated the significance of two kind of pipelining, namely as Inter round and Intra round. They selected inter round pipelining stages as 1,2,5 and 10, while the intra round pipelining stages as 0,1,2 and 3. A total of 25 combinations were proposed with resultant throughput and efficiency factor (throughput per slice). The combination of 10 inter round pipelining stages with 3 intra round pipelining achieved the highest throughput of 23.57Gbps using 16,938 slices on a Vertex-II FPGA. The architecture employed for implementation can also be termed as fully sub-pipelined architecture. A similar fully sub-pipelined architecture, but with 32-bit data width was implemented by Tanzilur Rehman et al.,[38] in 2010 and achieved 3.22Gbps using only 6605 slices on a Spartan-III FPGA. Even though the performance is below the earlier implementations, the authors have presented a statistics of different combinations of strategies applied. They were namely as, Composite field SubstituteByte for both data path and key expansion, Composite field SubstituteByte for key expansion and S-Box for data path, Composite field for SubstituteByte data path and S-Box for key expansion, Look-up table for both key expansion and SubstituteByte and finally sub-pipelined SubstituteByte. All of these combinations employed, fully pipelined inter round blocks.

A novel architectural transformation technique to map 32-bit wide AES algorithm to byte vector serial systolic architecture was proposed by Farhan et al.[44]. The complex matrix multiplication component in MixColumn operation of AES was implemented using systolic architecture. The design achieved a moderate throughput of 41Mbps for encryption and 37Mbps for decryption while utilizing 236 and280 slices respectively on Vertex-II FPGA.

High throughput implementations of AES on ASIC platforms were also reported through various publications. Architectures like fully pipelined and iterative inner-round pipelining, in which the look-up tables were replaced by OTF computations to achieve better performance in terms of throughput per Kilo-gates.
Tsung-Fu et al. [39], iteratively used the internally pipelined data round, for all the ten rounds of AES-128 with a data width of 128-bits for all key sizes. The design was implemented on 350nm CMOS technology and achieved highest throughput of 2.381Gbps, for 128-bit key size. Compared to the reported publications of that period, this implementation had highest throughput/gate-count as 41.49Mbps/K-gates.

Hodjat et al. [40], extensively done the analysis of throughput-area tradeoff for AES implementations. The experimentations revealed that the design with LUT S-Box implementations without pipeline was costlier by around 37% than the design with three pipelined stages of SubstituteByte. Moreover, the 4 pipeline stage of SubstituteByte requires 33% less area than the design with LUT S-Box implementation at same speed. For one pipelined stage per round LUT S-Box based design, if the demanded throughput increases from 30Gbps to 45Gbps, the area also increases by 43%. Moreover of a 4 stage pipelined per round composite SubstituteByte based design, an increase in throughput from 45Gbps to 65Gbps will require just 17% increase in area. The use of higher pipeline stages in composite SubstituteByte based designs, result into higher throughput with less penalties on the area in terms of gate count, whereas in case of LUT based designs the penalties on area in terms of gate count is higher.

### 2.3 Summary of Literature Survey

The summary of literature survey presented in earlier sub-section is put again in a tabular form. This also includes references to few more implementations, which follow more or less the same strategies employed by the ones mentioned in earlier sub-section.

Since the FPGAs device family members, Virtex II and Spartan II onwards, have Block RAMs (BRAMs) readily available on board, most of the design implementations on these devices required less number of slices than the older version devices in their respective families. The presence of BRAMs in the devices, favored the LUT based designs, and hence lots of CLB slices were spared for other control logic implementation. A proper comparison of the design implementations can be done, only if these BRAMs resources are counted using equivalent number of gates. Table 2.1 shows the comparisons in tabular form for some of the key implementation results published till date. Our primary objective was to understand the various possible
architectural transforms employed for AES implementation, and their impact on the performance, hence we did not attempt to convert BRAMs into equivalent gate count.

<table>
<thead>
<tr>
<th>Table 2.1</th>
<th>Summary of Implementations Targeted on FPGAs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>E XCV1000BG560 10992 31.8 1940 Partial Pipelined Architecture with 5 stages and one sub-pipeline stage</td>
</tr>
<tr>
<td>[42]</td>
<td>E / D XCV300BG432 2358 22 259 Key required in next round is calculated in earlier round</td>
</tr>
<tr>
<td>[34]</td>
<td>E XCV1000BG560 11022 168.4 21556 Composite Field Arithmetic GF((2^16)) instead of GF(2^8) for S-Box, Round key generated on the fly</td>
</tr>
<tr>
<td>[42]</td>
<td>E EP1S20F780C5 12827 82 10490 Pipelined Partial &amp; Rolled Architecture with 8 S-Modules of 16 bits each</td>
</tr>
<tr>
<td>[19]</td>
<td>E XCV1000BG560 18210 120 384 GF((2^16)) in key scheduler and Encryption Unit, Rolled and inner pipelining method</td>
</tr>
<tr>
<td>[43]</td>
<td>E XC2V1000-6 3720 194.7 24922 Pipelining, dynamic and partial reconfiguration</td>
</tr>
<tr>
<td>[44]</td>
<td>D XC2V1000-6 280 105 37 8-bit systolic architecture, Cipher Block Chaining (CBC) mode</td>
</tr>
<tr>
<td>[45]</td>
<td>E XC2VP70-7 200 232.6 29800 Fully Pipelined</td>
</tr>
<tr>
<td>[45]</td>
<td>E / D XC2VP70-7 11433 125.3 16080 Fully Pipelined</td>
</tr>
</tbody>
</table>
The summary of ASIC based implementations is listed in Table 2.2. An advertent selection of higher CMOS technology can achieve higher throughput, due to the lower
path delays of the complete design, but the selection of architectural topology can provide even higher throughput over an optimized silicon area.

Very high throughput of 70Gbps can be achieved on ASIC, provided offline key expansion is implemented. The cost of this high throughput is in the form of higher gate count due to the provisioning of memory for storing round keys. We have observed that certain strategies implemented on FPGAs are not tried on ASIC platforms. Systolic architecture and the architectures maximally sharing the hardware resources between encryption and decryption can be employed for low area implementation of AES on CMOS platforms.