AES on FPGA from the fastest to the smallest

Tim Good and Mohammed Benaissa

Department of Electronic & Electrical Engineering,
University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK
{t.good, m.benaissa}@sheffield.ac.uk

Abstract. Two new FPGA designs for the Advanced Encryption Standard (AES) are presented. The first is believed to be the fastest, achieving 25 Gbps throughput using a Xilinx Spartan-III (XC3S2000) device. The second is believed to be the smallest and fits into a Xilinx Spartan-II (XC2S15) device, only requiring two block memories and 124 slices to achieve a throughput of 2.2 Mbps. These designs show the extremes of what is possible and have radically different applications from high performance e-commerce IPsec servers to low power mobile and home applications. The high speed design presented here includes support for continued throughput during key changes for both encryption and decryption which previous pipelined designs have omitted.

Keywords: Advanced Encryption Standard (AES), Field Programmable Gate Array (FPGA), finite field, design exploration, high throughput, pipelined, low area, Application Specific Instruction Processor (ASIP).

1. Introduction

The research objective is to explore the design space associated with the Advanced Encryption Standard (AES) algorithm and in particular its Field Programmable Gate Array (FPGA) hardware implementation in terms of speed and area.

The Rijndael cipher algorithm developed by Vincent Rijmen and Joan Daemen won the competition run by the US government (NIST) in 2000 to select a new commercial cryptographic algorithm and was accorded the accolade the Advanced Encryption Standard (AES). This algorithm is documented in the freely available US government publication, FIPS-197 [1].

Subsequently, the AES has been the topic of much research to find suitable architectures for its hardware implementation. Architectural choices for a given application are driven by the system requirements in terms of speed and the resources consumed. This can simply be viewed as throughput and area, however, latency may also be important as may the cipher's mode of operation. The FIPS-197 specification details a number of modes of operation for the cipher, for example, the simplest is the Electronic Code Book (ECB). Additional resilience to attack can be gained by using one of the feedback modes, for example, Output Feed Back (OFB) mode unfortunately such modes also limit the effectiveness of pipelining.

The use of FPGA has been expanding from its traditional role in prototyping to mainstream production. This change is being driven by commercial pressures to reduce design cost, risk and achieve a faster time to market. Advances in technology have resulted in mask programmed mass produced versions of FPGA fabrics being offered by the leading manufacturers which, for some applications, remove the necessity to move prototype designs from FPGA to ASIC whilst still achieving a low unit cost.

Previous attempts [2,3] at high speed pipelined design have been to use what is an effectively ASIC number-of-gates-in-critical-path design flow to place the pipeline cuts. This is fine where the target device is an ASIC, however, does not result in optimal pipeline cut placement for a given FPGA fabric. This paper presents an alternative flow specific to FPGA which results in optimal pipeline placement thus increased performance. The new high speed design reported here achieves a throughput of 25 Gbps on a Xilinx Spartan-III FPGA and has applications in the area of hardware accelerators for IPsec servers.

An additional novelty of the new high speed design presented in this paper is that the key may be periodically changed without loss of throughput and the operating mode may be changed between encryption and decryption at will. This enables the design to support a mode of operation where a batch of blocks may be encrypted or decrypted for each of a number of differently keyed concurrent channels without loss in throughput.

Reported low area architectures [4,5] have been based around a 32-bit datapath. As the AES operations MixColumns and KeyExpansion are fundamentally 32-bit it was previously believed that this was optimal. An ASIC design by Feldhofer et al [6] used an 8-bit datapath married to a 32-bit MixColumns operator. However, even MixColumns may be rewritten in an 8-bit form accepting a higher control overhead and reduced throughput. To the authors' knowledge, no 8-bit Application Specific Instruction Processor (ASIP) for AES has been reported in the literature. The results from the design of such a processor, which is believed to be the smallest, are documented in this paper. This design only occupies 60% of the smallest available Xilinx Spartan-II device (XC2S15) and achieves a throughput of 2.2 Mbps which is suitable for numerous applications in the mobile and home communications areas. This 8-bit design was compared to the two latest reported low area FPGA designs [4,5] which were based on a 32-bit architecture. Brief details of these designs are included together with their area cost results. A rival 'PicoBlaze' implementation is also presented as a benchmark to demonstrate the performance of a soft core microcontroller based design.

This paper concludes with a discussion on the relative merits of each architecture.

2. The Design

The intention here is to contrast a number of different architectures from the highest speed to the lowest area. An FPGA design flow is used throughout and performance results are presented together with comparison with the previously known best designs. The designs presented all support a 128-bit key. Xilinx ISE version 6.3 was used for the design flow and the results quoted are from post place and route figures including all input and output delays. The new designs were coded in VHDL and validated using ModelSim.

2.1 Fully Parallel Loop Unrolled Architecture

FPGAs are particularly fast in terms of throughput when designs are implemented using deep pipeline cuts [2, 3, 7, 8, 9, 10, 11, 12]. The attainable speed is set by the longest path in the design so it is important to place the pipeline cuts such that the path lengths between the pipeline latches are balanced.

First, the algorithm must be converted into a suitable form for deep pipelining [2,7]. This is achieved by removing all the loops to form a loop-unrolled design where the data is moved through the stationary execution resources. On each clock cycle, a new item of data is input and progresses over numerous cycles through the pipeline resulting in the output of data each cycle, however, with some unavoidable latency.

One of the key optimisations was to express the SubBytes operation in computational form rather than as a look-up table. Earlier implementations used the look-up table approach (the “S” box) but this resulted in an unbreakable delay due to the time required pass through the FPGA block memories. The FIPS-197 specification provided the mathematical derivation for SubBytes in terms of Galois Field (2^8) arithmetic. This was efficiently exploited by hardware implementations using composite field arithmetic [2,7] which permitted a deeper level of pipelining thus improved throughput.

The method of placement of pipeline latches (or cuts) was to consider the synthesis estimates for various units within the design. In particular, for Xilinx FPGAs, the number of cascaded 4-input LUTs in the critical path together with routing delays dominate the minimum cycle period. The first stage of optimisation is to consider the routing delay as constant and only consider change in the number of cascaded LUTs. In further optimisation design cycle iterations, the secondary effects of excessive routing delays and fan out load were considered.

A simple function, such as the reduction-OR of a bit vector, can be used to generate LUT-levels versus cycle time results for the internal fabric of a specific technology. Table 1 shows such results for Xilinx Virtex-E. As can be seen from the table having pipeline registers between each LUT would yield the fastest design, however, there is a compromise in terms of the amount of fan-in required by the logic expressions in the design, the acceptable latency and realistic routing.

Logic Levels	Path Delay, ns	Max Clock Freq, MHz
1	2.176	459.6
2	3.321	301.1
3	4.466	223.9
4	5.611	178.2
5	6.756	148.0
6	7.901	126.6
7	9.046	110.5
8	10.191	98.1

Table 1. Virtex-E performance versus logic levels

The Virtex FPGA slice consists of two LUTs and one D-type flip-flop (FD) so a single level of logic between FDs would under utilise cells resulting in an approximate factor of two increase in the design area thus an impact on speed due to the larger distances. Similarly, two levels of logic between FDs would not provide sufficient flexibility (number of input terms in an expression) for the AES algorithm thus is likely to result in a significant increase of area. Further, with only two LUT levels routing, propagation time, fan-out and congestion from a lack of suitable routing resources are very likely to dominate the cycle time. This leaves three logic levels as the aiming point for pipeline register placement.

There is a further complication in that the slice architecture includes a number of multiplexer (MUX) resources in addition to the LUTs these can be used to implement 2-input XOR and 2-input MUX functions without recourse to an extra level of LUTs. This factor must also be considered when placing the pipeline cuts.

For a given set of pipeline cuts the synthesis results may be examined to verify that the critical path contains the correct number of cascaded LUTs. This design process yielded the following optimal cut set; Figure 1 shows the composite field implementation of SubBytes [2] followed by ShiftRows (SR) and MixColumns (M) operations. The number of LUT-levels is shown adjacent to each design unit and the total in a given pipeline step (represented by the dashed lines) at the bottom of the diagram. From an initial implementation it was found that additional time had to be allowed for the excessively long routing associated with the ShiftRows operation. Thus both the ShiftRows and its inverse require extra time compared to the remainder of the design. The excess time is approximately equivalent to a time associated with using two LUTs. The circuit shown can perform both encryption and decryption operations.

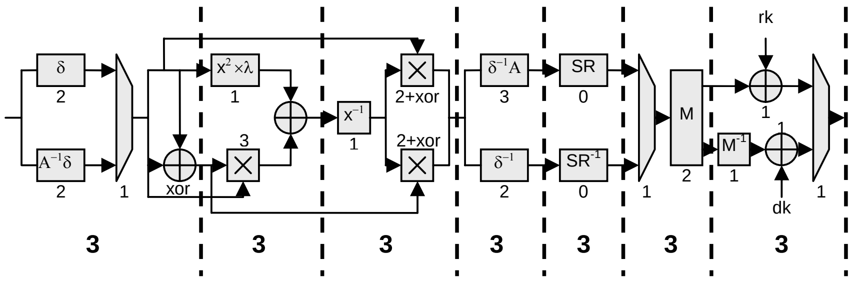

Fig. 1. Block diagram for each middle round

The same treatment was given to the placement of pipeline cuts in the final round (Figure 2) which conveniently turned out to require one less cut than the middle rounds. This was used to accommodate the single cut required for the first round to yield a regular timing pattern.

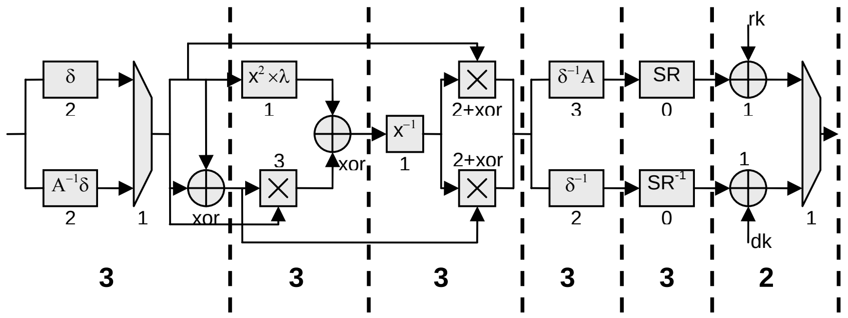

Fig. 2. Block diagram for final round

The key expansion also required implementing and in some previous designs had been overlooked. One key design decision was how frequently the key must be changed and whether continued throughput is required. In this design, it was decided that throughput should be maintained during key changes and that it was desirable to change between encryption and decryption on each cycle with key changes made on similar order to the latency.

The pipeline cuts chosen meant that a given data item will pass completely through the AES cipher in 70 cycles. One issue with the AES KeyExpansion is that the decryption process starts with the final RoundKey and the only method of obtaining the final RoundKey is to progress through the entire key expansion. This issue was resolved by having separate encryption and decryption RoundKey registers and the new key being supplied suitably in advance to its data (140 cycles). Although the additional registers occupy a sizable amount of area it does permit maintaining throughput during key changes. The KeyExpander takes 10 cycles to fully generate the required set of RoundKeys. In order to match the latency through the main datapath, the KeyExpander was placed in a separate clock domain running at 1/7 of the datapath clock. This allowed for many more levels of logic in the KeyExpander without it forming part of the critical path.

The architecture for the KeyExpander is shown in Figure 3. The InvMixColumns unit (M^{-1}) is included to maintain the order of the operations the same for both encryption and decryption. This is referred to in the FIPS-197 specification as “Equivalent Decryption”. There is no pipelining in the KeyExpander and it evaluates one RoundKey every clock cycle (in its clock domain). The “RCON” values, defined in the FIPS-197 specification, are computed using repeated finite field doubling (FFM2 unit). Four non-pipelined, forward transform only, versions of the SubBytes operation were implemented using composite field arithmetic (S units). The output RoundKeys are registered to permit correct operation given key changes and selection between encryption and decryption (rk1 to rk10 for encryption and dk1 to dk9 for decryption). The first RoundKey (rk0) is obtained by directly registering the key input.

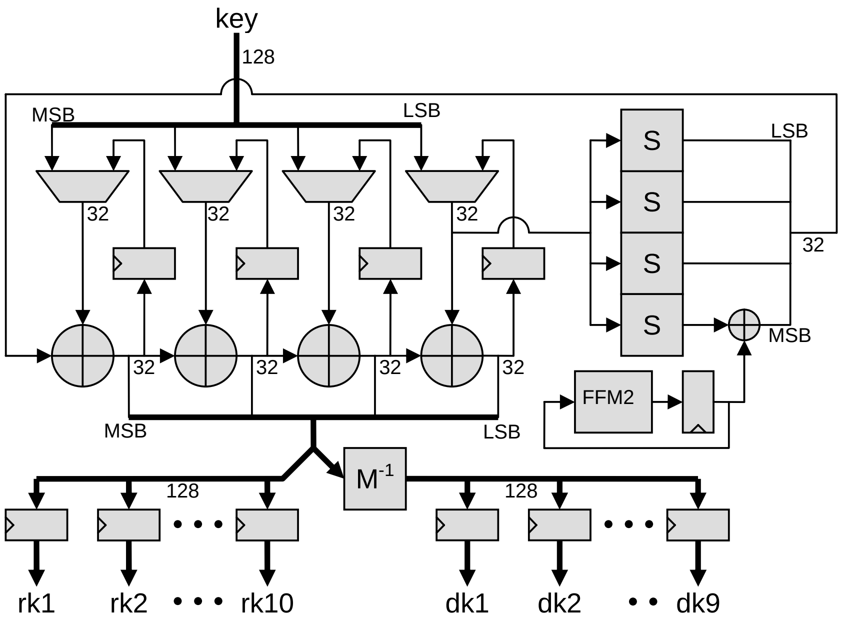

Fig. 3. Block Diagram of KeyExpander.

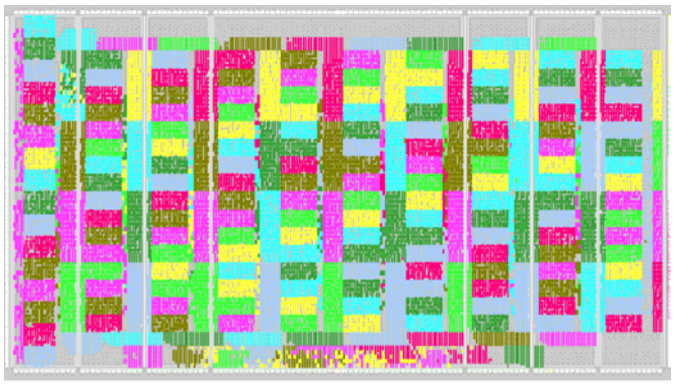

Fig. 4. Placement of design on Virtex-E (XCV2000E).

Design	FPGA Part	Freq. MHz	Thro'put Mbps	Latency ns	Area slices	Mbps / slice	Data path
Jarvinen et al [9]	Virtex-E XCV1000e-8	129.2	16,500		11,719	1.408	Enc
Saggesse et al [10]	Virtex-E XCV2000e-8	158	20,300		5810 + $100B_{RAM}$	1.091	Enc
Standaert et al [11]	Virtex-E XCV3200e-8	145	18,560		15,112	1.228	Enc
Hodjat et al [3] Excl. key expand	Virtex-II Pro XC2VP20	169.1	21,640	420	9,446 Excl. KE	2.290	Enc
Zambreno et al [8]	Virtex-II XC2V4000	184.1	23,570	163	16,938	1.391	Enc
Zhang (r=7), [2] Excl. key expand	Virtex-E XCV1000E-8	168.4	21,556	416	11,022 Excl. KE	1.956	Enc/ Dec
This work, 3LUT cut, key change support	Virtex-E XCV2000E-8	184.8	23,654	379	16,693	1.417	Enc/ Dec
This work, 3LUT cut, key change support	Spartan-III XC3S2000-5	196.1	25,107	357	17,425	1.441	Enc/ Dec

Table 2. Performance comparison of this work with previous designs.

The placement of the design on a Virtex-E is shown in Figure 4 and the comparative results in Table 2. When comparing the quoted performance figures it is important to recognise the differences caused by changes in FPGA technology or more importantly with the level of support for key agility, encryption and decryption. Some designs did not include the key expansion in the results and other only supported the encryption datapath. This design shows an improvement in throughput over the previously known best design [2] of approximately 10% using the same FPGA technology. However, further savings can be made by moving from the Virtex-E to the lower cost Spartan-III devices with an increase in performance due to the more modern technology. The design achieves 25 Gbps throughput on the Spartan-III XC3S2000-5 device.

Further improvement in throughput, of say 20%-30%, is possible by adopting a 2-LUT cut, however factors such as fanout and congestion are likely to be a significant obstacle to obtaining an improved throughput-area figure.

Traditionally, such pipelined designs [3, 8, 9, 10, 11] only demonstrated any key agility in encryption only modes such as Counter mode (CTR). However, this design supports key agility for both encryption and decryption thus can support Electronic Code Book (ECB) mode. In a multi channel environment the key can be changed once ever 70 cycles thus support batch processing for a number of differently keyed concurrent channels without loss in throughput.

Further, it is a relatively simple task to extend the design by pipelining the key expansion, repeating its instantiation for all ten rounds and include registers (equivalent to approximately 15232 flip-flops) to support key changes each cycle. This would, in a multi channel environment, support any of the feedback modes, including Cipher Block Chaining (CBC) thus gaining improved security.

2.2 Round based architecture using 32-bit Datapath

There already exist a number of good 32-bit based designs [4, 5, 13]. In these designs the AES is implemented by breaking up a round into a number of smaller 32-bit wide operations. Thus a number of cycles is required to complete each round.

Such designs are based around a store for the “state” data (16 bytes for 128-bit) and look-up tables to perform the required AES operations of SubBytes and MixColumns. One of the optimisations documented in the FIPS-197 specification is to combine the look-up table for the MixColumns and SubBytes operation into a single one. This is often referred to as the “T-box”.

One of the key optimisations used by Rouvroy [5] was to exploit the larger (18kbit) block memories afforded by the Spartan-III and Virtex-II series FPGAs. This allowed for 4 off 32 bit x 256 word look up tables (ROMs) to be implemented per (dual port) block memory. So the required number of 8 off 32 bit x 256 word lookup tables can be implemented in two block memories, providing the four address buses needed (8-bits data in + 3-bits mode). The contents of the look up tables were chosen to provide convenient access to SubBytes and InvSubBytes required by the key expander and the finite field multiplications of the SubBytes table required for the combined SubBytes – MixColumns operation. The operation is completed by computing the exclusive-or of the four partial “T-box” values. The values stored are given by the following expression for the 8-bit value, a, using the SubBytes transformation and finite field multiplication by the given constant:

$$T(a) = \left[\begin{array}{c|c} 2 \bullet SB(a) & 14 \bullet ISB(a) \\ SB(a) & 9 \bullet ISB(a) \\ ISB(a) & 13 \bullet ISB(a) \\ 3 \bullet SB(a) & 11 \bullet ISB(a) \end{array}\right] \tag{1}$$

A further block memory was used to store the RoundKeys together with several multiplexers to route the data and key.

The following results (Table 3) were quoted together with a comparison with the previous design by Chadoweic and Gaj [4]. However, the figures quoted for throughput versus area failed to take into account the size of the block memories. This is of particular importance as the block memories on the Spartan-II are 4 kbit whereas those found on the Spartan-III and Virtex-II are 18 kbit. If these costs are taken into account then the result is substantially changed.

The cost of using a block memory in terms of an equivalent number of slices is still a matter of some debate. One option would be to make the comparison based on the physical area occupied by a slice and a block memory but quotable figures are not forthcoming from the manufacturers. An alternative is to consider the number of slices required to implement the equivalent distributed memory, however, this varies depending on the functionality required (for example single or dual port). Such estimates vary between 8 and 32 bits/slice. For this analysis a worst case figure of 32 bits/slice was used. The relative merits of the various designs and thus conclusions remain unchanged when the analysis was repeated for the lower estimate.

	Chodowiec & Gaj [4]	Rouvroy et al [5]	Pramstaller et al [13]	Rouvroy et al [5]
Device	XC2S30-6	XC3S50-4	XCV1000E	XC2V40-6
Slices	222	163	1125	146
Throughput (Mbps)	166	208	215	358
RAM blocks	3	3	0	3
Throughput / Area(kbps / slice) ignoring block ram	750	1260	191	2450
Bits of block ram used	9600	34176	0	34176
Equiv slices for block ram	300	1068	0	1068
Total equiv. slices (area)	522	1231	1125	1214
Throughput / Area (kbps / slice)	318	169	191	295

Table 3. Performance of existing 32-bit FPGA designs

2.3 Application Specific Processor Architecture using 8-bit Datapath

The objective was to develop a small AES implementation. One option was to use the freely available Xilinx PicoBlaze soft core processor [14] for which the first version only requires 76 slices. However, for a practical design a small memory was needed thus the larger 96 slice KCPSM3 was selected. Additionally, the size of the ROM required to implement 365 instructions for the AES had to be considered together with an implementation for SubBytes. This results in a final design using the PicoBlaze of 119 slices plus the block memories which are accounted for here by an equivalent number of slices (once again 32 bits per slice was used). The resulting design had an equivalent slice count of 452 and with a 90.2 MHz maximum clock. Key expansion followed by encipher took 13546 cycles and key expansion followed by decipher 18885 cycles. The average encipher-decipher throughput was 0.71 Mbps.

An application specific instruction processor (ASIP) was developed based around an 8-bit datapath and minimal program ROM size. Minimisation of the ROM size resulted in a requirement to support subroutines and looping. This added area to the control portion of the design but the saving was made in terms of the size of the ROM. The total design, including an equivalent number of slices for the block memories only occupies 259 slices to give a throughput of 2.2Mbps.

The datapath consisted of two processing units, the first to perform the SubBytes operation using resource shared composite field arithmetic and the second to perform multiply accumulate operations in Galois Field 2^8. A minimal set of instructions was developed (15 in total) to perform the operations required for the AES. The processor (Figure 5) used a pipelined design permitting execution of a new instruction every cycle.

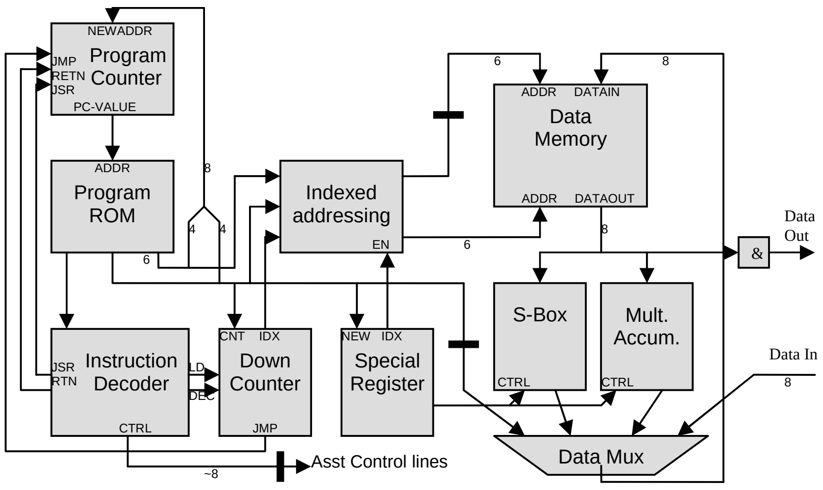

Fig. 5. ASIP Architecture

Figure 6 is a pie chart depicting the balance of area between the various design units. Of the processor hardware approx 60% of the area is required for the datapath and 40% for the control.

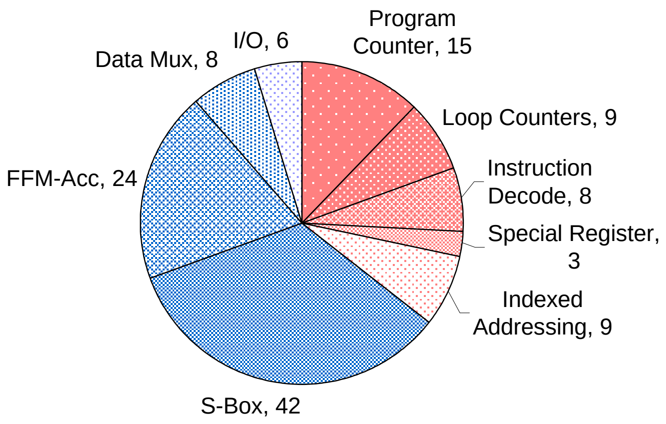

Fig. 6. Slice utilization versus design unit

As a good FPGA based 8-bit datapath for comparison could not be found, Table 4 shows comparison of this design with the state-of-the-art 32-bit designs using the relatively low cost Xilinx Spartan FPGAs. The two reference designs both quote throughput figures for a mode of operation where the key remains constant thus the time taken for key expansion was not included. A throughput figure was calculated for each design inclusive of the time taken for key expansion. The average for encipher and decipher was then calculated and is reported in Table 4 as the average throughput.

	This design	Picoblaze based	Chodowiec & Gaj [4]	Rouvroy et al [5]
FPGA	Spartan-II XC2S15-6	Spartan-II XC2S15-6	Spartan-II XC2S30-6	Spartan-III XC3S50-4
Clock Frequency (MHz)	67	90	60	71
Datapath Bits	8	8	32	32
Slices	124	119	222	163
No. of Block RAMs used	2	2	3	3
Block RAM Size (kbits)	4	4	4	18
Bits of block RAM used	4480	10666	9600	34176
Est. equiv. slices for memory	140	333	300	1068
Total Equiv. Slices (area)	264	452	522	1231
Max Throughput (Mbps)	-	-	166	208
Ave. Throughput (Mbps)	2.2	0.71	69	87
Throughput/slice (kbps/slice)	8.3	1.9	132	70
Summary	Smallest	Software	Best speed/area	Fastest

Table 4. Comparison with other designs using low cost FPGAs

Figure 7 shows the placement of this design on a Spartan-II (XC2S15) part. The design requires 124 slices and two block memories. One memory formed the program ROM and the second was used as the ASIP's main memory (RAM). The AES application only required 360 bits of RAM thus the block memory was only partially utilized and could have be implemented as distributed memory with a cost of 42 additional slices and would then free up one of the block memories. There are also some concerns over the particular vulnerability of the block memories to power attacks so avoiding their use for key and data storage may be desirable. However, even avoiding use of the block memories does not negate such risks.

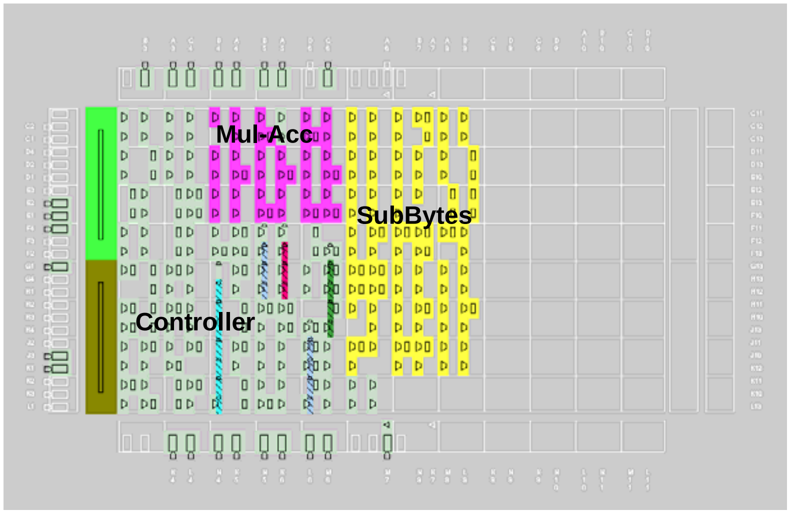

Fig. 7. Placement of low area design on Spartan-II (XC2S15)

3 Conclusions

This paper has presented a number of FPGA implementations from the fastest to the smallest. In terms of speed to area ratio the unrolled designs perform the best as there is no controller overhead. However, such designs are very large and need a 1 – 2 million gate device but achieve throughputs up to 25 Gbps. These designs have applications in fixed infrastructure such as IPsec for e-commerce servers.

The low area design described here achieves 2.2 Mbps which is sufficient for most wireless and home applications. The area required is small thus fundamentally low power so has utility in the future mobile area. The design requires just over half the available resources of the smallest available Spartan-II FPGA.

The advantage of the 8-bit ASIP over the more traditional 8-bit microcontroller architecture (PicoBlaze) is shown by the approximate factor of three improvement in throughput and 40% reduction in area (including estimated area for memories).

The 32-bit datapath designs occupy the middle ground between the two extremes and have utility where moderate throughput in the 100 – 200 Mbps is required.

The advantage of an FPGA specific optimisation over an ASIC number-of-gates approach has been demonstrated by the speed improvement made in the loop unrolled design.

The best architectural decision is to select the design of the lowest possible area meeting the throughput and operating mode requirement for the system being developed. Figure 8 shows the different designs in terms of their throughput and area.

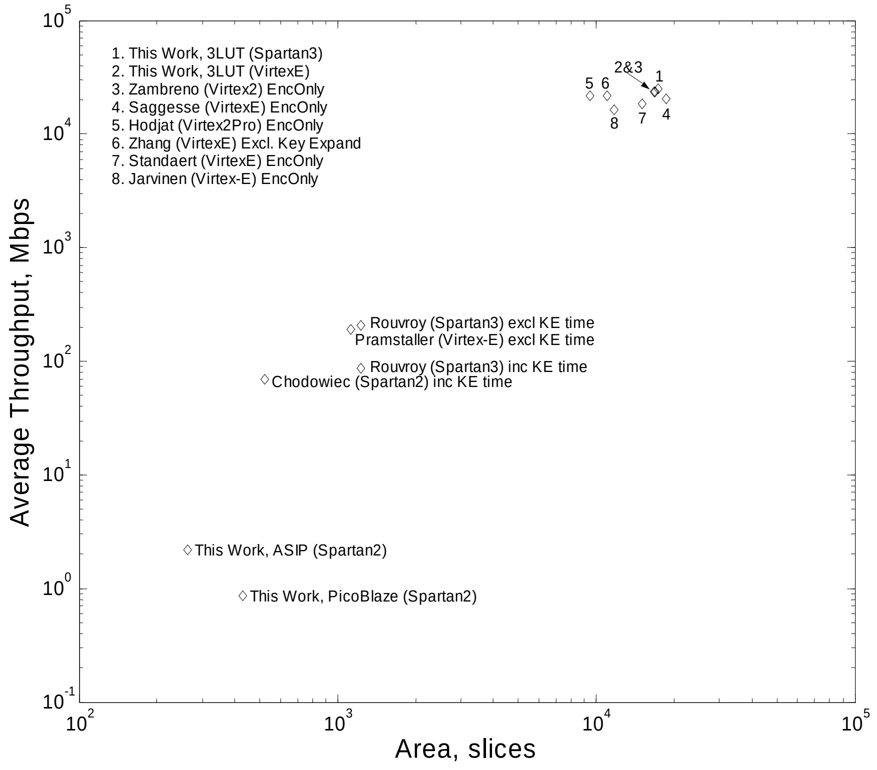

Fig. 8. Throughput versus area for the different FPGA designs

4 References

[1] National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), *Advanced Encryption Standard (AES)*, Federal Information Processing Standards (FIPS) Publication 197, November 2001

[2] X. Zhang, K. K. Parhi, *High-speed VLSI architectures for the AES algorithm*, IEEE Trans. VLSI Systems, Vol. 12, Iss. 9, pp. 957 - 967, Sept. 2004

[3] A. Hodjat, I. Verbauwhede, *A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA*, 12th Annual IEEE Sypmosium on Field-Programmable Custom Computing Machines (FCCM'04), pp. 308-309, April 2004

[4] P. Chodowiec, K. Gaj, *Very Compact FPGA Implementation of the AES Algorithm*, Cryptographic Hardware and Embedded Systems (CHES 2003), LNCS Vol. 2779, pp. 319 – 333, Springer-Verlag, October 2003

[5] G. Rouvroy, F. X. Standaert, J. J. Quisquater, J. D. Legat, *Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications*, Procedings of the international conference on Information Technology: Coding and Computing 2004 (ITCC 2004), pp. 583 – 587, Vol. 2, April 2004

[6] M. Feldhofer, S. Dominikus and J. Wolkerstorfer, *Strong Authentication for RFID Systems Using the AES Algorithm*, CHES 2004, LNCS 3156, pp. 357-370, Springer-Verlag, 2004.

[7] A. Satoh, S. Morioka, K. Takano, S. Munetoh, *A Compact Rijndael Hardware Architecture with S-Box Optimization*, Proceedings of ASIACRYPT 2001, LNCS Vol. 2248, pp. 239 - 254, Springer-Verlag, December 2001

[8] J. Zambreno, D. Nguyen, A. Choudhary, *Exploring Area/Delay Trade-offs in an AES FPGA Implementation*, Proc. FPL 2004, 2004

[9] K. U. Jarvinen, M. T. Tommiska, and J. O. Skytta, *A fully pipelined memoryless 17.8 Gbps AES-128 encryptor*, Proc. Int. Symp. Field-Programmable Gate Arrays (FPGA 2003), Monterey, CA, pp. 207–215, Feb. 2003

[10] G. P. Saggese, A. Mazzeo, N. Mazocca, and A. G. M. Strollo, *An FPGA based performance analysis of the unrolling, tiling and pipelining of the AES algorithm*, Proc. FPL 2003, Portugal, Sept. 2003.

[11] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, *Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements & design tradeoffs*, Proc. CHES 2003, Cologne, Germany, Sept. 2003.

[12] M. McLoone and J.V. McCanny, *High Performance Single-Chip FPGA Rijndael Algorithm Implementations*, CHES 2001, Paris, France, 2001

[13] N. Pramstaller and J. Wolkerstorfer, *A Universal and efficient AES co-processor for Field Programmable Logic Arrays*, FPL 2004, LNCS Vol. 3203, pp. 565-574, Springer-Verlag, 2004.

[14] K. Chapman, *PicoBlaze 8-bit Microcontroller*, Xilinx, 2002 http://www.xilinx.com/products/design_resources/proc_central/grouping/picoblaze.htm

5 Acknowledgement

This work was funded by the UK Engineering and Physical Sciences Research Council (EPSRC).