FPGA Implementations of the Round Two SHA 3 Candidates

Brian Baldwin, Neil Hanley, Mark Hamilton, Liang Lu,
Andrew Byrne, Maire O'Neill and William P. Marnane

Claude Shannon Institute for Discrete Mathematics,
Coding and Cryptography
Department of Electrical & Electronic Engineering,
University College Cork, Ireland
Institute of Electronics, Communications & Information Technology,
Queens University Belfast, Belfast, UK
School of Mathematical & Geospatial Sciences,
RMIT University, Melbourne, Australia

Overview

Overview

- Round Two Analysis

Overview

- Round Two Analysis
- The Hash Functions

Overview

- Round Two Analysis
- The Hash Functions
- Hardware Wrapper
  - Interfacing
  - Padding

Overview

- Round Two Analysis
- The Hash Functions
- Hardware Wrapper
  - Interfacing
  - Padding
- Results
  - Wrapper Results
  - Padding Results
  - Interface Results

Overview

- Round Two Analysis
- The Hash Functions
- Hardware Wrapper
  - Interfacing
  - Padding
- Results
  - Wrapper Results
  - Padding Results
  - Interface Results
- Conclusions

Round Two Analysis

Round Two Analysis

- 6.C- Round Two Technical Evaluation
- Message digest sizes:

Round Two Analysis

- 6.C- Round Two Technical Evaluation
- Message digest sizes:
  - *Round Two testing by NIST will be performed on the required message digest sizes*

Round Two Analysis

- 6.C- Round Two Technical Evaluation
- Message digest sizes:
  - *Round Two testing by NIST will be performed on the required message digest sizes*
- Efficiency testing:
  - *The calculation of the time required to compute message digests for various length messages*

Round Two Candidates

Hash Functions

Round Two Candidates

Hash Functions

- BLAKE, Blue Midnight Wish,
- Cubehash, ECHO,
- Fugue, Grøstl,
- Hamsi, JH,
- Keccak, Luffa,
- Shabal, SHAvite-3,
- SIMD, Skein

Description

			224/256				384/512			
Design	**Structure**	**Type**	**Counter**	**Message**	**Salt**	**State**	**Counter**	**Message**	**Salt**	**State**
SHA-2	Merkle-Damgård	Add-XOR-Rotate	64	512	0	512	128	1024	0	1024
Blake	HAIFA	Add-XOR-Rotate	64	512	128	512	128	1024	256	1024
BMW	Iterative	Add-XOR-Rotate	64	512	0	2048	64	1024	0	4096
Cubehash	Iterative	Add-XOR-Rotate	0	256	0	1024	0	256	0	1024
Echo	HAIFA	AES based	64	1536	128	2048	64	1536	128	2048
Fugue	Iterative	AES based	64	32	0	960	64	32	0	1148
Grøstl	Iterative	AES based	64	512	0	512	64	1024	0	1024
Hamsi	Conc-Permute	Serpent based	64	32	0	512	64	64	0	1024
JH	Iterative	Block Cipher based	128	512	0	1024	128	512	0	1024
Keccak	Sponge	Add-XOR-Rotate	0	1088	0	1600	0	576	0	1600
Luffa	Sponge	S-box based	0	256	0	768	0	256	0	1280
Shabal	Iterative	Add-XOR-Rotate	0	512	0	1408	0	512	0	1408
SHAvite-3	HAIFA	AES based	64	512	256	256	128	1024	512	512
SIMD	Iterative	Block Cipher based	64	512	0	512	64	1024	0	1024
Skein	UBI	Add-XOR-Rotate	96	512	0	512	96	512	0	512

Description

			224/256				384/512			
Design	**Structure**	**Type**	**Counter**	**Message**	**Salt**	**State**	**Counter**	**Message**	**Salt**	**State**
SHA-2	Merkle-Damgård	Add-XOR-Rotate	64	512	0	512	128	1024	0	1024
Blake	HAIFA	Add-XOR-Rotate	64	512	128	512	128	1024	256	1024
BMW	Iterative	Add-XOR-Rotate	64	512	0	2048	64	1024	0	4096
Cubehash	Iterative	Add-XOR-Rotate	0	256	0	1024	0	256	0	1024
Echo	HAIFA	AES based	64	1536	128	2048	64	1536	128	2048
Fugue	Iterative	AES based	64	32	0	960	64	32	0	1148
Grøstl	Iterative	AES based	64	512	0	512	64	1024	0	1024
Hamsi	Conc-Permute	Serpent based	64	32	0	512	64	64	0	1024
JH	Iterative	Block Cipher based	128	512	0	1024	128	512	0	1024
Keccak	Sponge	Add-XOR-Rotate	0	1088	0	1600	0	576	0	1600
Luffa	Sponge	S-box based	0	256	0	768	0	256	0	1280
Shabal	Iterative	Add-XOR-Rotate	0	512	0	1408	0	512	0	1408
SHAvite-3	HAIFA	AES based	64	512	256	256	128	1024	512	512
SIMD	Iterative	Block Cipher based	64	512	0	512	64	1024	0	1024
Skein	UBI	Add-XOR-Rotate	96	512	0	512	96	512	0	512

- Keccak- message size increases to 1152 for $\{224\}$ and 832 for $\{384\}$,
- Luffa- the state size decreases to 1024 for $\{384\}$

Hash Variants

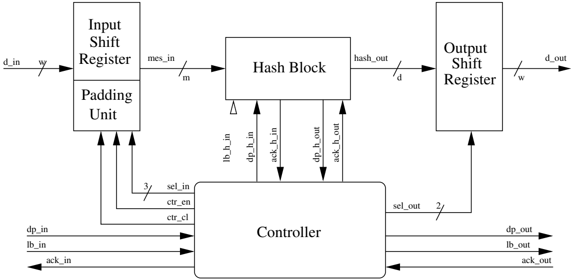
Single Design	Two Variants	Three Variants	Four Variants
Cubehash	SHA-2	Fugue	Keccak
JH	BLAKE	Luffa	-
Shabal	BMW	-	-
Skein(512)	ECHO	-	-
-	Grøstl	-	-
-	Hamsi	-	-
-	SHAvite-3	-	-
-	Simd	-	-

Hash Variants

Single Design	Two Variants	Three Variants	Four Variants
Cubehash	SHA-2	Fugue	Keccak
JH	BLAKE	Luffa	-
Shabal	BMW	-	-
Skein(512)	ECHO	-	-
-	Grøstl	-	-
-	Hamsi	-	-
-	SHAvite-3	-	-
-	Simd	-	-

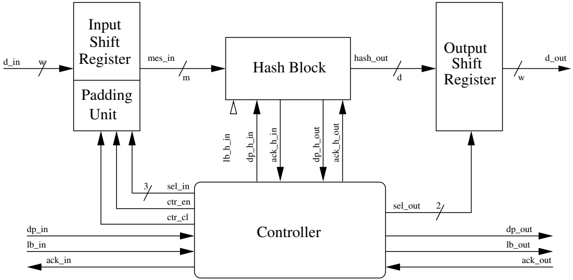
- Necessary to implement 28 different designs to cover all hash variants for the Round Two competitors
  - Also implemented 2 different designs for SHA-2 benchmark

Hardware Wrapper



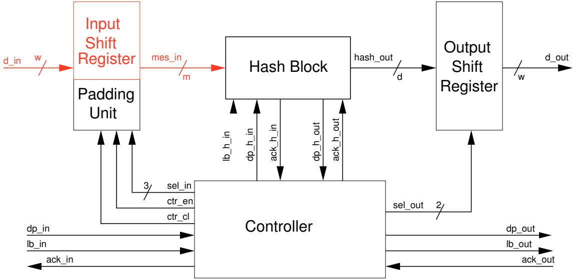
- We developed a hardware wrapper interface

Hardware Wrapper



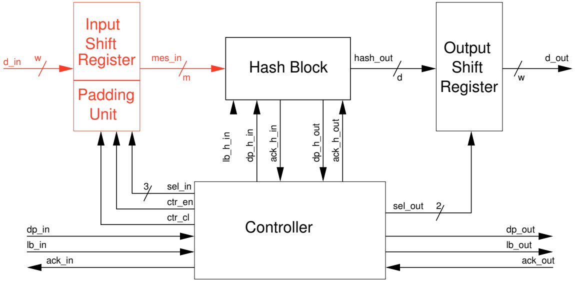
- We developed a hardware wrapper interface
  - An Input Register which includes any padding required
  - An Output Shift-Register
  - Control circuitry

Communications



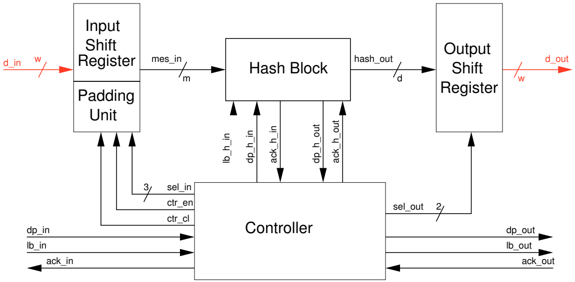
- Input shift-register reads in and stores w-bit values to the size required, m, which is the message block size of the hash function under test

Communications



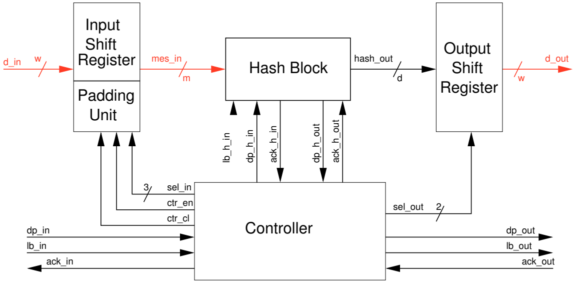
- If a message ends prior to this register being completely filled, padding is added to the partial message to bring it to the required size

Communications



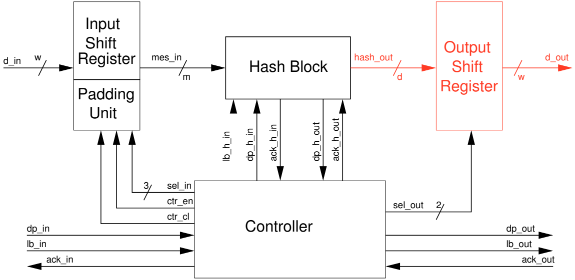
- We set the Input and Output bus, w, to 32-bits, a standard word size

Communications



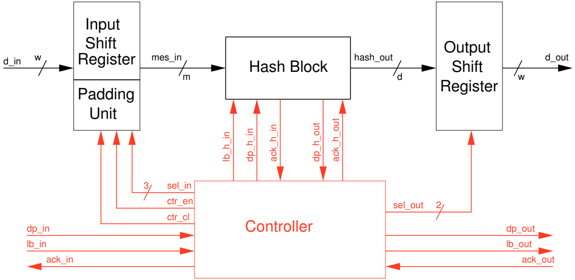
- Any hash function requiring a large message size, m, will be subject to a latency of m/w clock cycles

Communications



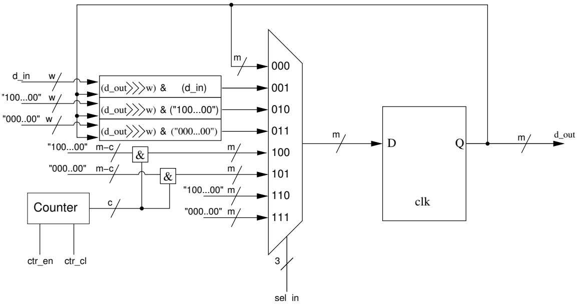
- The output shift-register performs a similar task, holding the hashed message digest of size d, while the output bus reads it out w bits at a time

Communications



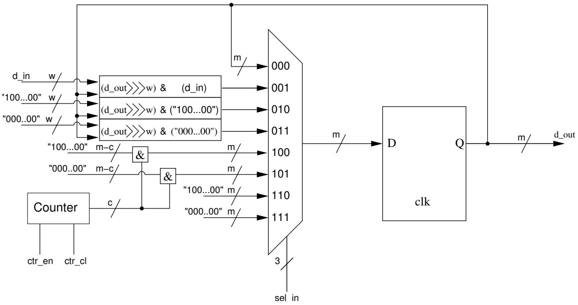
- Control circuitry controls the Shift Register operation, padding, and all communication signals

Padding



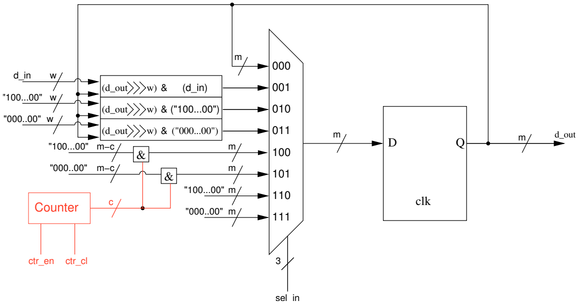
- Produces the padding scheme required for a particular hash function

Padding



- Produces the padding scheme required for a particular hash function
- It allows re-use of any padding scheme that can be used in multiple hash functions

Padding



- While the wrapper itself does not affect the clock frequency, counters in the padding block may form the critical path and thus affect the timing

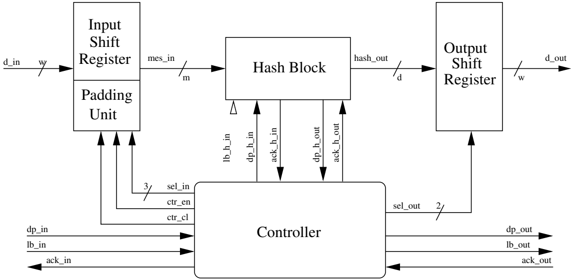
Padding Schemes

Design	Padding Scheme
SHA224/256	1, 0's until congruent (448 mod 512), 64-bit message length
SHA384/512	1, 0's until congruent (896 mod 1024), 128-bit message length
Blake224	1, 0's, until congruent (448 mod 512), 64-bit message length
Blake256	1, 0's, until congruent (447 mod 512), 1, 64-bit message length
Blake384	1, 0's, until congruent (895 mod 1024), 128-bit message length
Blake512	1, 0's, until congruent (894 mod 1024), 1, 128-bit message length
BMW224/256	1, 0's until congruent (448 mod 512), 64-bit message length
BMW384/512	1, 0's until congruent (960 mod 1024), 64-bit message length
Cubehash	1, 0's until a multiple of 256 (256 = 8 * b, b=32)
Echo224/256	1, 0's until congruent (1392 mod 1536), 16-bit message digest, 128-bit message length
Echo384/512	1, 0's until congruent (880 mod 1024), 16-bit message digest, 128-bit message length
Fugue	0's until a multiple of 32, 64-bit message length
Grøstl224/256	1, 0's until congruent (448 mod 512), 64-bit block counter
Grøstl384/512	1, 0's until congruent (960 mod 1024), 64-bit block counter
Hamsi224/256	1, 0's until a multiple of 32, 64-bit message length
Hamsi384/512	1, 0's until a multiple of 64, 64-bit message length

Padding Schemes

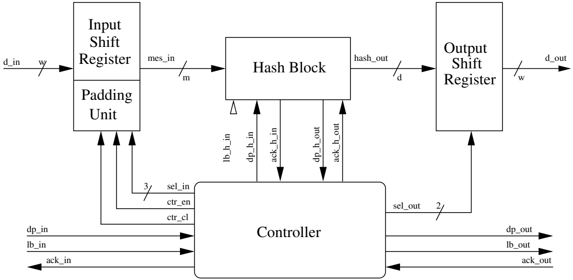
Design	Padding Scheme
JH	1, 0's until congruent (384 mod 512), 128-bit message length, min 512-bits added
Keccak224	1, 0's until a multiple of 8, append 8-bit representation of 28, append 8-bit representation of 1152/8, 1, 0's until a multiple of 1152
Keccak256	1, 0's until a multiple of 8, append 8-bit representation of 32, append 8-bit representation of 1088/8, 1, 0's until a multiple of 1088
Keccak384	1, 0's until a multiple of 8, append 8-bit representation of 48, append 8-bit representation of 832/8, 1, 0's until a multiple of 832
Keccak512	1, 0's until a multiple of 8, append 8-bit representation of 64, append 8-bit representation of 576/8, 1, 0's until a multiple of 576
Luffa	1, 0's until a multiple of 256
Shabal	1, 0's until a multiple of 512
SHAvite3-224/256	1, 0's until congruent (432 mod 512), 64-bit message length, 16-bit digest length
SHAvite3-384/512	1, 0's until congruent (880 mod 1024), 128-bit message length, 16-bit digest length
Simd224/256	0's until a multiple of 512, extra block with message length
Simd384/512	0's until a multiple of 1024, extra block with message length
Skein	0's if multiple of 8, else 1, 0s, until a multiple of 512

Implementation



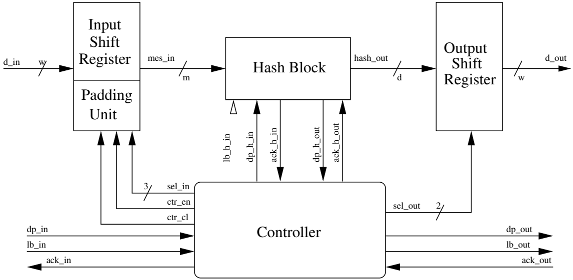
- All designs implemented using slice logic

Implementation



- All designs implemented using slice logic
  - S-boxes and multipliers implemented using distributed memory

Implementation



- All designs implemented using slice logic
  - S-boxes and multipliers implemented using distributed memory
  - Slices used for registers and data storage

Defining the Paramaters

Defining the Paramaters

- Some hash designs require extra time to load in the padding scheme

Extra Padding Stages

- Fugue, Hamsi, JH, SIMD

Defining the Paramaters

- Some hash designs require extra time to load in the padding scheme

Extra Padding Stages

- Fugue, Hamsi, JH, SIMD

- Others have finalisation stages comprising of one or more rounds

Finalisation Stages

- BMW, Cubehash, Echo, Fugue, Hamsi, Luffa, Shabal, SHAvite-3, SIMD, Skein

Clock Count

Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
SHA224/256	16	0	0	64	1	65	0	0	65
SHA384/512	32	0	0	80	1	81	0	0	81
Blake224/256	16	0	0	10	4	40	0	0	40
Blake384/512	32	0	0	14	4	56	0	0	56
BMW224/256	16	0	0	1	4	4	1	3	7
BMW384/512	32	0	0	1	4	4	1	3	7
Cubehash	8	0	0	16	17	17	160	161	178
Echo224/256	48	0	0	8	1	8	1	1	9
Echo384/512	32	0	0	10	1	10	1	1	11
Fugue224/256	1	2	1	1	7	7	13	91	98
Fugue384	1	2	1	1	10	10	20	180	190
Fugue512	1	2	1	1	13	13	22	264	277
Grøstl224/256	16	0	0	10	1	10	0	0	10
Grøstl384/512	32	0	0	14	1	14	0	0	14
Hamsi224/256	1	3	1	3	2	6	6	24	31
Hamsi384/512	2	3	1	6	2	12	12	48	61

- We define a short message as the time required to process the padding, a message block and finalisation

Clock Count

Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
JH	16	1	1	35	1	38	0	0	38
Keccak224	36	0	0	24	1	25	0	0	25
Keccak256	34	0	0	24	1	25	0	0	25
Keccak384	26	0	0	24	1	25	0	0	25
Keccak512	18	0	0	24	1	25	0	0	25
Luffa224/256	8	0	0	8	1	8	1	8	16
Luffa384	8	0	0	8	1	8	2	16	24
Luffa512	8	0	0	8	1	8	2	16	24
Shabal	16	0	0	1	50	50	3	150	200
SHAvite3-224/256	16	0	0	12	3	36	1	1	37
SHAvite3-384/512	32	0	0	14	4	56 (70)	1	1	71
Simd224/256	16	1	1	4	8	32(41)	0.5	4	36(45)
Simd384/512	32	1	1	4	8	32(41)	0.5	4	36(45)
Skein	16	0	0	18	22	22	18	22	44

- We define a short message as the time required to process the padding, a message block and finalisation

Clock Count

Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
SHA224/256	16	0	0	64	1	65	0	0	65
SHA384/512	32	0	0	80	1	81	0	0	81
Blake224/256	16	0	0	10	4	40	0	0	40
Blake384/512	32	0	0	14	4	56	0	0	56
BMW224/256	16	0	0	1	4	4	1	3	7
BMW384/512	32	0	0	1	4	4	1	3	7
Cubehash	8	0	0	16	17	17	160	161	178
Echo224/256	48	0	0	8	1	8	1	1	9
Echo384/512	32	0	0	10	1	10	1	1	11
Fugue224/256	1	2	1	1	7	7	13	91	98
Fugue384	1	2	1	1	10	10	20	180	190
Fugue512	1	2	1	1	13	13	22	264	277
Grøstl224/256	16	0	0	10	1	10	0	0	10
Grøstl384/512	32	0	0	14	1	14	0	0	14
Hamsi224/256	1	3	1	3	2	6	6	24	31
Hamsi384/512	2	3	1	6	2	12	12	48	61

- We define a long message as just the time to process the message block

Clock Count

Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
JH	16	1	1	**35**	**1**	**38**	0	0	38
Keccak224	36	0	0	**24**	**1**	**25**	0	0	25
Keccak256	34	0	0	**24**	**1**	**25**	0	0	25
Keccak384	26	0	0	**24**	**1**	**25**	0	0	25
Keccak512	18	0	0	**24**	**1**	**25**	0	0	25
Luffa224/256	8	0	0	**8**	**1**	**8**	1	8	16
Luffa384	8	0	0	**8**	**1**	**8**	2	16	24
Luffa512	8	0	0	**8**	**1**	**8**	2	16	24
Shabal	16	0	0	**1**	**50**	**50**	3	150	200
SHAvite3-224/256	16	0	0	**12**	**3**	**36**	1	1	37
SHAvite3-384/512	32	0	0	**14**	**4**	**56 (70)**	1	1	71
Simd224/256	16	1	1	**4**	**8**	**32(41)**	0.5	4	36(45)
Simd384/512	32	1	1	**4**	**8**	**32(41)**	0.5	4	36(45)
Skein	16	0	0	**18**	**22**	**22**	18	22	44

- We define a long message as just the time to process the message block

Clock Count Comparison

Hash Design	Long Msg #Cycles	Short Msg #Cycles	Hash Design	Long Msg #Cycles	Short Msg #Cycles
SHA 224/256	65	65	BMW 224/256/384/512	4	7
SHA 384/512	81	81	Hamsi 224/256	6	31
Blake 224/256	40	40	Cubehash	17	178
Blake 384/512	56	56	Hamsi 384/512	12	61
Echo 224/256	8	9	Fugue 224/256	7	98
Echo 384/512	10	11	Luffa 224/256	8	16
Grøstl 224/256	10	10	Fugue 384	10	190
Grøstl 384/512	14	14	Luffa 384/512	8	24
JH	38	38	Fugue 512	13	277
Keccak 224/256/384/512	24	25	Simd 224/256/384/512	32(41)	36(45)
SHAvite3 224/256	36	37	Shabal	50	200
SHAvite3 384/512	56(70)	71	Skein	22	44

- As the size of the message to be hashed increases, these padding and finalisation stages will have less of an impact on the overall calculation time

Clock Count Comparison

Hash Design	Long Msg #Cycles	Short Msg #Cycles	Hash Design	Long Msg #Cycles	Short Msg #Cycles
SHA 224/256	65	65	BMW 224/256/384/512	4	7
SHA 384/512	81	81	Hamsi 224/256	6	31
Blake 224/256	40	40	Cubehash	17	178
Blake 384/512	56	56	Hamsi 384/512	12	61
Echo 224/256	8	9	Fugue 224/256	7	98
Echo 384/512	10	11	Luffa 224/256	8	16
Grøstl 224/256	10	10	Fugue 384	10	190
Grøstl 384/512	14	14	Luffa 384/512	8	24
JH	38	38	Fugue 512	13	277
Keccak 224/256/384/512	24	25	Simd 224/256/384/512	32(41)	36(45)
SHAvite3 224/256	36	37	Shabal	50	200
SHAvite3 384/512	56(70)	71	Skein	22	44

- As the size of the message to be hashed increases, these padding and finalisation stages will have less of an impact on the overall calculation time
- For short messages, they can have significant impact

Throughput

- In the design of the hash function architectures, our main goal was to give a baseline comparison between the hash functions using area and throughput

Throughput

Throughput

- In the design of the hash function architectures, our main goal was to give a baseline comparison between the hash functions using area and throughput

Throughput

$$\text{Throughput} = \frac{\text{\# Bits in a message block} \times \text{Maximum clock frequency}}{\text{\# Clock cycles per message block}}$$

Throughput

- In the design of the hash function architectures, our main goal was to give a baseline comparison between the hash functions using area and throughput

Throughput

$$\text{Throughput} = \frac{\text{\# Bits in a message block} \times \text{Maximum clock frequency}}{\text{\# Clock cycles per message block}}$$

- We also present Throughput/Area results to show which hash functions make best use of the FPGA

Throughput

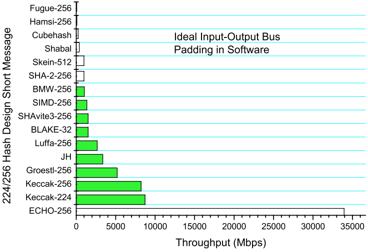
- In the design of the hash function architectures, our main goal was to give a baseline comparison between the hash functions using area and throughput

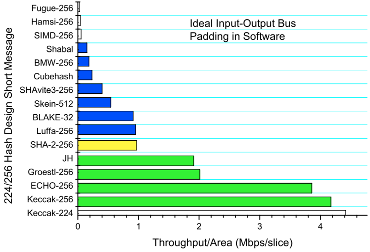
Throughput

$$\text{Throughput} = \frac{\text{\# Bits in a message block} \times \text{Maximum clock frequency}}{\text{\# Clock cycles per message block}}$$

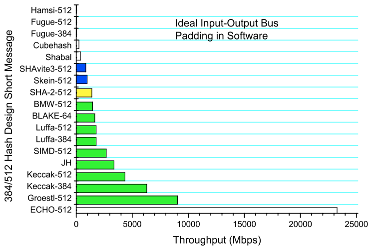
- We also present Throughput/Area results to show which hash functions make best use of the FPGA
- FPGA platform : Xilinx Virtex-5 *xc5vlx330T-2-ff1738*

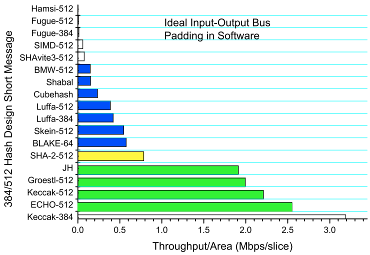
Unconstrained System, Short Message



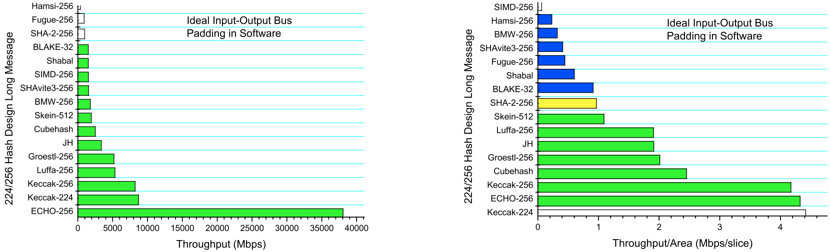


Unconstrained System, Short Message

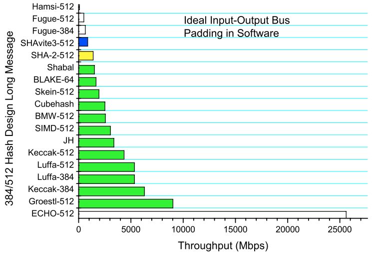


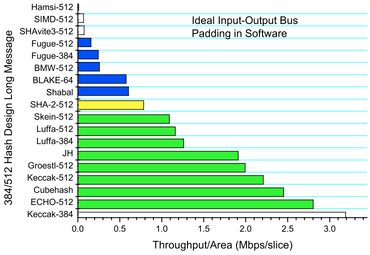


Unconstrained System, Long Message



Unconstrained System, Long Message





Load Latency

- The larger state sizes are affected by the loading latency, m/w clock cycles

Load Latency

- The larger state sizes are affected by the loading latency, m/w clock cycles
  - In cases where the time required to hash the message is larger than the time required to load the message this only affects the initial message loading

Load Latency

- The larger state sizes are affected by the loading latency, m/w clock cycles
  - In cases where the time required to hash the message is larger than the time required to load the message this only affects the initial message loading
  - In cases where the load time is longer than the hash time, there will be a wait delay as the hash waits for data

Load Latency

- The larger state sizes are affected by the loading latency, m/w clock cycles
  - In cases where the time required to hash the message is larger than the time required to load the message this only affects the initial message loading
  - In cases where the load time is longer than the hash time, there will be a wait delay as the hash waits for data
  - For these cases the clock count for the throughput needs to take this additional delay into consideration

Load $>$ Processing Time

- BMW, Echo, Grøstl, Keccak

Fixed Input Output Bus

Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
SHA224/256	16	0	0	64	1	65	0	0	65
SHA384/512	32	0	0	80	1	81	0	0	81
Blake224/256	16	0	0	10	4	40	0	0	40
Blake384/512	32	0	0	14	4	56	0	0	56
BMW224/256	**16**	0	0	1	4	**4**	1	3	**7**
BMW384/512	**32**	0	0	1	4	**4**	1	3	**7**
Cubehash	8	0	0	16	17	17	160	161	178
Echo224/256	**48**	0	0	8	1	**8**	1	1	**9**
Echo384/512	**32**	0	0	10	1	**10**	1	1	**11**
Fugue224/256	1	2	1	1	7	7	13	91	98
Fugue384	1	2	1	1	10	10	20	180	190
Fugue512	1	2	1	1	13	13	22	264	277
Grøstl224/256	**16**	0	0	10	1	**10**	0	0	**10**
Grøstl384/512	**32**	0	0	14	1	**14**	0	0	**14**
Hamsi224/256	1	3	1	3	2	6	6	24	31
Hamsi384/512	2	3	1	6	2	12	12	48	61

- 32-bit Input-Output Bus

Fixed Input Output Bus

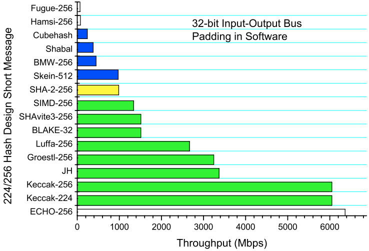
Hash Design	32-bit load #Cycles	Extra Padding	Padding #Cycles	Message Rounds	Round #Cycles	Long Msg #Cycles	Final Rounds	Final #Cycles	Short Msg #Cycles
JH	16	1	1	35	1	38	0	0	38
Keccak224	**36**	0	0	24	1	**25**	0	0	**25**
Keccak256	**34**	0	0	24	1	**25**	0	0	**25**
Keccak384	**26**	0	0	24	1	**25**	0	0	**25**
Keccak512	18	0	0	24	1	25	0	0	25
Luffa224/256	8	0	0	8	1	8	1	8	16
Luffa384	8	0	0	8	1	8	2	16	24
Luffa512	8	0	0	8	1	8	2	16	24
Shabal	16	0	0	1	50	50	3	150	200
SHAvite3-224/256	16	0	0	12	3	36	1	1	37
SHAvite3-384/512	32	0	0	14	4	56 (70)	1	1	71
Simd224/256	16	1	1	4	8	32(41)	0.5	4	36(45)
Simd384/512	32	1	1	4	8	32(41)	0.5	4	36(45)
Skein	16	0	0	18	22	22	18	22	44

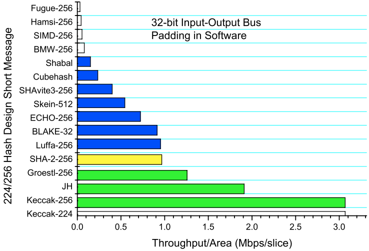
- 32-bit Input-Output Bus

Fixed Input Output Bus

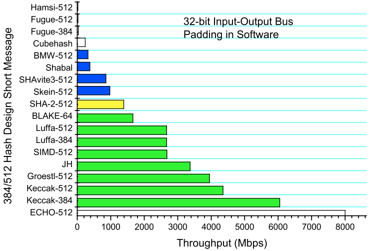
Hash Design	32-bit load (#Cycles)	Long Msg (#Cycles)	Short Msg (#Cycles)
BMW224/256	16	4	7
BMW384/512	32	4	7
Echo224/256	48	8	9
Echo384/512	32	10	11
Grøstl224/256	16	10	10
Grøstl384/512	32	14	14
Keccak224	36	25	25
Keccak256	34	25	25
Keccak384	26	25	25

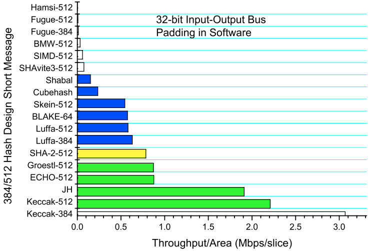
Fixed Input Output Bus, Short Message



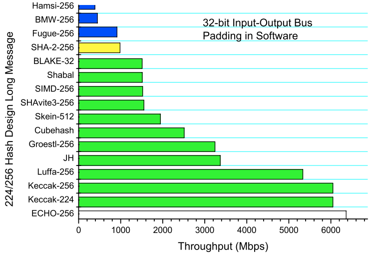


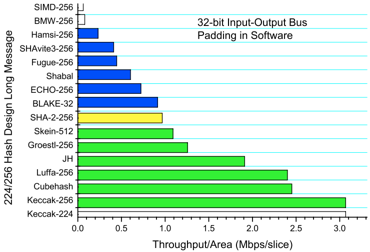
Fixed Input Output Bus, Short Message



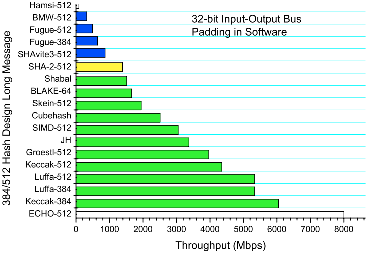


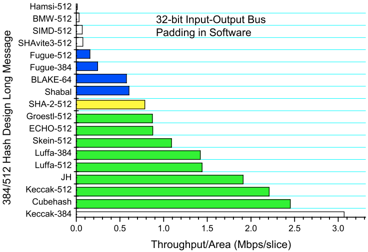
Fixed Input Output Bus, Long Message





Fixed Input Output Bus, Long Message





Padding

- In some cases, counters in the padding block form the critical path and thus affect the maximum frequency

Padding Critical Path

Padding

- In some cases, counters in the padding block form the critical path and thus affect the maximum frequency

Padding Critical Path

- Echo-256

Padding

- In some cases, counters in the padding block form the critical path and thus affect the maximum frequency

Padding Critical Path

- Echo-256
- Fugue-256/512

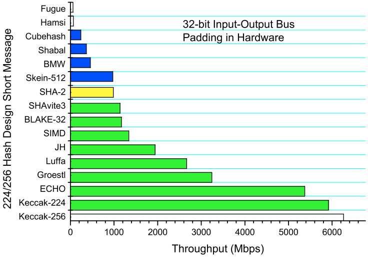
Padding

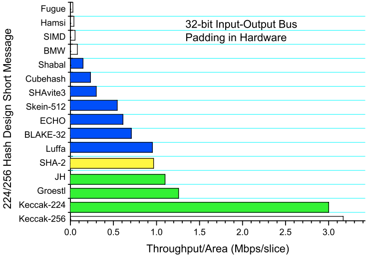
- In some cases, counters in the padding block form the critical path and thus affect the maximum frequency

Padding Critical Path

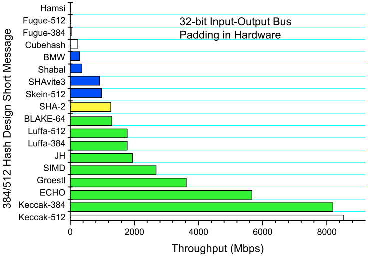
- Echo-256
- Fugue-256/512
- JH
  - Hash Max Frequency: 250.125 MHz
  - Wrapper Max Frequency: 144.113 MHz

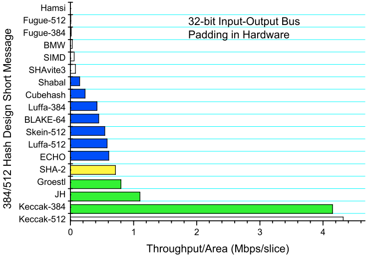
Padding in Hardware, Short Message



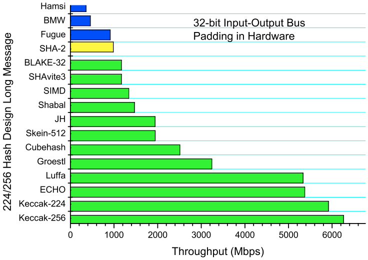


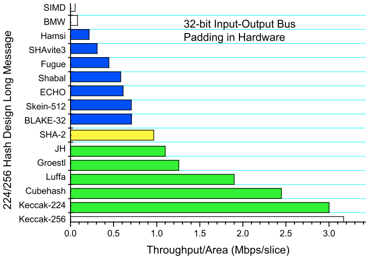
Padding in Hardware, Short Message



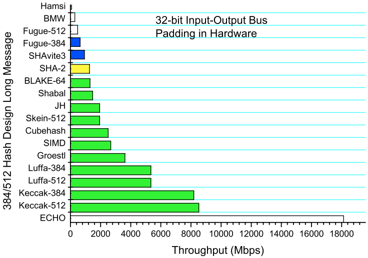


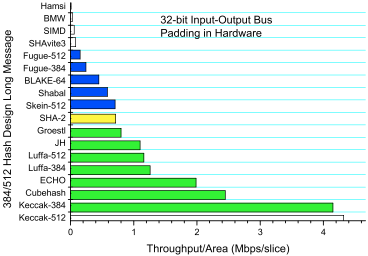
Padding in Hardware, Long Message





Padding in Hardware, Long Message





Top 10 Performers

- Three designs consistently outperformed SHA-2 in all above categories

Top Three

Top 10 Performers

- Three designs consistently outperformed SHA-2 in all above categories

Top Three

- 1: Keccak
- 2: Grøstl
- 3: JH

Top 10 Performers

- Six designs outperformed SHA-2 at least half of the time

Top 10 Performers

- Six designs outperformed SHA-2 at least half of the time
  - 4: ECHO
  - 5: Luffa
  - 6: Cubehash
  - 7: Skein(512)
  - 8: SIMD
  - 9: BLAKE

Top 10 Performers

- Six designs outperformed SHA-2 at least half of the time
  - 4: ECHO
  - 5: Luffa
  - 6: Cubehash
  - 7: Skein(512)
  - 8: SIMD
  - 9: BLAKE
- Two designs came close

Top 10 Performers

- Six designs outperformed SHA-2 at least half of the time
  - 4: ECHO
  - 5: Luffa
  - 6: Cubehash
  - 7: Skein(512)
  - 8: SIMD
  - 9: BLAKE
- Two designs came close
  - 10: Shabal
  - 11: SHAvite-3

Top 11 Performers

- 1: Keccak
- 2: Grøstl
- 3: JH
- 4: ECHO
- 5: Luffa
- 6: Cubehash
- 7: Skein(512)
- 8: SIMD
- 9: BLAKE
- 10: Shabal
- 11: SHAvite-3

Thank you for your time
Any Questions?

- brianb, neilh, markh, andrewb, liam@eleceng.ucc.ie
- l.lu, m.oneill@ecit.qub.ac.uk