Implementing SHA-1 and SHA-2 Standards on the Eve of SHA-3 Competition

Marcin Rogawski, Xin Xin,
Ekawat 'Ice' Homsirikamol, David Hwang
and Kris Gaj

Cryptographic Engineering Research Group
George Mason University (USA)

Motivation

- existing standard FIPS 180-3
- research interest of industry and academia (interesting research done before)
- reference point for SHA-3 competition
- primary goal – highest throughput (but also best trade-off of throughput vs. area)
- single stream of data (one message) implementations

SHA-1 and SHA-2 facts

	SHA-1	SHA-256	SHA-512
publication in year	1993	2001	2001
output digest size	160	256	512
best attack complexity	2^{63}	2^{128}	2^{256}
number of rounds	80	64	80
word size	32	32	64
input block size	512	512	1024
other output size support	-	224	384

SHA-1 and SHA-2 pseudo code

$A=HA=IV_A$, $B=HB=IV_B$, $C=HC=IV_C$,
$D=HD=IV_D$, $E=HE=IV_E$,

for each data_block do

for (t=0; t<80; t++)
W_t = msg_scheduler(data_block)
$T = RotL^5(A) + f_t(B, C, D)$
$+ E + K_t + W_t$
$E = D$, $D = C$, $C = RotL^{30}(B)$,
$B = A$, $A = T$
end for

HA=A=HA+A, HB=B=HB+B,
HC=C=HC+C, HD=D=HD+D,
HE=E=HE+E,
end for

$HA=A=IV_A$, $HB=B=IV_B$, $HC=C=IV_C$, $HD=D=IV_D$,
$HE=E=IV_E$, $HF=F=IV_F$, $HG=G=IV_G$, $HH=H=IV_H$,

for each data_block do

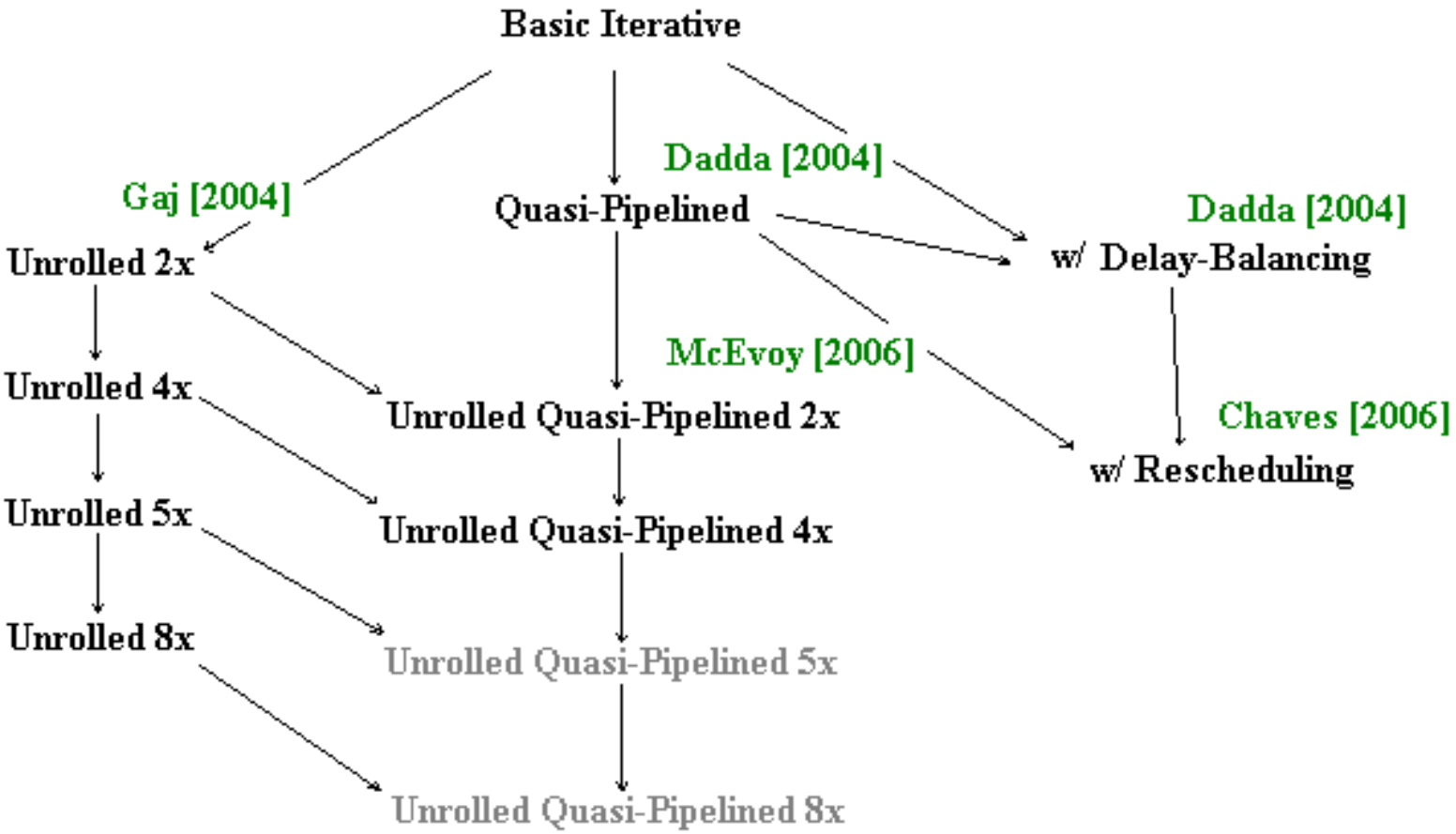
for (t=0; t<ROUNDS; t++)
W_t = msg_scheduler(data_block)
$T_1 = H + \Sigma_1(E) + Ch(E, F, G) + K_t + W_t$
$T_2 = \Sigma_0(A) + Maj(A, B, C)$
$H = G$, $G = F$, $F = E$, $E = D + T_1$,
$D = C$, $C = B$, $B = A$, $A = T_1 + T_2$
end for

HA=A=HA+A, HB=B=HB+B, HC=C=HC+C,
HD=D=HD+D, HE=E=HE+E, HF=F=HF+F,
HG=G=HG+G, HH=H=HH+H,
end for

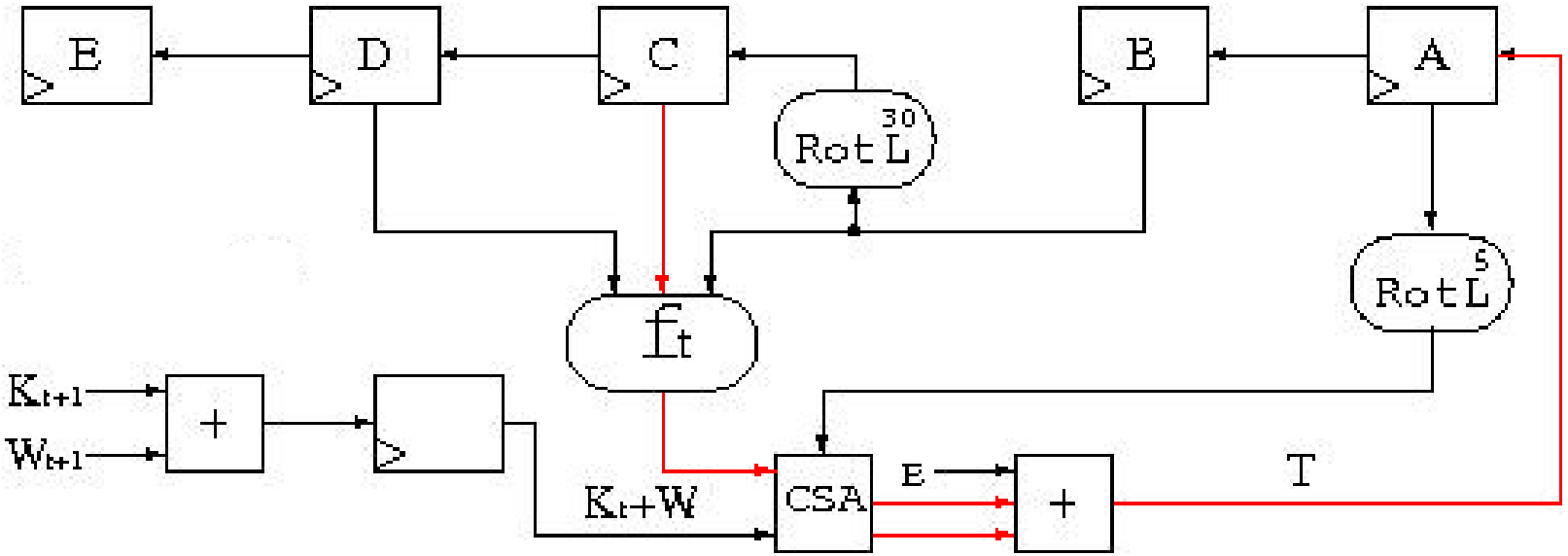
Optimization techniques for hash functions

- moving one addition to the message block expansion stage
- use of parallel counters or well balanced Carry Save Adders (CSA)
- use of BRAMs for constants
- unrolling – Gaj et al. [2004]
- quasi-pipelining – Dadda et al. [2004]
- delay-balancing – Dadda et al. [2004]
- unrolled quasi-pipelining – McEvoy et al.[2006]
- rescheduling – Chaves et al. [2006, 2008]

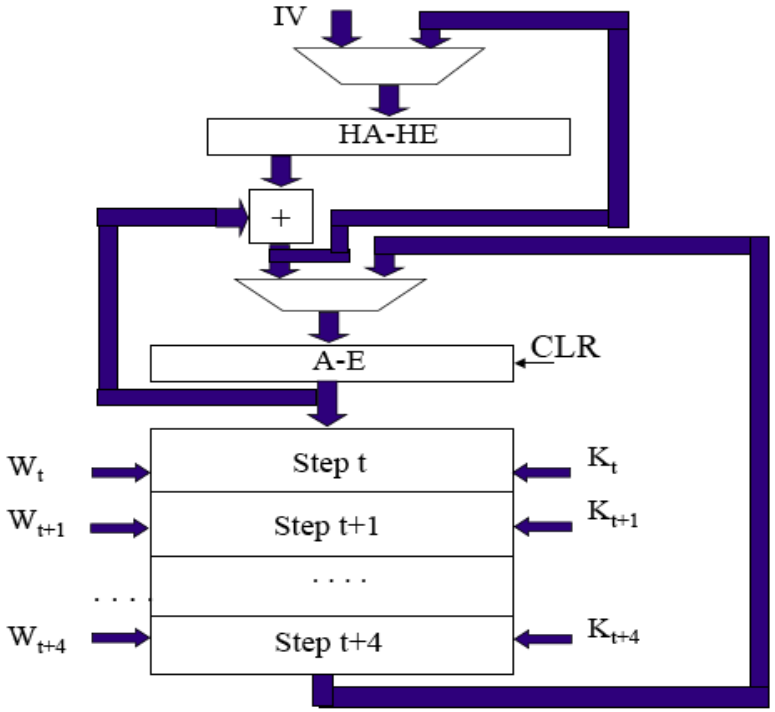
Development of SHA hardware architectures

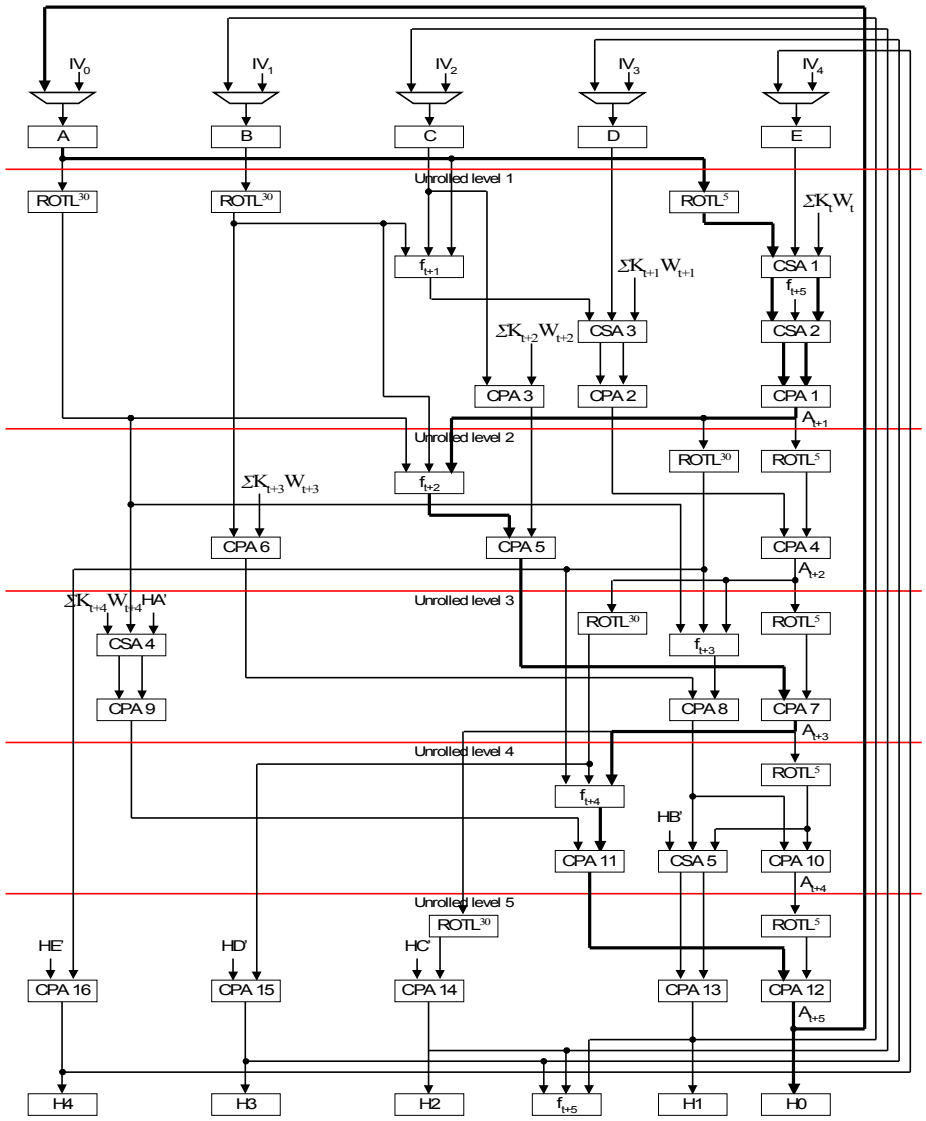


Basic Iterative architecture of SHA-1



5 x Unrolled Architecture of SHA-1 (Gaj et al.)





Latency = $(1 + 16*n)T = (1 + 80*n)T_{MC}$

T – clock period from regular analysis

T_{MC} – fast clock period from multi-cycle path analysis

n – number of message blocks

Quasi-Pipelining in SHA-1 (Dadda et al)

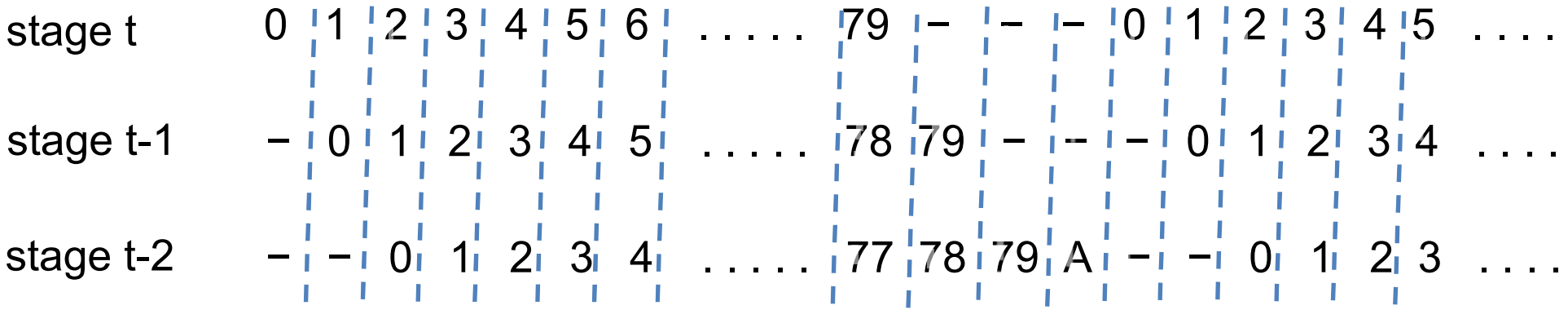
$$A_{t+1} = T_t = K_t + W_t + E_t + f_t(B_t, C_t, D_t) + RotL^5(A_t)$$

stage t $(CM_t, SM_t) = CSA(K_t, W_t, E_t)$

stage t-1 $(CL_{t-1}, SL_{t-1}) = CSA(CM_{t-1}, SM_{t-1}, f_{t-1}(B_{t-1}, C_{t-1}, D_{t-1}))$

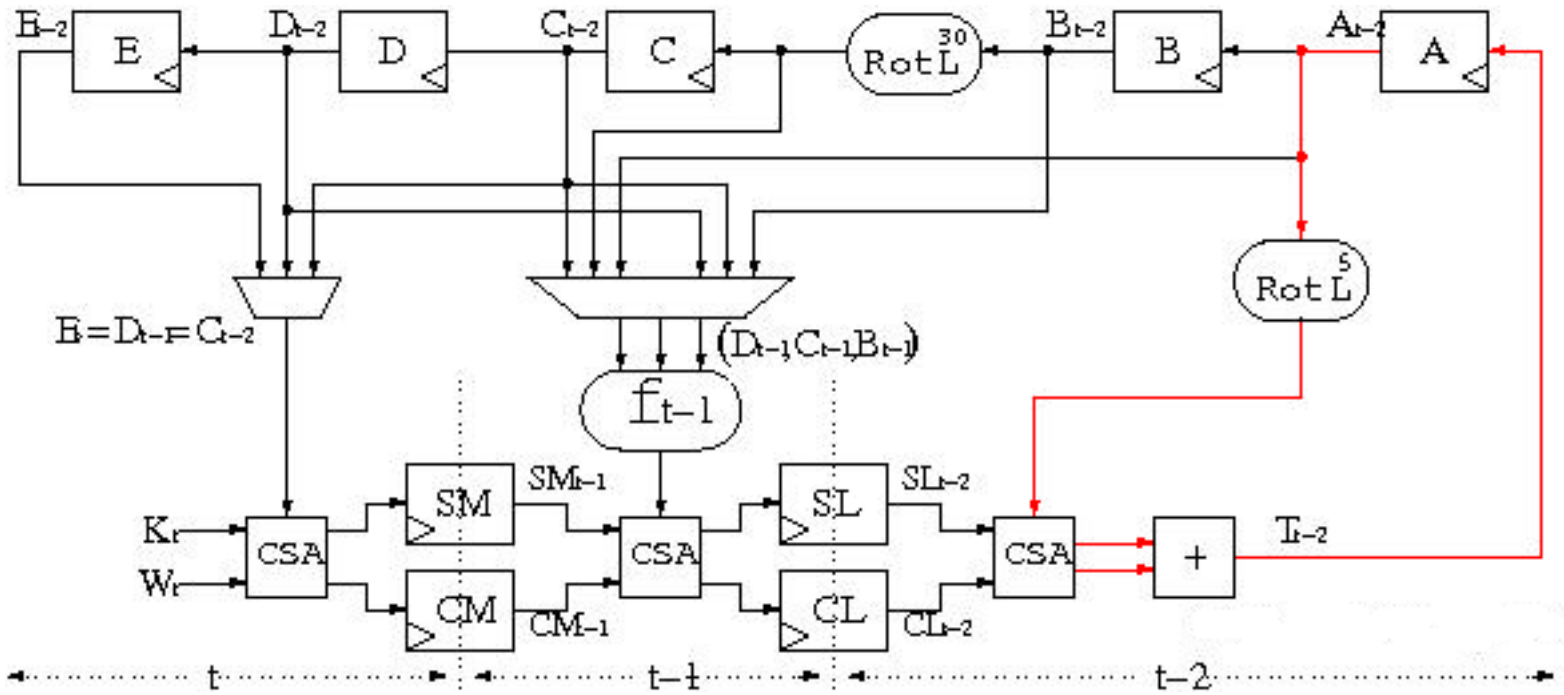
stage t-2 $(CT_{t-2}, ST_{t-2}) = CSA(CL_{t-2}, SL_{t-2}, RotL^5(A_{t-2}))$

$T_{t-2} = CT_{t-2} + ST_{t-2}$



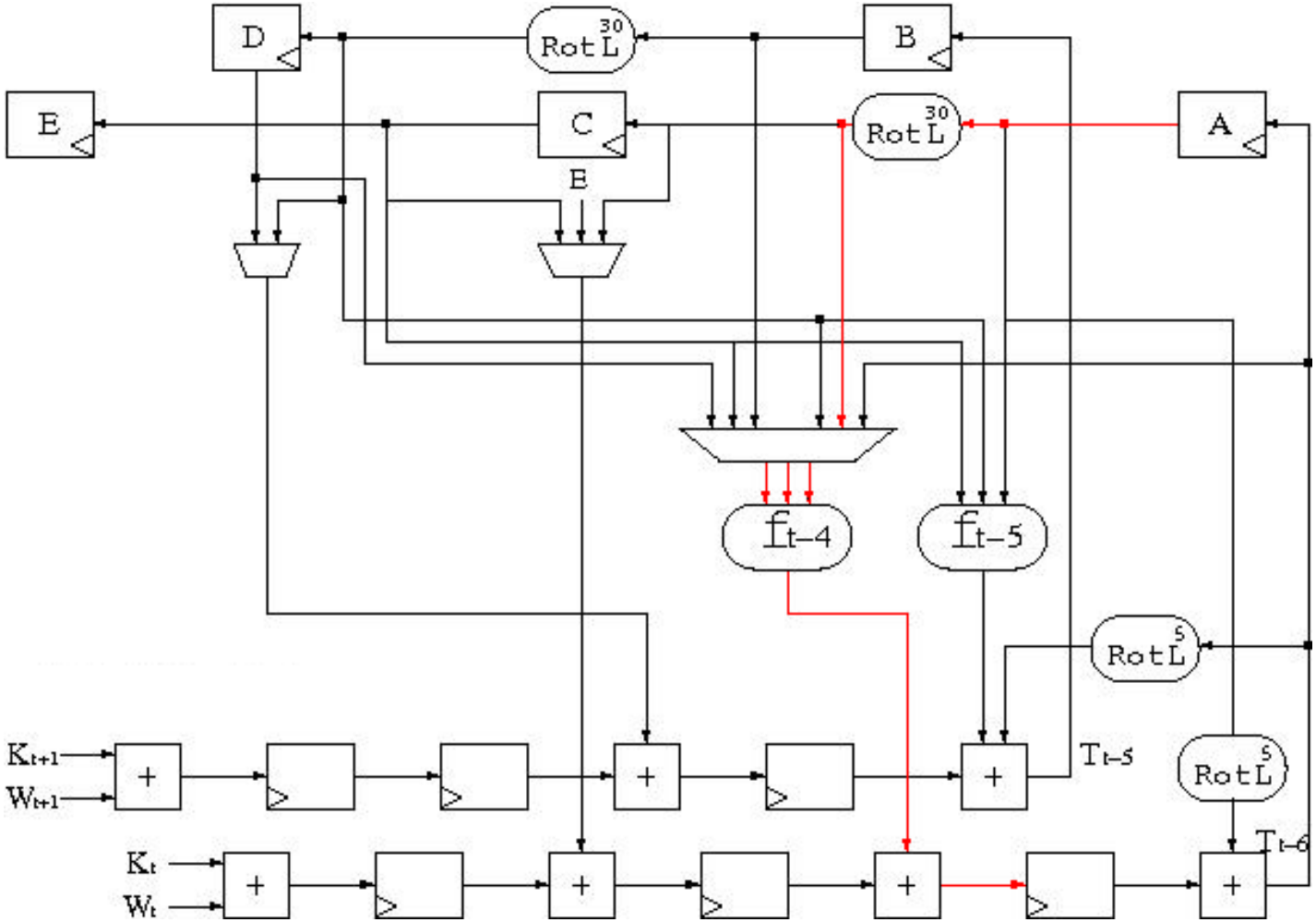
Round numbers for each stage
A – final addition = calculation of new values of A..E using HA..HE

Quasi-Pipelining in SHA-1

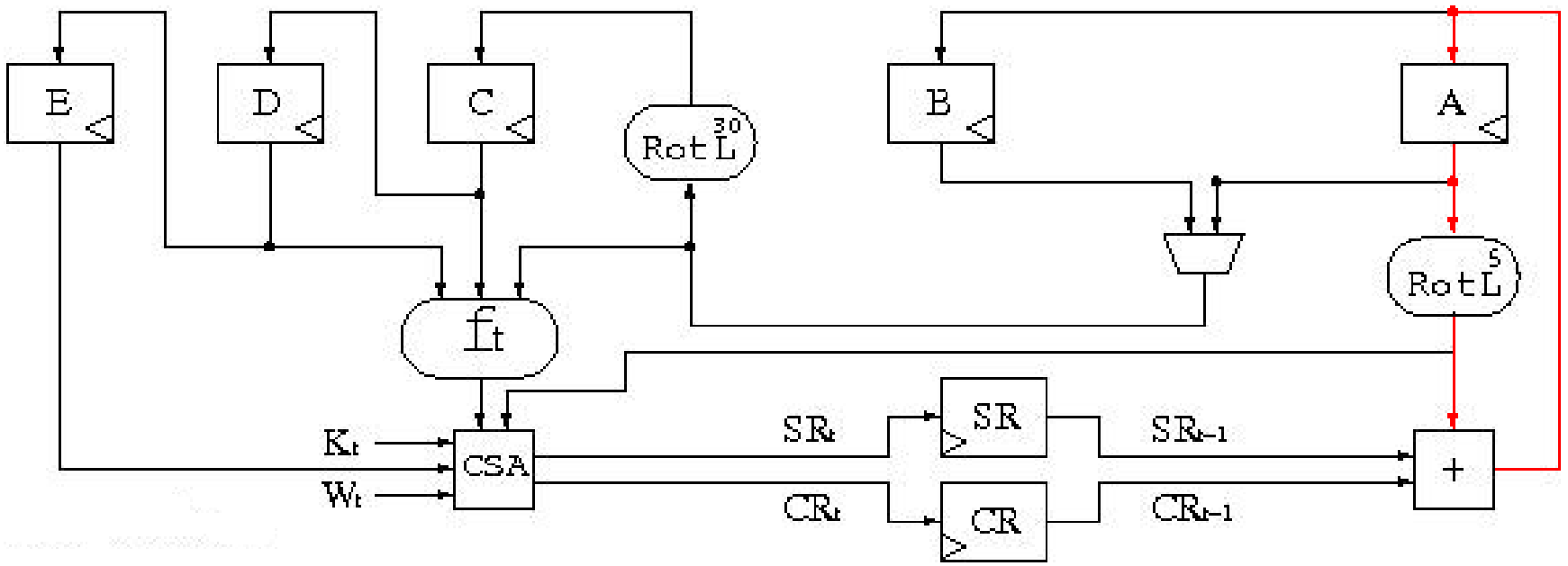


stage t	0	1	2	3	4	5	6	. . . . .	79	–	–	–	0	1	2	3	4	5	. . . .
stage t-1	–	0	1	2	3	4	5	. . . . .	78	79	–	–	–	0	1	2	3	4	. . . .
stage t-2	–	–	0	1	2	3	4	. . . . .	77	78	79	A	–	–	0	1	2	3	. . . .
e_M	1	1	1	1	1	1	1	. . . . .	1	0	0	0	1	1	1	1	1	1	. . . .
e_L	0	1	1	1	1	1	1	. . . . .	1	1	0	0	0	1	1	1	1	1	. . . .
e_{AH}	0	0	1	1	1	1	1	. . . . .	1	1	1	1	0	0	1	1	1	1	. . . .

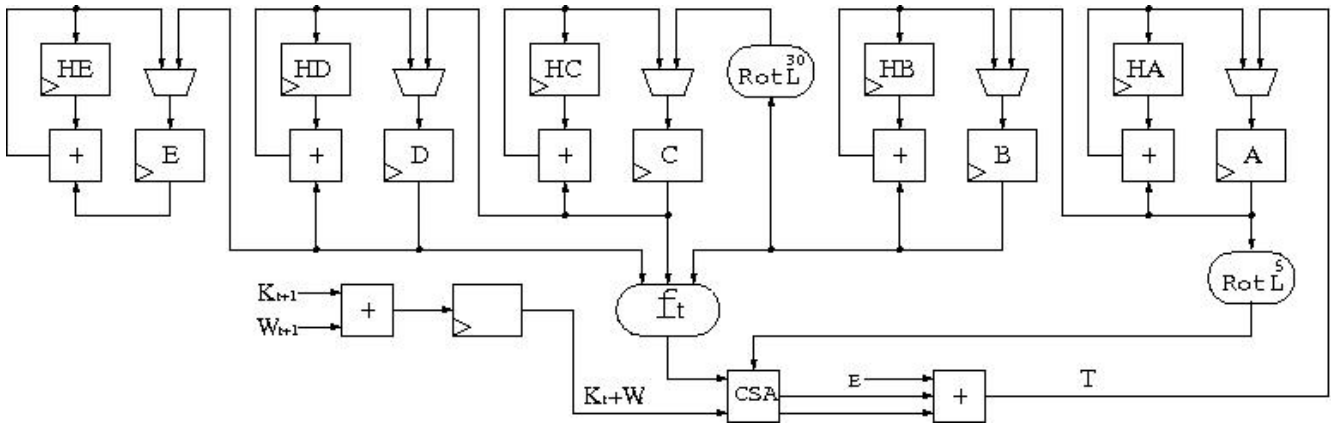
Unrolled Quasi-Pipelined architecture (McEvoy et al.)



Basic Iterative architecture (Chaves et al. - rescheduling)



Final addition



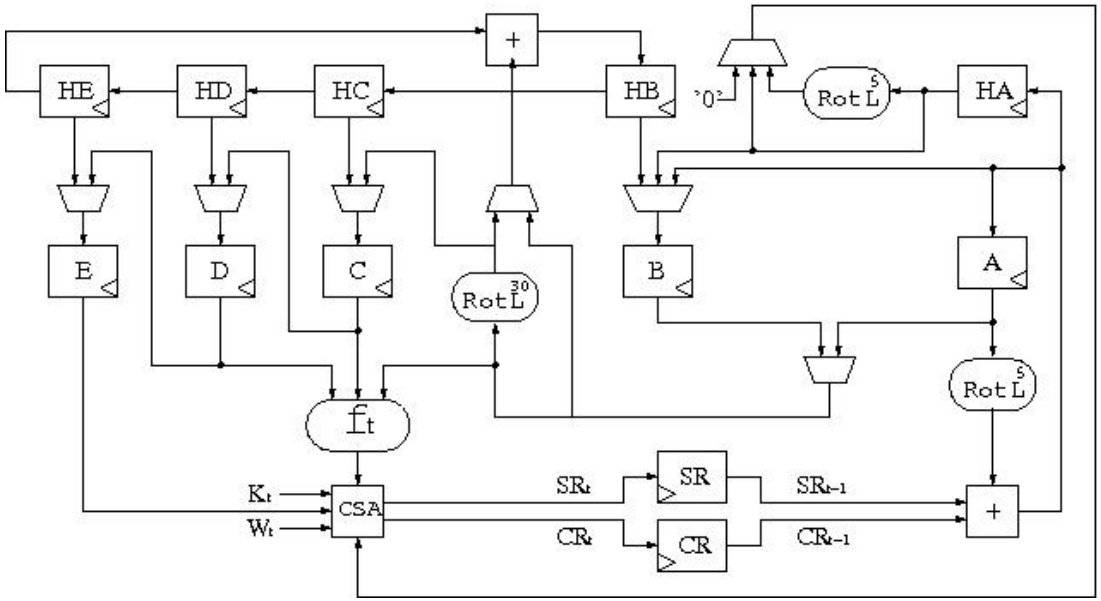
$$E_t = D_{t-1} = C_{t-2} = RotL^{30}(B_{t-3})$$

$$HE_i = E_{80} + HE_{i-1} = RotL^{30}(B_{77}) + HE_{i-1}$$

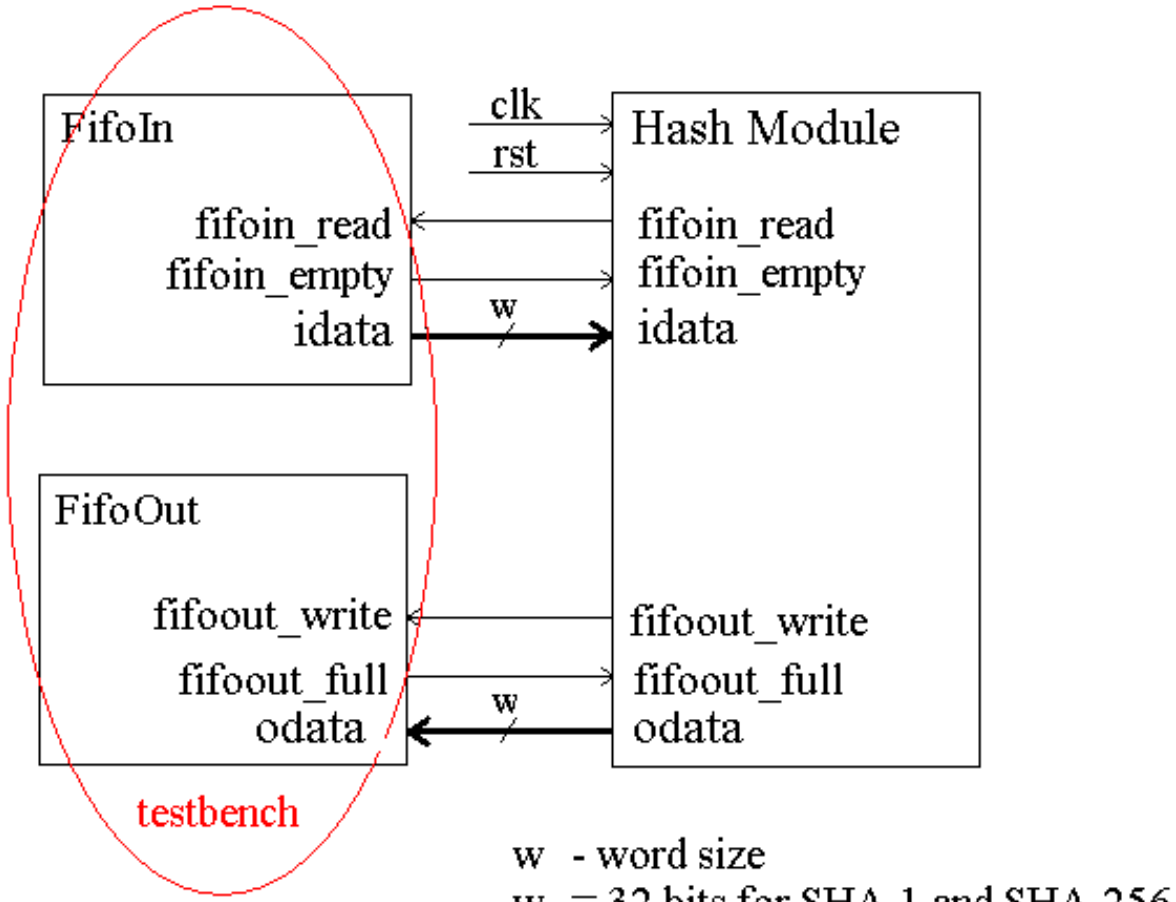
$$HD_i = D_{80} + HD_{i-1} = RotL^{30}(B_{78}) + HD_{i-1}$$

$$HC_i = C_{80} + HC_{i-1} = RotL^{30}(B_{79}) + HC_{i-1}$$

$$HB_i = B_{80} + HB_{i-1}$$



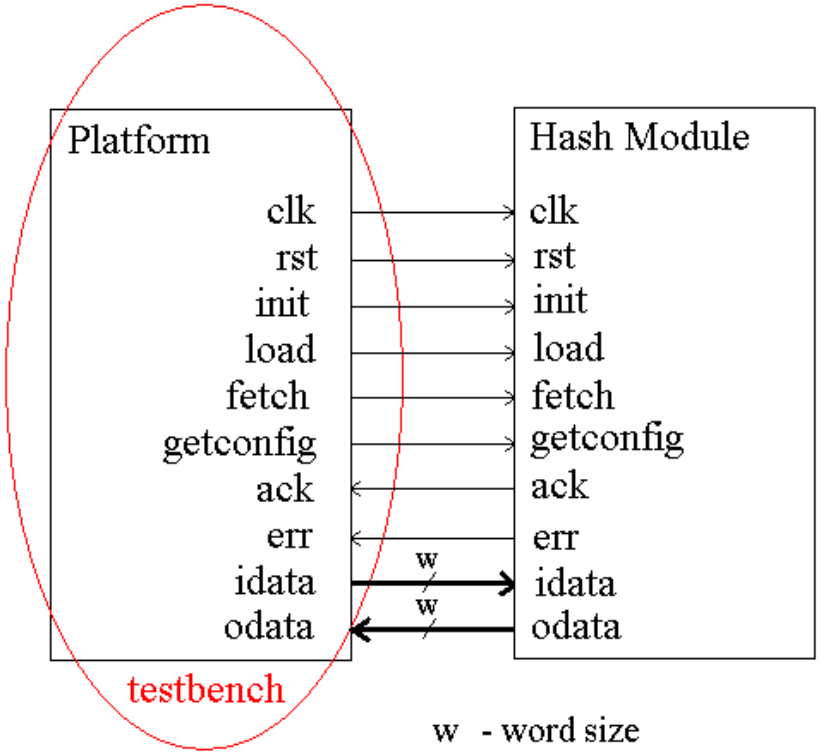
GMU Interface



w - word size
w = 32 bits for SHA-1 and SHA-256
w = 64 bits for SHA-512

VT Interface Support

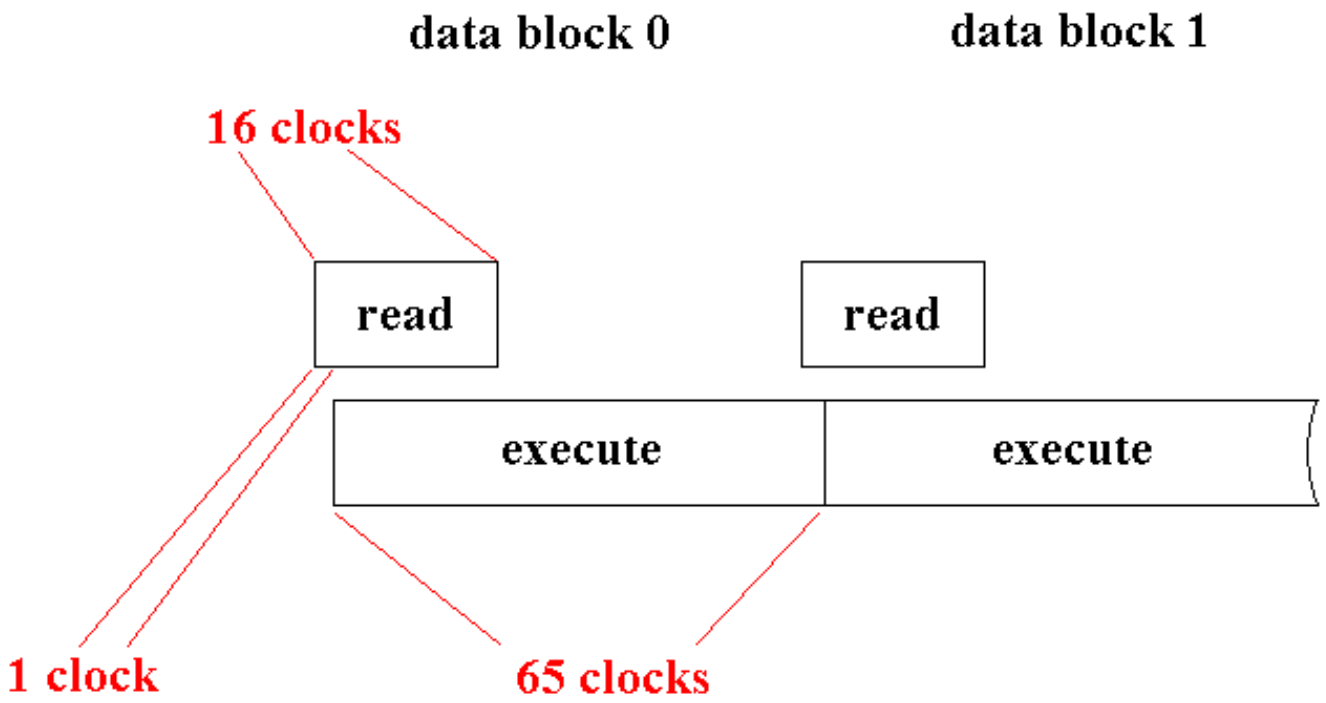
- http://eprint.iacr.org/2008/529



w - word size
w = 32 bits for SHA-1 and SHA-256
w = 64 bits for SHA-512

Overlapping Effect

SHA-256 example



Implementation Environment

- Synthesis: Simplify Pro 8.6, XST WebPack 9.1
- Implementation: ISE WebPack 9.1
- Simulation: ModelSim 6.3 SE, Active-HDL 7.2 SE
- Devices: Virtex, Virtex II Pro, Virtex 5
- Application of ATHENa project
- Specified interface (GMU and VT interface support http://eprint.iacr.org/2008/529)

Results – Throughput (Virtex II Pro)

	SHA-1	SHA-256	SHA-512
Basic Iterative	1053	1096	1138
Rescheduling	1410	1316	1723
Unrolled x2	-	1395	1840
Unrolled x4	-	1428	1957
Unrolled x5	1484	-	-
Unrolled x8	-	1557	*
Quasi-Pipelining	1406	1428	1897
U2 Quasi-Pipelining	1595	1992	2685

SHA-1 results comparison

Designer	Device	Throughput
Intron Ltd.	Virtex	450
GMU Rescheduling	**Virtex**	**660**
Ocean Logic	Virtex II	502
Alma Technologies	Virtex II Pro	1008
Helion Technology	Virtex II Pro	1211
Chaves et al. Rescheduling	Virtex II Pro	1420
GMU Rescheduling	**Virtex II Pro**	**1410**
GMU U2 Quasi-Pipelining	**Virtex II Pro**	**1595**
Helion Technology	Virtex-5	1960
GMU Rescheduling	**Virtex-5**	**2165**

SHA-256 results comparison

Designer	Device	Throughput
Chaves et al. Rescheduling	Virtex	646
GMU Rescheduling	**Virtex**	**647**
GMU U2 quasi-pipelining	**Virtex**	**856**
Alma Technologies	Virtex II Pro	947
Helion Technology	Virtex II Pro	977
Chaves et al. Rescheduling	**Virtex II Pro**	**1370**
GMU Rescheduling	**Virtex II Pro**	**1316**
GMU quasi-pipelining	**Virtex II Pro**	**1428**
GMU U2 quasi-pipelining	**Virtex II Pro**	**1992**
Alma Technologies	Virtex 5	947
Helion Technology	Virtex 5	1720
GMU Rescheduling	**Virtex-5**	**1741**
GMU quasi-pipelining	**Virtex 5**	**2128**
GMU U2 quasi-pipelining	**Virtex 5**	**2892**

SHA-512 results comparison

Designer	Device	Throughput
Chaves et al. Rescheduling	Virtex	889
GMU Rescheduling	**Virtex**	**871**
GMU Quasi-Pipelining	**Virtex**	**896**
Chaves et al. Rescheduling	Virtex II Pro	1780
GMU Rescheduling	**Virtex II Pro**	**1723**
GMU Quasi-Pipelining	**Virtex II Pro**	**1897**
GMU U2 Quasi-Pipelining	**Virtex II Pro**	**2685**
Helion Technology	Virtex-5	2345
GMU Rescheduling	**Virtex-5**	**2029**
GMU Quasi-Pipelining	**Virtex 5**	**2340**
GMU U2 Quasi-Pipelining	**Virtex 5**	**3902**

Summary

- All designs for single stream of data,
- Both quasi-pipeline and unrolling techniques outperform several basic approaches,
- Trade-off speed-area (basic → unrolling and quasi-pipelining →unrolled quasi-pipelining),
- Chaves final addition optimization applicable for quasi-pipelining
- Project easily applicable to VT interface
- Implementations as a reference to SHA-3 contest

Possible Future Work

- Front-end libraries ASIC's implementation
- Unrolled x5, x8 quasi-pipelined architectures
- Compact implementation of SHA-1/SHA-2
- Resource sharing SHA-1/SHA-2 implementation
- Result Generation for Altera and Actel devices (ATHENa project)

References

- http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf
- R. Lien, T. Grembowski, K. Gaj, "A 1 Gbit/s Partially Unrolled Architecture of Hash Functions SHA-1 and SHA-512," LNCS 2964, RSA Conference 2004, Cryptographers' Track, CT-RSA 2004, San Francisco, CA, Feb. 2004
- L. Dadda, M. Macchetti, J. Owen: The Design of a High Speed ASIC Unit for the Hash Function SHA-256 (384, 512). DATE 2004
- L. Dadda, M. Macchetti, J. Owen: An ASIC design for a high speed implementation of the hash function SHA-256 (384, 512). ACM Great Lakes Symposium on VLSI 2004
- R. P. McEvoy, F. M. Crowe, C. C. Murphy and W. P. Marnane: “Optimisation of the SHA-2 Family of Hash Functions on FPGAs” IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI’06), pp. 317–322, 2006
- R. Chaves, G. Kuzmanov, L. Sousa, S. Vassiliadis, Improving SHA-2 Hardware Implementations, In Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer, pages 298-310, October 2006
- R. Chaves, G. Kuzmanov, L. Sousa, S. Vassiliadis: Cost-Efficient SHA Hardware Accelerators. IEEE Trans. VLSI Syst. 16(8): 999-1008, 2008