2.B Algorithm Specifications and Supporting Documentations

# 2.B.2 STATEMENT of the NaSHA's estimated computational efficiency and memory requirements in hardware and software

We have programmed NaSHA hash algorithm in HDL/Verilog language using free (Nisc) System Generator - Center for Embedded Computer Systems - University of California and with Xilinx ISE 10.1 application we mapped the algorithm in Virtex 2 device family. Projected logic utilizations are given in Table 1.

| Device Family | No. Slices | No. Slice Flip Flops | No. 4 input LUTs | No. bounded IOBs | No. MULT 18x18s | No. GCLKs |
|---|---|---|---|---|---|---|
| virtex2 Device xc2v2000/ Package ff896 Speed -6 | 5802/ 10752 | 1264/ 21504 | 10679/ 21505 | 67/624 | 3/56 | 2/16 |

Table 1: Projected logic utilizations of different FPGA platforms

Starting memory requirements for implementing NaSHA algorithm are quite small, only 0.625KB (0.25KB for starting bijection and 0.375 KB for 48 64-bit initial values).

## 1 NIST SHA-3 Reference Platform

**1.**

**a. Description of the platform**: Wintel personal computer, with

an Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 32-bit (x86) Edition. **Compiler**: the ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition.

**b. Speed estimate**: A comparison of NaSHA-$(m, 2, 6)$ performance in Cycles/Byte Versus Message on 32-bit architecture, where $m \in \{224, 256, 384, 512\}$ is given in the Table 2.

| Length (bytes) | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| NaSHA–$(224, 2, 6)$ | 3018.00 | 303.20 | 55.24 | 40.64 | 39.35 | 38.97 |
| NaSHA-$(256, 2, 6)$ | 3039.00 | 299.70 | 55.24 | 40.58 | 39.35 | 39.01 |
| NaSHA-$(384, 2, 6)$ | 5629.00 | 567.10 | 56.15 | 39.22 | 38.07 | 37.64 |
| NaSHA-$(512, 2, 6)$ | 5867.00 | 586.70 | 58.25 | 39.47 | 38.10 | 38.97 |

Table 2: Performance in Cycles/Byte Versus Message of NaSHA-$(m, 2, 6)$, where $m \in \{224, 256, 384, 512\}$ on 32-bit architecture

The set up of NaSHA hash algorithm requires 71 cycles for NaSHA-$(224, 2, 6)$ and NaSHA-$(256, 2, 6)$ and 85 cycles for NaSHA-$(384, 2, 6)$ and NaSHA-$(512, 2, 6)$.

**c. Speed/memory tradeoffs**: One way to change NaSHA performances is to take as starting bijection a permutation of order $2^{16}$, instead of $2^8$. The cost of this replacement is the usage of larger memory of 64KB, instead of 0.25KB. In that case the smallest works used in the algorithm will consist of 16 bits, instead of 8 bits used in the present algorithm. In such a way, by decreasing the number of operations, we can increase the performances. On the other side, searching on a larger table will decrease performances again. It is an open question the behavior of the overall performances in this case. Another problem is the construction of suitable permutation of order $2^{16}$, that we could not resolve, and we could not make a comparison between these two options.

We can speed the performances of NaSHA-$(m, k, 6)$, $m \in \{224, 256, 384, 512\}$, by working with quasigroups of order $2^{128}$ or $2^{256}$ (i.e., for $r = 7$ or $r = 8$). Our opinion is that in that case, the security will be somewhat weakened if permutation of order $2^8$ is used. We think that the same level of security as NaSHA-$(m, k, 6)$ can be obtained for NaSHA-$(m, k, 7)$ if we use a permutation of order $2^{16}$.

If we use a higher complexity $k = 4$, instead of $k = 2$, we are obtaining speed slowdowns by factors that ranges from 1.75 to 1.9 for NaSHA-

$(224, k, 6)$ and NaSHA-$(256, k, 6)$ and from 1.78 to 2 for NaSHA-$(384, k, 6)$ and NaSHA-$(512, k, 6)$.

**2.**

**a. Description of the platform**: Wintel personal computer, with an Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 64-bit (x64) Edition. **Compiler**: the ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition.

**b. Speed estimate**: A comparison of NaSHA-$(m, 2, 6)$ performance in Cycles/Byte Versus Message on 64-bit architecture, where $m \in \{224, 256, 384, 512\}$ is given in the Table 3.

| Length (bytes) | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| NaSHA-$(224, 2, 6)$ | 1912.00 | 196.10 | 37.60 | 28.84 | 28.26 | 28.31 |
| NaSHA-$(256, 2, 6)$ | 1968.00 | 198.90 | 37.88 | 29.07 | 28.39 | 28.43 |
| NaSHA-$(384, 2, 6)$ | 3977.00 | 402.60 | 40.19 | 30.30 | 29.77 | 29.39 |
| NaSHA-$(512, 2, 6)$ | 4075.00 | 406.80 | 40.68 | 30.33 | 29.65 | 29.39 |

Table 3: NaSHA Performance in Cycles/Byte Versus Message Length on 64-bit architecture

The set up of of NaSHA hash algorithm requires 15 cycles for NaSHA-$(224, 2, 6)$ and NaSHA-$(256, 2, 6)$ and 22 cycles for NaSHA-$(384, 2, 6)$ and NaSHA-$(512, 2, 6)$.

**c. Speed/memory tradeoffs**: If we use a higher complexity $k = 4$, instead of $k = 2$, we are obtaining speed slowdowns by factors around 2 for NaSHA-$(m, k, 6)$, $m \in \{224, 256, 384, 512\}$2.

## 2 8-bit processor

**a. Description of the platform**: Intel 8052AH, 12MHz clock speed, 8 KB ROM, 256B on-chip RAM. **Environment**: $\mu$Vision IDE/ Debugger/ Simulator, RTX-51 Full OS **Compiler**: KEIL C51 ANSI C compiler.

**b. Speed estimate**: A comparison of NaSHA-$(m, 2, 6)$ performance in Cycles/Byte Versus Message on 8-bit architecture, where $m \in \{224, 256, 384, 512\}$ is given in the Table 4.

This is surely not the optimal implementation for an 8-bit architec-

| Length (bytes) | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| NaSHA-$(224, 2, 6)$ | 531919.50 | 53114.85 | 5274.66 | 890.54 | 397.22 | 358.70 |
| NaSHA-$(256, 2, 6)$ | 544516.50 | 54374.55 | 5400.63 | 903.14 | 398.48 | 358.83 |
| NaSHA-$(384, 2, 6)$ | 821146.50 | 82037.55 | 8166.93 | 1178.76 | 406.64 | 358.69 |
| NaSHA-$(512, 2, 6)$ | 911539.50 | 91076.85 | 9070.86 | 1269.14 | 415,67 | 359,60 |

Table 4: NaSHA Performance in Cycles/Byte Versus Message Length on 8-bit architecture

ture. The set up of NaSHA algorithm requires 1099272 cycles for NaSHA-$(224, 2, 6)$, 1099512 cycles for NaSHA-$(256, 2, 6)$, 1385004 cycles for NaSHA-$(384, 2, 6)$ and 1385244 cycles NaSHA-$(512, 2, 6)$. Estimated working memory is 62KB.

We want to acknowledge Simona Samardziski for her 8-bit implementation of NaSHA hash algorithm and hardware estimations.

**c. Speed/memory tradeoffs**: If instead of $k = 2$ in NaSHA-$(m, k, 6)$, $m \in \{224, 256, 384, 512\}$, we use higher complexity $k = 4$, we obtain slowdowns by factors around 2.