SWASAD: AN ASIC DESIGN FOR HIGH SPEED DNA SEQUENCE MATCHING

Tony Han
Computer Science and Electrical Engineering
The University of Queensland
QLD 4072 , Australia

Sri Parameswaran
Computer Science and Engineering
The University of New South Wales
NSW 2052, Australia

Abstract *This paper presents the Smith and Waterman Algorithm-Specific ASIC Design (SWASAD) project. This is a hardware solution that implements the S&W algorithm.. The SWASAD is an improved implementation of the Biological Information Signal Processor (BISP) design [1], The SWASAD chip fabricated on a 0.5 µm process achieves 3200 million Matrix Cells Per Second (MCPS) per chip, with a layout size of 7.1 mm by 7.1 mm. This is a large improvement over existing designs and improves data throughput by using a smaller datawidth.*

1. Introduction

1.1 Motivation

In recent times, large, government-sponsored scientific projects such as the Human Genome Project in the United States, have contributed to a massive increase in the quantity of data regarding proteins, DNA, and RNA. When scientists search for similar sequences amongst a vast array of sequences, the speed of search is of importance, particularly when a search on a software-based system can take up to a month to complete. The Smith and Waterman (S&W) algorithm is one of the fundamental methods of analysing and searching the large amounts of data now available. However, an alignment algorithm such as this requires a large computational load which leads to a performance bottleneck on CPU-based computers and servers. As researchers investigate larger organisms such as humans, the need for high performance sequence analysis and database similarity search become even more demanding. . An application-specific hardware design, which is both high speed and cost-effective is the best solution for the required computational load.
The SWASAD is an application-specific hardware system that executes the DNA sequence comparison algorithm 'Smith and Waterman' faster and with a lower engineering design cost than existing products (for example, the Kestrel chip). The SWASAD is an improved implementation of the BISP design [1]. The SWASAD design has used an improved algorithm, and has

1.2 Related Work

Algorithms for sequence comparison can be categorized into two groups: dynamic and heuristic. Dynamic algorithms give optimal solutions, and well known searching algorithms like S&W, Needleman & Wunch (N&W) and Hidden Markov Models (HMM) are of the dynamic kind. Examples of heuristic algorithms are the BLAST, FASTA and Feng & Doolittle algorithms. Heuristic algorithms are statistically driven sequence searching and alignment methods, and might not be as sensitive for all protein searches as the full dynamic programming algorithms such as the S&W and N&W algorithms.
Software approaches for the above algorithms are the heuristic programs such as BLAST [2] for the Blast Algorithm, FASTA [3] for the FASTA Algorithm, and HMMER [4] for HMM algorithm.
The following are the well-known hardware approaches to the sequence matching problem. *DeCypher Accelerator* [5] is a reconfigurable computer that runs CPU-based and hardware methods on the same system for both heuristic and dynamic algorithms. *SAMBA* [6, 7] is a 128 systolic array processors for high performance comparisons based on S&W algorithm. *BIOCCELERATOR* [8] is a reconfigurable system that operates based on the N&W algorithm and elucidated for optimal local alignments by S&W algorithm. *GeneMatcher2* [9] is a system with parallel architecture implemented in ASIC technology. It performs similarity search algorithms for both position-independent (S&W, N&W) and position-specific (HMM) optimal algorithms. BLAST is the heuristic (non-optimal) algorithm that GeneMatcher2 also performs. *Kestrel* [10] is a single-board programmable parallel processor with 512 Processing Elements (PEs), and the two primary target applications are the S&W and HMM algorithms. *BISP* [1]: the system is based on the S&W algorithm and is implemented in ASIC technology (more details in section 3).

1.3 The SWASAD

The SWASAD is designed to highlight the feasibility of incorporating sensitive searching algorithms like S&W into application-specific hardware systems. The SWASAD executes only the S&W searching algorithm, while other similar products can generally execute multiple algorithms. However, the cost of the SWASAD is significantly less than that of other products, both in terms of engineering design cost and actual product money cost, and the SWASAD design is retargetable, which highlights its potential to deal with the future growth in Bioinformatics. The clock speed of the SWASAD design is 50MHz. While the design is capable of being executed at many time more than this speed, getting the data out of the chip is far more difficult, as we had a PCI bus to interface to. The SWASAD chip incorporates a new distance approach which makes the output data smaller, and uses new design techniques to make it smaller and faster.

1.4 Paper Outline

The remaining sections of this paper detail the SWASAD project. Section 2 reviews the S&W similarity approach in the BISP and then introduces the distance approach applied in the SWASAD. Section 3 discusses in detail the implementation of the SWASAD, while section 4 examines the results and compares the SWASAD to other existing products. The overall conclusions are drawn in section 5.

2. Algorithms

The S&W similarity approach in the BISP states that the similarity of the two sequences can be determined by finding the accumulated weights (maximum score) introduced by the steps of the best alignment.

For the purposes of this project only the Deoxyribo-Nucleic Acid (DNA) sequences are analysed, and A, T, G and C are referred to as DNA bases which make up the sequences.

When *query sequence* $a = a_1a_2...a_{11} = ACAGGACTACA$ is compared to *database sequence* $b = b_1b_2...b_{11} = ACAGACTATCA$, the best alignment can be shown to be:

database sequence: *ACAG ΔACTATCA*
query sequence: *ACAGGACTAΔCA*

The symbol Δ stands for a gap, the insertion of which enables superior alignment.

In the similarity approach, the higher the maximum score, the more similar the two compared sequences. Alternatively, the relationship of two compared sequences can be viewed as 'how different they are' instead of 'how similar they are'. This is referred to as a 'minimum distance approach' in [11] (the biological equivalence for both approaches with a complete proof appears in [12]).

In order to return the relationship between two closely related sequences in terms of distance approach, scores that are close to zero are more significant than scores close to the maximum value of the data bus. Minimum distance theory is useful for improving the VLSI implementations of the S&W algorithm since reducing the data-width of the system actually minimizes the chip size and optimises the performance and power consumption. Figure 1 shows the pseudo-code for the S&W distance approach. In the given pseudo code, constants $u = 2$ and $v = 1$, and $(s_i\,y_j) = 3$ if bases are matched, -3 otherwise). N and M are lengths of the database and the query sequences.

The is initialised with $H_{0,0} = H_{i,0} = H_{0,j} = 0$, *for* $i \geq 0$ *and* $j \geq 0$ for the similarity approach. In the distance approach it is more complex. E and F are both intermediate scores.

```
-- Initialisations
for i = 1 to M do
   $E_{i,0} \Leftarrow u_d + v_d\,(i-1)$;
   $H_{i,0} \Leftarrow u_d + v_d\,(i-1)$;
end for;
for j = 1 to N do
   $F_{0,j} \Leftarrow u_q + v_q\,(j-1)$;
   $H_{0,j} \Leftarrow u_q + v_q\,(j-1)$;
end for;
-- Process
for i = 1 to M do
   for j = 1 to N do
      $E_{i,j} \Leftarrow \mathbf{min}\{H_{i,j-1} + u_d,\ E_{i,j-1} + v_d\}$
      $F_{i,j} \Leftarrow \mathbf{min}\{H_{i-1,j} + u_q,\ F_{i-1,j} + v_q\}$
      $H_{i,j} \Leftarrow \mathbf{min}\{H_{i-1,j-1} + (s_i\,y_j),\ E_{i,j},\ F_{i,j}\}$
   end for;
end for;
```

Figure 1: the pseudo-code of the S&W distance approach

With the same query and database sequence bases it can be shown that we get the same alignment under the distance approach as we got with the similarity approach. The trade-off between the advantage of reducing the bus-width (by using distance theory), and the disadvantage of the extra hardware logic necessary (some adders, registers and control signals) for the initialisations need to be considered. The reason why the distance approach is implemented in SWASAD will be explained in the next section.

3. Implementation

The design implementation is discussed in a top-down fashion from the chip architecture to the Processor Elements (PEs).

3.1 Chip Architecture

Figure 2 is the schematic diagram of the SWASAD chip. The control logic schedules the SWASAD chip's internal operations and responds to control signals from the microprocessor. The constant registers store SWASAD constants (like v and u). The status outputs reports the status of the chip to the microprocessor and then the microprocessor will respond to it. The clock

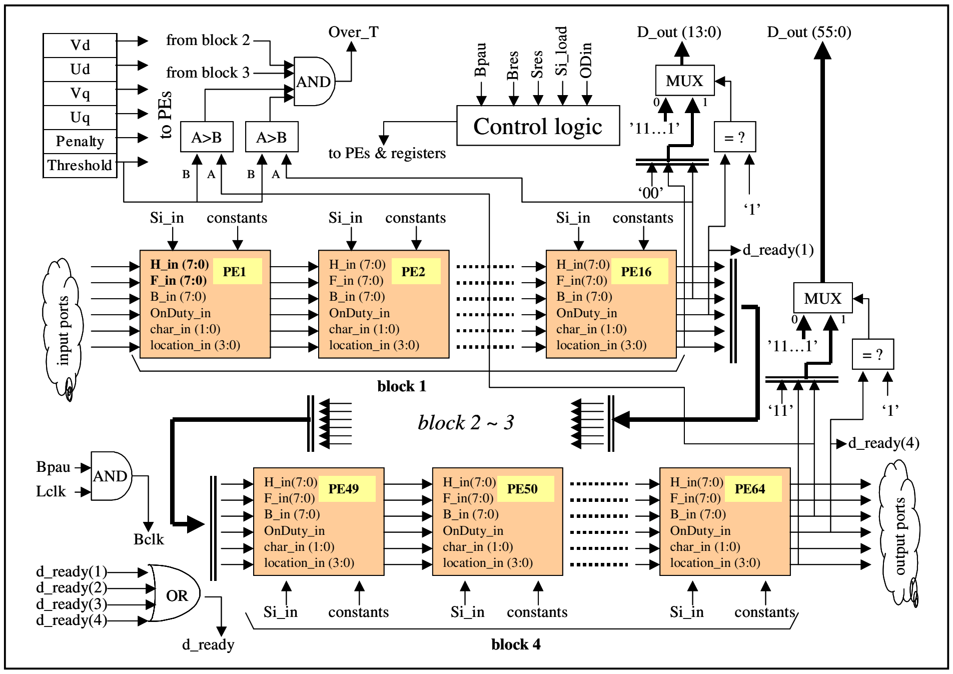

Figure2: Schematic diagram of the SWASAD chip

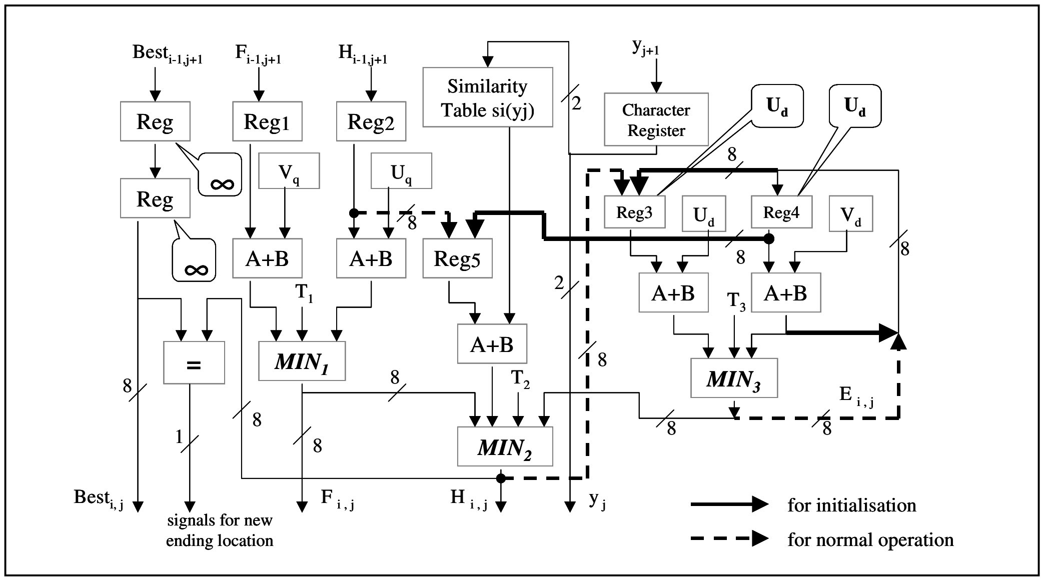

Figure 3: Architecture of the SWASAD processor element

control logic enables the microprocessor to turn the SWASAD chip clock on and off. The main modules are blocks of PEs, where each PE is serially connected to each other to form the systolic array processor.

3.2 Processor Elements

The PEs are the main modules of the SWASAD chip, and small modifications done to each PE will affect the overall chip greatly, in terms of chip size, power consumption and chip performance, since SWASAD chips are composed of a number of identical PEs.

The operational state of the SWASAD begins exactly one clock cycle after the initialisation finishes. All bold connections in the initialised state are disconnected while all the dotted connections are established and are ready for operations, as shown in the PE architecture in Figure 3. Each register indicated by the callout blocks needs to be pre-loaded to the value (such as ∞, U_d etc) within its corresponding callout block for initialisation purpose.

The rest of section 3 discusses techniques to improve PE by optimising its main components.

3.3 Initialisation Logic

The distance approach is chosen for this thesis, because a methodology to implement the initialisation hardware for the distance approach without inferring any extra adder or register hardware is given here. As shown in Figure 3, both initialised and operational states share common registers and adders by switching the wiring connections. Moreover, the chip performance is enhanced since the initialisations are done on the fly while the chip is calculating the scores.

3.4 Comparator

Each PE implies three comparators and each comparator takes up significant area and increases the delay of the critical path, therefore the comparator is an excellent candidate for optimisation. From the hardware point of view, a subtractor is generally more efficient than a comparator in both area and speed [13]. Figure 4 shows two architectures implemented in VHDL code (***RTL1*** and ***RTL2***) that perform the same task, i.e. compares the two inputs data1 and data2, and signals the result of the comparison to the output via output port *LE* [13]. ***RTL1*** implies a comparator (line 9) while ***RTL2*** implies a subtractor (line 19). From [13], ***RTL2*** infers hardware with less area and less delay than ***RTL1***.

```
1  Entity LE16 is
2    Port (
3       data1:in std_logic_vector (15 downto 0);
4       data2:in std_logic_vector (15 downto 0);
5       LE: out std_logic);
6  End LE64;
7  Architecture RTL1 of LE16 is
8  Begin
9    LE <= '1' when (data1 <= data2) else '0';
10 End RTL1;
11
12 Architecture RTL2 of LE16 is
13 Begin
14    Process (data1, data2)
15       Variable x, y, z: unsigned (16 downto 0);
16          Begin
17             x:= '0' & unsigned (data1);
18             y:= '0' & unsigned (data2);
19             z:= y – x;
20             If (z(16) = '0') then
21                LE<= '1';
22             Else
23                LE <= '0';
24             End if;
25    End process;
26 End RTL2;
```

Figure 4: VHDL codes of RTL1 and RTL2

The subtractor inferred in ***RTL2*** from above example is 17 bits while each of the input ports data1 and data2 is 16 bits only. The 17-bit-subtracter is needed, in order to signal whether the subtraction has under flowed or not (that returns the comparison result - line 20 ~ 24). From the nature of how the S&W algorithm scores are accumulated and compared, the additional MSB of the subtractor (for example, *z(16)*) can be eliminated for the purpose of reducing the size and delay of the PE comparison modules without altering the algorithm. Instead, *Z(15)* is used to signal the subtraction result in this case.

3.5 Overflow Logic

The worst case of the S&W distance approach is that when there is an array of hundreds of PEs while two non-related sequences are compared, the accumulated *H* score will eventually exceed the bus-width of the SWASAD adders. Therefore overflow will occur and incorrect scores will be reported. The associated overflow logic is implemented by introducing an extra input element for each comparator in PE, for example constant T_1 for $\boldsymbol{MIN_1}$ in Figure 3. This constant makes sure that largest possible score outputted from this comparator is always T_1. The value of T_1 is set in such a way to prevent the value outputted from the comparator to the next PE causing an overflow in the next PE and so on. Same terminology applies to the other two comparators $\boldsymbol{MIN_2}$ and $\boldsymbol{MIN_3}$.

The overflow logic is implemented with an equator instead of a comparator (since a comparator generally has worse hardware usage and delay). Assume system bus-width is 8 bits for example, and query sequence open gap and continuous gap penalties u_q and v_q are set to be 4 bits and 2 bits separately. Any scores larger than *'11111111'* for 8 bits adder will cause an overflow. From addition of *Reg1* in Figure 3 (coming from F score of the previous PE and v_q for the $F_{i,j}$with v_q *= 01, 10* or *11*). In general, for a system with bus-width of *M* bits and continuous gap penalty v_q of *N* bits, constant T_1 is set as '*111…10…00'* where there are *(M-N)* number of *'1'*s in the higher portion and *N* number of *'0'*s in the lower portion. Supposing 8-bit-register *Reg1* receives an incoming score of value T_1 that is constrained by the previous *F* score overflow logic. After adding *Reg1* to v_q (2 bits), the result exceeds T_1 which will cause an overflow in the next PE and therefore the result of $F_{i,j}$ is constrained to be T_1. This event is detected by a 6 bits equator instead of system bus-width of 8 bits and saves additional hardware.

4 Results

The front-end HDL design for SWASAD is implemented using behavioural VHDL, and is simulated using the ActiveHDL simulator [14], and ModelSim simulator [15] for verifying functionalities and timings. Then the VHDL is synthesized into netlist files using the Leonardo Spectrum synthesizer [16]. Chip area, chip performance and schematic layout of the synthesized design are all estimated using Leonardo. The IC layout is automatically generated, and standard-cells placed and routed, using the layout-editing tool ICstation [17] from Mentor Graphics. The design rule check and layout versus schematics check were then performed. Finally pad frames were created and the layout is manually edited to connect the I/O ports of the design to the pad cells. After all procedures mentioned above were completed, the design was ready to be fabricated. Figure 11 shows the layout of the SWASAD die with 32 PEs and 8-bit system bus.

Table 1 shows the overall comparisons between specifications and performances of the SWASAD and the BISP chips. The improvements achieved by the SWASAD over the BISP are printed in **bold**, and the disadvantages of the SWASAD are printed in *italics*. If it is assumed that the BISP chip can operate up to 50 MHz (with help of better fabrication process etc.) just like the SWASAD chip, it improves the BISP performance up to 800 millions MCPS, which is still ¼ of the SWASAD performance.

SPECIFICATIONS		
	SWASAD	**BISP**
Max no. of base sets allowed	4	128
Score overflow/underflow	only overflow needed	both
Outgoing dataframe size	14 bits	64 bits
No. of PEs per chip	64	16
No. of I/O pins per chip	145 or 161	208
Fabrication process	AMI 0.5 um	HP 1.0 um
Chip actual area	50mm^2	110mm^2
Chip gate counts	75,847	100,000
Transistor counts	303,388	400,000
Datapath	8 bits	16 bits
PERFORMANCE		
Clock speed	50 MHz	12.5 MHz
Rate of MCPS	3.2 x10^9	0.2x10^9

Table 1: Comparisons for the SWASAD and the BISP

The hardware implementations of the biological sequence comparisons stated in this thesis are compared in Table 2. The SWASAD is an improved version of the BISP, and its specifications are similar to the Kestrel design, which is produced under the same fabrication process (0.5 μm) with a similar die size. Although Kestrel performs multiple dynamic algorithms while the SWASAD is only S&W algorithm-specific, the performance of the SWASAD (MCPS per chip) is calculated to be better than that of Kestrel. The SWASAD has a faster clock speed than Kestrel, while both have the same number of PEs within a chip. The design cost of SWASAD is 13 person-months.

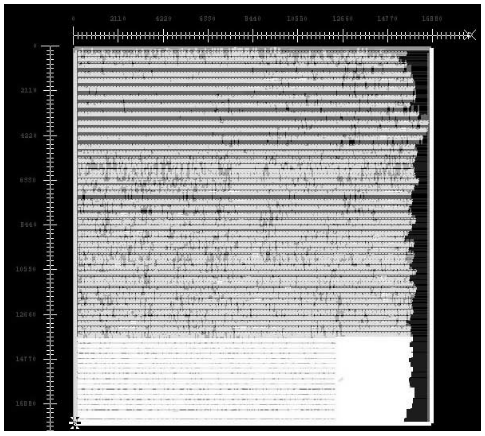

Figure 5: The SWASAD die layout (32-PE, 8-bit)

5 Conclusion

As science progresses further into genomics, there is a pressing need for researchers to be able to swiftly convert an ever-expanding volume of data into meaningful and useful information via sequence analysis. The purpose of this thesis has been to design an application-specific hardware tool, the SWASAD, to satisfy the high computational load requirement of sequence analysis.

The benefits gained from improving the performance of the SWASAD chip in relation to the BISP chip are a reduction of the databus bandwidth (achieved by applying the S&W distance approach), a restructuring of the PE architecture, and the use of more advanced fabrication technology to develop the chip.

The SWASAD chip with 64 PEs is estimated to have a size of 7.11 mm by 7.11 mm in a 0.5 μm process (this could be further reduced by using a hierarchical design methodology – at present we use a flat structure). Future technology of 0.1 μm process will make possible a SWASAD that has 1024 PEs, a controller core and a storage element all within a single chip. The essential tasks in this implementation are predicted to be the intercommunication between modules, the architecture of the controller, the format of the data frame, and operational instructions and layout of the storage element.

References

[1] E. Chow, T. Hunkapiller, J. Peterson, and M.S. Waterman, "Biological Information Signal Processor," in *Proceedings of the International Conference on Application-specific Array Processors* (Mateo Valero, Sun-Yuan Kung, Tomas Lang and Jose A.B. Fortes, Eds.). Los Alamitos, California: IEEE Computer Society Press, 1991, pp. 144-160.
[2] UK Human Genome Mapping Project Resource Centre, "How Does BLAST Work," [Online], Hinxton Genome Campus, Available: http://www.hgmp.mrc.ac.uk/MANUAL/faq/faq-dbsearch.html#blast_results
[3] CAPSL in the University of Delaware, "FASTA (Pearson and Lipman, 1988)," [PDF document], Available: http://www.capsl.udel.edu/courses/eleg667/2000/SLIDES/Topic2b_files/sld006.htm
[4] Washington University, "HMMER 2.1.1: Profile Hidden Markov Models for Biological Sequence Analysis," [Online], Available: http://hmmer.wustl.edu/
[5] Time Logic Inc., "Decypher II Product Literature," [Online], Incline Village, NV, 1996, Available: http://www.timelogic.com
[6] P. Guerdoux-Jamet, and D. Lavenier, "SAMBA: Hardware Accelerator for Biological Sequence Comparison," *CABIOS*, vol. 13, pp. 609-615, Dec., 1997.
[7] IRISA, "Systolic Accelerator for Molecular Biological Applications (SAMBA)," [Online], Cedex France, Available: http://www.irisa.fr/SAMBA/
[8] EMBL, "BIOCCELERATOR Product Information," [Online], Available: http://shag.embl-heidelberg.de:8000/Bic/docs/bicINFO.html
[9] Paracel Inc., "FDF-3 Product Information," [Online], Pasadena, CA, 1996, Available: http://www.paracel.com
[10] UCSC Bioinformatics (Computational Biology), "The UCSC Kestrel Parallel Processor," [Online], Santa Cruz, CA, 1994. http://www.cse.ucsc.edu/research/kestrel/index.html
[11] M.S. Waterman, "General Methods of Sequence Comparison," *Bull. Math. Biol.*, 46, pp. 473-500, 1984.
[12] T. F. Smith, M. S. Waterman, and W. M. Fitch, "Comparative Biosequence Metrics," *J. Mol. Evol.*, 18, pp. 38-46, 1981.
[13] K. C. Chang, *Digital Design and Modeling with VHDL and Synthesis*. Los Alamitos, California: IEEE Computer Society Press, 1997.
[14] Aldec Inc., "Active-HDL Product Information," [Online], Henderson, NV, 1997, Available: http://www.aldec.com/ActiveHDL/default.htm
[15] Model Technology Inc., "ModelSim," [Online], Portland, OR, 1990, Available: http://www.model.com/products/overview_datasheet.asp
[16] Exemplar Inc., "LeonardoSpectrum," [Online], San Jose, CA, 1987, Available: http://www.exemplar.com/products/leonardospectrum.html
[17] Mentor Graphics Inc., "IC station," [Online], Wilsonville, OR, 1981, Available: http://www.mentor.com/cicd/icstation.html

	DeCypher	SAMBA	BIOCELLERATOR	GeneMatcher2	Kestrel	SWASAD	BISP
chip hardware	FPGA	ASIC	FPGA	ASIC	ASIC	ASIC	ASIC
heuristic or dynamic algorithms	both	dynamic	dynamic	both	dynamic	dynamic (S&W only)	dynamic (S&W only)
no. of PEs per chip	n/a	4	4	192	64	64	16
no. of chips per board	8	32	4	16	8	4	16
chip clock speed (MHz)	10 ~ 120	n/a	20	n/a	20	50	12.5
S&W million MCPS per chip	312.5	40	62.5	~1916.7	1280[1]	**3200**	200
fabrication process (µm)	n/a	1.2	N/a	n/a	0.5	0.5	1
die size (mm x mm)	n/a	n/a	N/a	n/a	7.2 x 8.3	~ 7.1 x 7.1	10 x 11
power consumption (watt)	n/a	n/a	200[2]	2200[3]	n/a	n/a	n/a
cost	n/a	n/a	n/a	US $360k	US$15k~25k	n/a	n/a

Table2: Comparisons between various hardware implementations

[1] there are 64 PEs per chip that runs at 20 MHz, the maximum MCPS is calculated as: 64 x 20 = 1280 MCPS
[2] 90-135V or 190-265V
[3] optimised configuration version (9 boards), 2200w at 220V and 10A