White paper
Fujitsu SPARC64™ X+/X Software on Chip Overview for Developers

Table of Contents

Table of Contents

Software on Chip Innovative Technology

1 Software on Chip Innovative Technology

Fujitsu brings together innovations in supercomputing, business computing, and mainframe computing in the Fujitsu M10 enterprise server family to help organizations meet their business challenges. Technologies that were once reserved for data-intensive scientific computing have now become especially relevant to mission-critical business computing.

Fujitsu SPARC64™ X+/SPARC64™ X processors found in Fujitsu M10 servers introduce the innovative "Software on Chip" technology to achieve significant performance improvements by implementing functions previously performed by software into the CPU hardware.

Fujitsu SPARC64™ X+/X processors implement Software on Chip technology as an instruction set extension to the standard SPARC V9 microprocessor specification in the following areas:

- SIMD Single Instruction Multiple Data Vector Processing
- Extended Floating-Point Registers
- Decimal Floating-Point Processing
- Cryptographic Processing

SPARC64™ X+/X SIMD Vector Processing

2 SPARC64™ X+/X SIMD Vector Processing

SIMD (Single Instruction Multiple Data) is a technology that allows processing of multiple data in a single instruction. This results in a reduction of instructions and significant acceleration of execution.
With conventional serial (non-SIMD) processing, a single data element is processed at each instruction. Fujitsu SPARC64™ X+/X SIMD instructions are implemented as an instruction set extension to the standard SPARC V9 microprocessor specification. With SIMD, SPARC64™ X+/X processors can process as many as 16 data elements at each instruction.

2.1 How Oracle Databases take advantage of SPARC64™ X+/X SIMD vector processing

Business intelligence and analytics workloads often involve processing of an entire column of the database table; executing the same instruction on each item in the database table column of interest. SIMD vector processing can dramatically accelerate the analysis, provided that the database column is made available in the system memory and also placed in the columnar order.

In traditional SPARC servers, such as the SPARC Enterprise family (the predecessor of Fujitsu M10), only one data entry could be processed at each instruction, and the processor could execute two integer instructions at the same time. Therefore, two data entries could be processed in each cycle. In contrast, the SIMD instruction in SPARC64 X can process up to eight data entries per instruction, and the processor executes one SIMD instruction per cycle. SIMD is further enhanced in SPARC64 X+ and with a future release of Oracle Database will be able to process up to 16 data entries per instruction when the size of the data is one byte. Also, two of these SIMD instructions can be executed at the same time. This means that up to 32 data entries can be processed in each cycle. Oracle Database 12c currently supports single ALU (Arithmetic-Logic Unit) execution for SIMD compare instructions. Each ALU in SPARC64 X+ processors can process up to 8 data entries, and with all four ALUs up to 32 data entries can be processed in parallel (Figure 1).

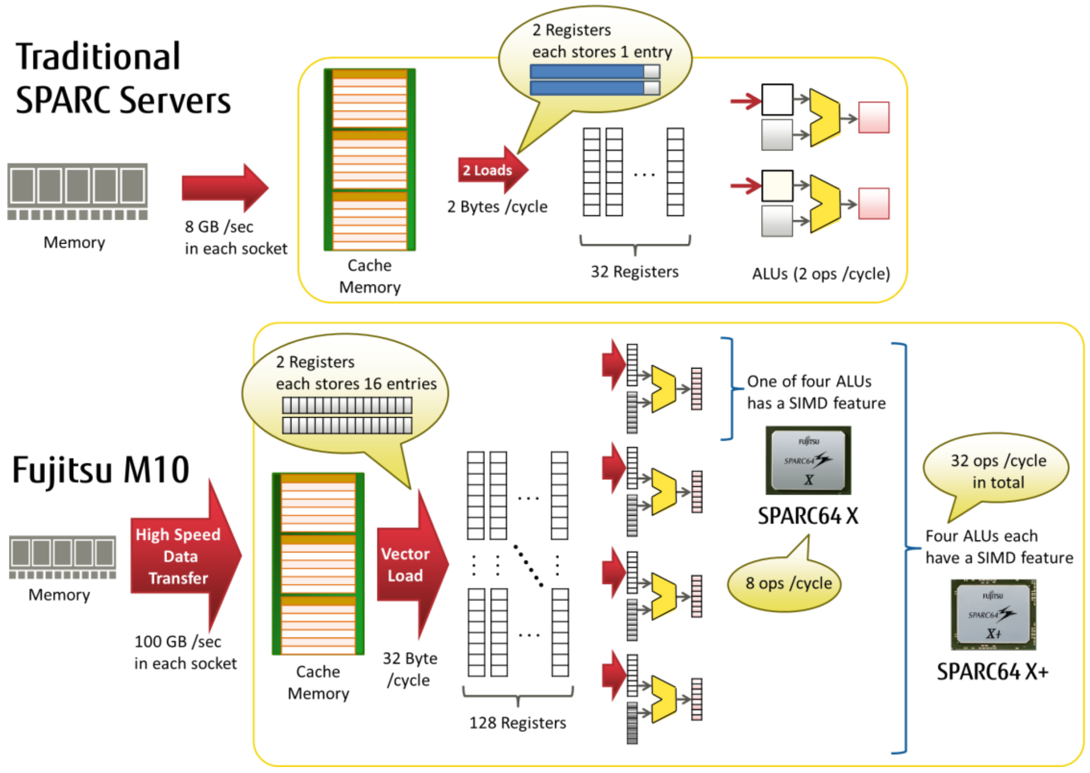

Figure 1 SIMD Vector Processing in Traditional SPARC Servers and in Fujitsu M10

The SIMD capability accelerates searching large amounts of data, compressing/decompressing data, in-memory databases, and many other workloads.

The Oracle Database 12c In-Memory option represents database tables in memory using traditional row format as well as the new in-memory columnar format. Oracle Database In-Memory takes full advantage of SPARC64™ X+/X architecture with SIMD instructions, extended registers, and the large memory capacity of Fujitsu M10 servers. For analytic queries, in-memory column format is used to load multiple database column items onto SPARC64™ X+/X CPUs, and all are processed in a single CPU instruction. Database scan, join, and aggregation operations extensively benefit from the SIMD vector processing capabilities to enable real-time business decisions.

The minimum requirements for SPARC64™ X+/X SIMD vector processing support in Oracle Database are:
- Oracle Database 12c Enterprise Edition (12.1.0.2 or later)
- Oracle Database In-Memory Option
- Oracle Solaris 11.1 or later

2.2 How to use of SPARC64™ X+/X SIMD instructions in user applications

Minimum requirements are:
- Oracle Solaris Studio Compiler 12.4 or later
- Oracle Solaris 11.2 SRU4 or later

  Note: Studio 12.4 does not require Solaris 11.2 SRU4 or later to compile. Compiling on Solaris 10/11.1 is supported. An application compiled by Studio 12.4 requires a Solaris 11.2 SRU4 or later execution environment.

Use one of the following compiler options:
- `-xtarget=sparc64x` for SPARC64™ X processors
- `-xtarget=sparc64xplus` for SPARC64™ X+ processors
- `-xtarget=native` when compiling applications and building the executables on a Fujitsu M10 server.

For these processor target architecture specifications, `-xvector=simd` is always effective with the specification of any `-xvector` option, except `-xvector=%none` and `-xvector=no%simd`. Basically, with the explicit `-xvector=simd` option, the compiler will perform loop transformation, enabling the generation of special vector-ized SIMD instructions to reduce the number of loop iterations. In addition, `-O` greater than `3` is required for `-xvector=simd`, otherwise it is skipped without any warning.
Usually a simple loop (like a `'for'` statement in C/C++ or `'DO'` loop in FORTRAN) can be accelerated with SIMD instructions.

In addition, the `memcpy()` and `memset()` functions in Oracle Solaris 11 are optimized with SPARC64™ X+/X SIMD instructions. Therefore, user applications can automatically receive the benefits when calling `memcpy()`/`memset()`.

2.3 How to check if the system and operating system is capable of SIMD execution?

Use the `isainfo` command to determine whether the Fujitsu M10 system has SIMD capabilities enabled.

```
PROMPT> isainfo -v
64-bit sparcv9 applications
       fjorclnum fjieeedec cbcond pause fjaes ima fjdes fjathhpc fmaf asi_blk_init
       vis2 vis popc vis3b fjathplus
32-bit sparc applications
       fjorclnum fjieeedec cbcond pause fjaes ima fjdes fjathhpc fmaf asi_blk_init
       vis2 vis popc v8plus div32 mul32 vis3b fjathplus
```

`fjathplus or fjathhpc` indicates that this particular Fujitsu M10 system is capable of using the SIMD instructions in SPARC64™ X+ or SPARC64™ X processors. Minimum requirements are Oracle Solaris 11.2 SRU4 or later on SPARC64 X or X+ processors.

2.4 How to check if SPARC64™ X+/X SIMD instructions indeed have been generated upon compilation of user source code?

The developer can check if the compiler environment and the selected compiler options indeed generate SIMD instructions in the target processor architecture.

The developer can use the`-S` compiler option to produce the assembly source file:

```
PROMPT> cc -S -xtarget=native hello.c
```

The `-S` option here causes a new `hello.s` file to be created that contains the assembly version of the C application. The file `hello.s` can be inspected for instructions that have '`,s`' suffix referring to the use of SIMD.

Example of non-SIMD (scalar) instruction

```
/* 0x0144          */       fmaddd  %f0,%f48,%f50,%f58
```

Example of SIMD (vector) instruction

```
/* 0x0220          */       fmaddd,s      %f32,%f38,%f42,%f56
```

2.5 Is SPARC64™ X+/X SIMD implementation compatible with Oracle's SPARC SIMD?

Fujitsu SPARC64™ X+/X implements SIMD vector processing capabilities as an instruction set extension to the standard SPARC V9 specification. Some instructions are implemented commonly across different SPARC V9 extensions, such as VIS3 – Visual Instruction Set, and some are not. It is possible that some applications that take advantage of SIMD implementation on other SPARC systems can indeed run on the Fujitsu M10 servers, and vice versa.

Example Compilation without SPARC64 X+ SIMD

```
PROMPT> cc -fast -xtarget=sparc64xplus -xvector=no%simd -o example1_nosimd example1_main.c example1_foo.c
PROMPT> time ./example1_nosimd ; time ./example1_nosimd ; time ./example1_nosimd

ANS = 3.90624e+13
real    0m2.754s
user    0m2.530s
sys     0m0.218s

ANS = 3.90624e+13
real    0m2.741s
user    0m2.515s
sys     0m0.218s

ANS = 3.90624e+13
real    0m2.732s
user    0m2.509s
sys     0m0.216s
```

Example Compilation with SPARC64 X+ SIMD

```
PROMPT> cc -fast -xtarget=sparc64xplus -o example1 example1_main.c example1_foo.c –
PROMPT> time ./example1 ; time ./example1 ; time ./example1

ANS = 3.90624e+13
real    0m1.742s
user    0m1.517s
sys     0m0.218s

ANS = 3.90624e+13
real    0m1.741s
user    0m1.517s
sys     0m0.217s

ANS = 3.90624e+13
real    0m1.740s
user    0m1.517s
sys     0m0.216s
```

example1_main.c

```
#include <stdio.h>

#define N 10000000
double A[N], B[N], C[N], D[N];

void foo(double* a, double* b, double* c, double* d, int n);

int main()
{
  for (int i = 0; i < N; ++i) {
    A[i] = (double)i / 16.0;
    B[i] = (double)i / 16.0;
    C[i] = (double)i / 16.0;
    D[i] = (double)i / 16.0;
  }

  for (int i = 0; i < 100; ++i)
    foo(A, B, C, D, N);
  printf("ANS = %g\n", D[N-1]);
  return 0;
}
```

example1_foo.c

```
void foo(double* a, double* b, double* c, double* restrict d, int n)
{
  for (int i = 0; i < n; ++i) {
    d[i] += a[i] * b[i] - c[i];
  }
```

SPARC64™ X+/X Decimal Floating Point Processing

3 SPARC64™ X+/X Decimal Floating Point Processing

Fujitsu SPARC64™ X+/X processors support Decimal Floating-Point numeric calculations following the IEEE754 standard in DPD (Densely Packed Decimal) and Oracle NUMBER formats. Arithmetic processing of the decimal number data is performed directly in the processor hardware, without conversion back and forth to binary by software. SPARC64™ X+/X processors inherit this high-speed computing technique from Fujitsu mainframe computers.

3.1 How Oracle Databases take advantage of SPARC64™ X+/X Decimal Floating-Point

Oracle Database 12c takes advantage of the Decimal Floating-Point capabilities of the SPARC64™ X+/X processors for Oracle NUMBER and floating-point representation directly out-of-the-box. The database performance benefits are realized across many industries such as retail, manufacturing, and finance by significantly accelerating common processing tasks like sales accounting, accounting, rebate accounting, and compound interest calculations.

Minimum Requirements:

- Oracle Database 12cR1 (12.1.0.1 or later) + patch , or Oracle Database 12cR1 Enterprise Edition (12.1.0.2 or later)
- Oracle Solaris 11.1 or later

3.2 Decimal Floating-Point processing in user applications

The Oracle Solaris Studio Compiler provides the various functions needed to handle data in decimal format.
Minimum requirements are:

- Oracle Solaris Studio Compiler 12.4 or later
  - `-xtarget=sparc64x, -xtarget=sparc64xplus or -xtarget=native` is needed.
  - `-m64` is also needed
- Oracle Solaris 11.2 SRU4 or later

Note: Studio 12.4 does not require Solaris 11.2 SRU4 or later to compile. Compiling on Solaris 10/11.1 is supported. An application compiled by Studio 12.4 requires a Solaris 11.2 SRU4 or later execution environment.

By using the appropriate intrinsic functions and data types in the source code, applications can take advantage of the Decimal Floating Point representation and achieve performance improvements.

To handle and operate on Decimal Floating-Point type variables, the compiler defined intrinsic functions should be employed. One cannot simply write `c = a + b;`. Literal syntax is not supported.

Example

If the addition of two decimal numbers is needed, the corresponding application would be written as:

```
#include <dpd_conf.h>
int main(void) {
  _Decimal64 a, b, c;

…
  c = __dpd64_add(a, b);
…
  return 0;
}
```

To represent Decimal Floating-Point numbers, `_Decimal64` intrinsic type is declared in `dpd_conf.h`. The intrinsic functions are also declared in `dpd_conf.h`. They are used to operate on `_Decimal64` type variables. This header file should be included in the source prior to use of the type declarations as in the above example.
For a complete list of intrinsic functions for Decimal Floating-Point type, please refer to Oracle Solaris Studio Compiler documentation.

SPARC64™ X+/X Extended Floating-Point Registers

4 SPARC64™ X+/X Extended Floating-Point Registers

The number of floating-point registers in the SPARC V9 architecture standard is specified as 32. Since this may not be sufficient for parallel arithmetic processing at high speeds, portions of the register need to be saved temporarily on the cache and then restored to the register, introducing inefficiencies in the execution.
SPARC64™ X+/X processors have four times the number of floating-point registers; extended from 32 to 128. This allows the wait-time of instructions to be reduced with less need to save and restore the data to the cache. The result is higher processing speeds and higher levels of parallelism.

4.1 How Oracle Databases take advantage of SPARC64™ X+/X Decimal Floating-Point

Fujitsu SPARC64™ X+/X SIMD instructions readily use the extended floating-point registers. Oracle Database and any user application which contain SPARC64™ X+/X SIMD instructions directly benefit from Extended Floating-Point Registers. No further user intervention is needed.

SPARC64™ X+/X On-Chip Cryptographic Processing Capabilities

5 SPARC64™ X+/X On-Chip Cryptographic Processing Capabilities

Fujitsu SPARC64™ X+/X processors implement two cryptographic arithmetic units per compute core, shared by each core's two threads. The dedicated on-chip cryptographic unit executes encryption/decryption instructions directly on the hardware for high speed cryptographic processing (encryption/decryption) without external adaptors or complex software. Applications and transactions achieve higher levels of security without performance overhead, additional costs, or complexity.

5.1 How to use the On-Chip Cryptographic Processing Capabilities

SPARC64™ X+/X processors support AES, DES, 3DES, RSA, SHA and DSA encryption models. OpenSSL and encrypt/decrypt commands are already compliant with the SPARC64™ X+/X cryptographic arithmetic processing functions. Standard libraries such as libpkcs11 also benefit from the cryptographic arithmetic processing enhancements.

On-chip cryptographic processing capability of SPARC64™ X+/X processors is supported by the Oracle Solaris Cryptographic Framework as well as by Oracle Database 11g and 12c and Oracle Advanced Security Transparent Data Encryption (TDE). The SPARC64™ X+/X processors can improve data security with full database encryption.

Minimum requirements for cryptographic processing are:

- Oracle Database 11gR2 Enterprise Edition + patch or Oracle Database 12cR1 Enterprise Edition
- Oracle Advanced Security Transparent Data Encryption (TDE) for AES and 3DES only
- Oracle Solaris 11.1 or later (for AES, DES, 3DES and SHA)
- Oracle Solaris 11.2 or later (for RSA and DSA)

For more information, please see: Developer's Guide to Oracle Solaris 11 Security and Oracle Advanced Security Transparent Data Encryption (TDE).

5.2 How to use the On-Chip Cryptographic Processing Capabilities in user applications

On-Chip Cryptographic processing capabilities can be directly used in user applications by calling the Oracle Solaris cryptographic framework.

Minimum Requirement for AES, DES, DES3, SHA:

- Oracle Solaris 11.1 or later

Minimum Requirement for RSA, DSA:

- Oracle Solaris 11.2 or later

There are two major frameworks(PKCS11 or OpenSSL) for implementing encryption related applications in C/C++.

Example for PKCS11 I/F:

```
#include <security/cryptoki.h>
#include <security/pkcs11.h>
…
  a = C_EncryptInit(Session, mechanism, Key)__;
…
```

For more information, including examples, please see: Developer's Guide to Oracle Solaris 11 Security

Example for OpenSSL I/F:

If the user application calls OpenSSL I/F cryptographic, set pkcs11 as a ENGINE at the initialization of the program.

```
    ENGINE              *e;
    const char          *engine_id = "pkcs11";

    e = ENGINE_by_id(engine_id);
    ENGINE_init(e);
    ENGINE_set_default_RSA(e);
    ENGINE_set_default_DSA(e);
    ENGINE_set_default_ciphers(e);
```

As for Java, JDK 7 or later supports the SPARC64™ X+/X On-Chip cryptographic framework. To use the On-Chip cryptographic engine in a Java application, set the `OracleUcrypto` as a provider.

Example for Java:

```
    SecretKey secretKey = KeyGenerator.getInstance("AES").generateKey();
    Cipher ci = Cipher.getInstance("AES/ECB/PKCS5Padding", "OracleUcrypto");
    ci.init(Cipher.ENCRYPT_MODE, secretKey);
    cryptedData = ci.doFinal(plainData);
```

For more information, please see: Java Cryptography Architecture (JCA) Reference Guide and The `OracleUcrypto` Provider.

Conclusions

6 Conclusions

Software on Chip features are designed to accelerate specific workloads involving data cryptographic processing, decimal number execution, and large scale data processing. Software on Chip functionality has been integrated with Oracle Solaris 11 and Oracle Database 12c, allowing users to easily take full advantage of this exciting new technology.

Developers can have their applications take advantage of these innovations through the familiar Oracle Solaris Studio Compilers and Oracle Solaris Operating System facilities.

About Fujitsu

Fujitsu is the leading Japanese information and communication technology (ICT) company offering a full range of technology products, solutions and services. Approximately 159,000 Fujitsu people support customers in more than 100 countries. We use our experience and the power of ICT to shape the future of society with our customers. Fujitsu Limited (TSE:6702) reported consolidated revenues of 4.8 trillion yen (US$40 billion) for the fiscal year ended March 31, 2015. For more information, please see http://www.fujitsu.com.

Contact
FUJITSU LIMITED
Website: www.fujitsu.com
2015-07-31 WW-EN