

Optimized GDD7000 library (ECC LINPACK) benchmark.

The functions tested here were `SGEFA()` and `SGESL()`, single precision with interruptible BLAS1 function calls. The other parameters of the benchmark are

CPU TMS320C6711
CPU clock rate 150MHz
L2 cache size 32KB, 64KB
External memory 256MB
Endian Little
Code interruptibility Interruptible, but no interrupts occurred during the benchmark
Code relocation CE3 external SDRAM
Data relocation CE2 external SDRAM

See Appendix A. For the benchmark code.

Table 1. Benchmark data for the L2 cache size 32KB

Matrix size N	CYCLE COUNT	TIME(mS)	MFLOPS
8	6277	0.0419	11.2143
16	22474	0.1498	21.6423
32	98631	0.6575	36.3373
64	573688	3.8246	47.8364
75	846050	5.6403	51.8586
90	1401699	9.3447	53.7419
96	1780707	11.8714	51.2372
100	2209908	14.7327	46.6083
128	6987117	46.5808	30.7180
130	6097749	40.6517	36.8611
140	7413450	49.4230	37.8070
150	9323937	62.1596	36.9211
160	11742578	78.2839	35.5356
170	14351532	95.6769	34.8374
180	16553856	110.3590	35.8176
190	19464150	129.7610	35.7956
200	24002632	160.0175	33.8296
300	77098264	513.9885	35.3704
500	334571200	2,230.4746	37.5854
800	1335016576	8,900.1104	38.4954
900	1915754752	12,771.6982	38.1797

Table 2. Benchmark data for the L2 cache size 64KB

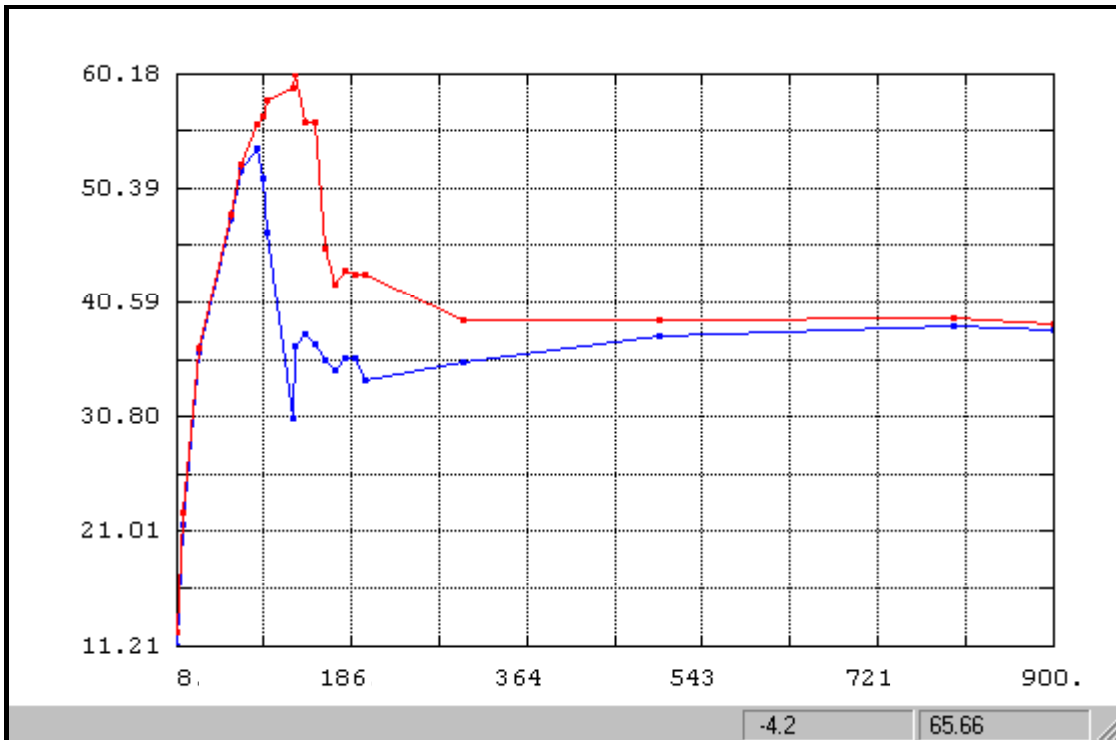
Matrix size N	CYCLE COUNT	TIME(ms)	MFLOPS
8	5707	0.0381	12.3344
16	21586	0.1439	22.5327
32	97618	0.6508	36.7145
64	571446	3.8096	48.0241
75	837549	5.5837	52.3850
90	1348625	8.9908	55.8569
96	1616682	10.7779	56.4356
100	1781809	11.8787	57.8064
128	3645589	24.3039	58.8740
130	3734971	24.8998	60.1798
140	5003570	33.3571	56.0160
150	5003570	33.3571	56.0160
160	9244879	61.6325	45.1363
170	11850102	79.0007	42.1912
180	13678445	91.1896	43.3470
190	13775896	91.8393	43.0404
200	13775896	91.8393	43.0404
300	69832272	465.5485	39.0507
500	69832272	465.5485	39.0507
800	1313415168	8,756.1006	39.1285
900	1892158464	12,614.3896	38.6559

The picture below shows performance in MFLOPS vs. matrix size N for two L2 cache sizes.

The blue graph shows performance for 32KB L2 cache size. Peak performance is achieved for N = 90.

The red graph shows performance for 64KB L2 cache size. Peak performance is achieved for N = 130.

Figure 1. Performance in MFLOPS vs. matrix size N.



Peak performances are in good correspondence with L2 cache size: $90 \times 90 \times 4 = 32400$ Byte, which is a little less than 32K (32768 Byte) and $130 \times 130 \times 4 = 67600$, which is a little above of 64K (65536 Byte).

Appendix A. Benchmark code

```
/*
Copyright (C) 1996-2005 by Andrew V. Nesterov. All rights reserved.
Copyright (C) 1996-2005 by GENERIC DIGITAL DESIGN INC. All rights
reserved.
Author: Andrew V. Nesterov

Created: 07/01/1999
Modified: 09/11/2005

ECCLINPACK. Extended C Callable LINPACK Library
TMS320C6000 DSP microprocessors. Part No GDD7000

This SOFTWARE is the module of LINPACK COMPUTATIONAL BENCHMARK.

If the SOFTWARE is being Licensed by the FEDERAL GOVERNMENT it is
provided with Restricted Rights. Use, duplication, or disclosure by
the government is subject to restrictions set forth in 48 CFR
52.227-19(c) (1) (2) or DOD FAR supplement 252.227-7013(c) (1) (ii)
as amended through the date hereof.
*/

#include <stdio.h>
#include <stdlib.h>
#include <gdd7000.h>

#include <real.h>
#include <time.h>

void matgen (int n, real **A);

#define ITS 10 // number of test iterations
#define CLOCK 150000000 // CPU clock rate cycles/sec
#define SQR(x) ((x)*(x)) // x**2
#define QUBE(x) ((x)*SQR(x)) // x**3

int main (void)
{
    int i, n, ifail, iter;
    int t0, tx, toverhead;
    int *ipvt;
    real **A;
    real *x, *xexact;
    real ts, taverage, tms, mflops;

    // BEGIN TEST
    puts ("-----");
    puts ("LINPACK N X N BENCHMARK");

    // SET MATRIX DIMENSIONS
    n = 140;

    // ALLOCATE MATRICES AND VECTORS

    ifail = 0;

    A = ssquare (n);
    if (A == NULL) ifail = 1;
}
```

```

    ipvt = (int *) malloc (n*sizeof(int));
    if (ipvt == NULL) ifail = 1;

    xexact = svector (n);
    if (xexact == NULL) ifail = 1;
    x = svector (n);
    if (x == NULL) ifail = 1;

    if (ifail == 1)
    {
        puts ("ALLOCATION FAILED, BENCHMARK TERMINATED");

        if (x != NULL) free (x);
        if (xexact != NULL) free (xexact);
        if (ipvt != NULL) free (ipvt);
        if (A != NULL) free (A);

        exit (-1);
    }

// TIMER OVERHEAD
t0 = clock ();
toverhead = clock () - t0;
ts = 0.0E0;
printf ("\n");

for (iter = 0; iter < ITS; iter++)
{
    // GENERATE TEST MATRIX
    matgen (n, A);

    // GENERATE EXACT SOLUTION AND RHS
    sfill (n, CNST(1.0E0), xexact, 1);

    for (i = 0; i < n; i++)
    {
        *(x+i) = sdot (n, row(A,i), 1, xexact, 1);
    }

    t0 = clock (); // START BENCHMARK

    SGEFA (n, A, ipvt, &ifail); // LU DECOMPOSITION
    SGESL (n, A, ipvt, x, 0); // SOLVE A*X=B

    tx = clock () - t0 - toverhead; // END BENCHMARK
    ts = ts + (real)tx;

    printf ("BENCHMARK %3d IFAIL %2d ", iter, ifail);
    printf ("COMPLETED IN %9d CYCLES ", tx);
    printf ("%9.4f MILLISECONDS", (real)tx/(real)CLOCK*1000.0);
    printf ("\n");
}

printf ("\n");
puts ("BENCHMARK COMPLETED");

taverage = ts/(real)ITS;
tms = taverage/(real)CLOCK*1000.0;

```

```

mflops = 2.0*(QUBE((real)(n))/3.0 + SQR((real)(n)))/tms/1000.0;

printf ("BENCHMARK AVERAGE COUNT   %9d CYCLES", (int)taverage);
printf ("\n");
printf ("BENCHMARK AVERAGE TIME     %9.4f MILLISECONDS", tms);
printf ("\n");
printf ("BENCHMARK AVERAGE MFLOPS    %9.4f", mflops);
printf ("\n");

// FREE MATRICES AND VECTORS

free (x);
free (xexact);
free (ipvt);
free (A);

puts ("PROGRAM TERMINATED");
puts ("-----");
return (0);
}

//  MATRIX GENERATOR

void matgen (int n, real **A)
{
    int i, k;

    for (i = 0; i < n; i++)
    {
        for (k = 0; k < n; k++)
        {
            value(A,i,k) = (real)rand()/(real)(RAND_MAX);
        }

        value(A,i,i) = CNST(1.0E0) + value(A,i,i);
    }

    return;
}

```



Kane Computing Ltd
7 Theatre Court, London Road,
Northwich, Cheshire, CW9 5HB, UK.
Tel: +44(0)1606 351006
Fax: +44(0)1606 351007/8
Email: sales@kanecomputing.com
Web: www.kanecomputing.co.uk