Exercise-3

Serial code is available at (`./trainingIPT/exercises/exercises_3_to_7/circuit.c`):

The complete code for this exercise can also be obtained from:
http://people.sc.fsu.edu/~jburkardt/c_src/satisfy/satisfy.html

Snippet of the serial code:

```
int main ( int argc, char *argv[] )
{
# define N 23

  int bvec[N];
  int i;
  int ihi;
  int j;
  int n = N;
  int solution_num;
  int value;

  printf ( "\n" );
  timestamp ( );
  printf ( "\n" );
  printf ( "SATISFY\n" );
  printf ( "  C version\n" );
  printf ( "  We have a logical function of N logical arguments.\n" );
  printf ( "  We do an exhaustive search of all 2^N possibilities,\n"
);
  printf ( "  seeking those inputs that make the function TRUE.\n" );
/*
  Compute the number of binary vectors to check.
*/
  ihi = 1;
  for ( i = 1; i <= n; i++ )
  {
    ihi = ihi * 2;
  }

  printf ( "\n" );
  printf ( "  The number of logical variables is N = %d\n", n );
  printf ( "  The number of input vectors to check is %d\n", ihi );
  printf ( "\n" );
  printf ( "   #        Index     ---------Input Values-----------------
-------\n" );
  printf ( "\n" );
/*
  Check every possible input vector.
*/
  solution_num = 0;

  for ( i = 0; i < ihi; i++ )
  {
    i4_to_bvec ( i, n, bvec );

    value = circuit_value ( n, bvec );

    if ( value == 1 )
    {
      solution_num = solution_num + 1;

      printf ( "  %2d  %10d:  ", solution_num, i );
      for ( j = 0; j < n; j++ )
      {
        printf ( " %d", bvec[j] );
      }
      printf ( "\n" );
    }
  }

//  Report.

  printf ( "\n" );
  printf ( "  Number of solutions found was %d\n", solution_num );
/*
  Shut down.
*/
  printf ( "\n" );
  printf ( "SATISFY\n" );
  printf ( "  Normal end of execution.\n" );
  printf ( "\n" );
  timestamp ( );

  return 0;
# undef N
}
```

Ritu Arora 9/13/2017 10:11 AM
Comment [1]: hotspot for parallelization

Steps for Using IPT

```
login3$ idev

c557-202$ source runBeforeIPT.sh

c557-202$ ../IPT circuit.c
"/work/01698/rauta/trainingIPT/exercise/circuit.c", line 274: warning:
          variable "len" was set but never used
    size_t len;
           ^
```

Ritu Arora 9/13/2017 10:03 AM
Comment [2]: check the path to this file

```
NOTE: We currently support only C and C++ programs.
```

Please select a parallel programming model from the following available options:

```
1. MPI
2. OpenMP
3. CUDA
2
```

NOTE: As per the OpenMP standard, a parallelized region/block of statements can have only one entry point and only one exit point. Branching out or breaking prematurely from a parallelized region/block of statements is not allowed. Please make sure that there are no return/break statements in the region selected for parallelization. However, exit/continue statements are allowed in parallel regions.

A list containing the functions in the input file will be presented, and you may want to select one function at a time to parallelize it using multi-threading.

Please choose the function that you want to parallelize from the list below

```
1 : main
2 : circuit_value
3 : i4_to_bvec
4 : timestamp
1
```

Please select one of the following options (enter 1 or 2 or 3)
1. Create a parallel region (a group of threads will be created and each thread will execute a block of code redundantly but in parallel)
2. Parallelize a for-loop (a group of threads will be created and each thread will execute a certain number of iterations of a for-loop)
3. Create a parallel section (TBD - this mode is currently unavailable)

2

```
for (i = 1; i <= n; i++) {
  ihi = (ihi * 2);
}
```
Is this the for loop you are looking for?(y/n)
n

OK - will find the next loop if available.

Note: With your response, you will be selecting or declining the parallelization of the outermost for-loop in the code region shown below. If instead of the outermost for-loop, there are any inner for-loops in this code region that you are interested in parallelizing, then, you will be able to select those at a later stage.

```
for (i = 0; i < ihi; i++) {
  i4_to_bvec(i,n,bvec);
  value = circuit_value(n,bvec);
  if (value == 1) {
    solution_num = (solution_num + 1);
    printf("  %2d  %10d:  ",solution_num,i);
    for (j = 0; j < n; j++) {
      printf(" %d",bvec[j]);
    }
    printf("\n");
  }
}
```
Is this the for loop you are looking for?(y/n)
y

Reduction variables are the variables that should be updated by the OpenMP threads and then accumulated according to a mathematical operation like sum, multiplication,etc.

Do you want to perform reduction on any variable ?(Y/N)
y

Please select a variable to perform the reduction operation on (format 1,2,3,4 etc.). List of possible variables are:
1. n type is int
2. value type is int
3. solution_num type is int
4. i type is int
3

Please enter the type of reduction you wish for variable [solution_num]
1. Addition
2. Subtraction
3. Min
4. Max
5. Multiplication
1

IPT is unable to perform the dependency analysis of the array named [bvec] in the region of code that you wish to parallelize. Please enter 1 if the entire array is being updated in a single iteration of the loop that you selected for parallelization, or, enter 2 otherwise.
1

Are there any lines of code that you would like to run either using a single thread at a time (hence, one thread after another), or using only one thread?(Y/N)
n

Would you like to parallelize another loop in the previously selected function or another one?(Y/N)
n

Are you writing/printing anything from the parallelized region of the code?(Y/N)
n

Running Consistency Tests

Compiling and Running the Generated Code

```
c557-202$ ls -ltr
total 45
-rw------- 1 rauta G-25072 14726 Sep 11 11:59 md.c
-rw------- 1 rauta G-25072  1856 Sep 11 11:59 matrix_mul.cc
-rw------- 1 rauta G-25072  1286 Sep 11 11:59 heat_serial.c
-rw------- 1 rauta G-25072  5833 Sep 11 11:59 circuit.c
-rw------- 1 rauta G-25072    81 Sep 11 11:59 calc_up.h
```

```
-rw------- 1 rauta G-25072   184 Sep 11 11:59 calc_up.c
-rw------- 1 rauta G-25072  1192 Sep 11 12:55 README.txt
-rw-r--r-- 1 rauta G-25072  5962 Sep 11 12:58 rose_circuit_OpenMP.c


c557-904$ icpc -qopenmp -o rose_circuit_OpenMP rose_circuit_OpenMP.c
c557-904$ ls -ltr
total 77
-rw-r--r-- 1 rauta G-25072 14726 Sep 11 11:59 md.c
-rw-r--r-- 1 rauta G-25072  1856 Sep 11 11:59 matrix_mul.cc
-rw-r--r-- 1 rauta G-25072  1286 Sep 11 11:59 heat_serial.c
-rw-r--r-- 1 rauta G-25072  5833 Sep 11 11:59 circuit.c
-rw-r--r-- 1 rauta G-25072    81 Sep 11 11:59 calc_up.h
-rw-r--r-- 1 rauta G-25072   184 Sep 11 11:59 calc_up.c
-rw-r--r-- 1 rauta G-25072  5962 Sep 11 12:58 rose_circuit_OpenMP.c
-rw-r--r-- 1 rauta G-25072  1358 Sep 11 12:59 README.txt
-rw-r--r-- 1 rauta G-25072 15691 Sep 11 13:03 matrix_mul.ti
-rw-r--r-- 1 rauta G-25072  2076 Sep 11 13:05 rose_matrix_mul_OpenMP.cc
-rw-r--r-- 1 rauta G-25072  4049 Sep 11 13:19 using_ipt_circuit.txt
-rwxr-xr-x 1 rauta G-25072 26668 Sep 11 13:37 rose_circuit_OpenMP


c557-904$ export OMP_NUM_THREADS=1


c557-904$ time ./rose_circuit_OpenMP

12 September 2017 10:08:23 PM

SATISFY
  C version
  We have a logical function of N logical arguments.
  We do an exhaustive search of all 2^N possibilities,
  seeking those inputs that make the function TRUE.
  The number of logical variables is N = 23
  The number of input vectors to check is 8388608

   #       Index    ---------Input Values------------------------

   1     3656933:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 1
   2     3656941:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1
   3     3656957:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
   4     3661029:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1
   5     3661037:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1
  6    3661053:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1
  7    3665125:   0 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1
  8    5754104:   1 0 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0
  9    5754109:   1 0 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
  10    5758200:   1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 0
  11    5758205:   1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1
  12    7851229:   1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 0 1
  13    7851261:   1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
  14    7855325:   1 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1
  15    7855357:   1 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1

  Number of solutions found was 15

SATISFY
  Normal end of execution.

12 September 2017 10:08:23 PM

real    0m0.402s
user    0m0.324s
sys     0m0.003s


c557-904$ export OMP_NUM_THREADS=16


c557-904$ time ./rose_circuit_OpenMP

12 September 2017 10:08:39 PM

SATISFY
  C version
  We have a logical function of N logical arguments.
  We do an exhaustive search of all 2^N possibilities,
  seeking those inputs that make the function TRUE.
  The number of logical variables is N = 23
  The number of input vectors to check is 8388608

   #      Index    ---------Input Values------------------------

   1    3656933:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0 1
   2    3656941:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1
   3    3656957:   0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
   4    3661029:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1
  5    3661037:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1
  6    3661053:   0 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1
  7    3665125:   0 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 1
  1    7851229:   1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 0 1
  2    7851261:   1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
  3    7855325:   1 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1
  4    7855357:   1 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1
  1    5754104:   1 0 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0
  2    5754109:   1 0 1 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1
  3    5758200:   1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 0
  4    5758205:   1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1

  Number of solutions found was 15

SATISFY
  Normal end of execution.

12 September 2017 10:08:40 PM

real     0m0.112s
user     0m1.294s
sys      0m0.004s
```