Introduction

Spare cells are designed for further design change. The spare cells always are placed evenly at the empty placeable space where is unused by other standard cells. When functional changes are desired after the design is finished, the selected spare cells would be activated to implement the desired changes by make changes to fewer upper layer masks for the design.

Figure 1 shows the spare cells placement example. The green blocks are I/O cells and other hard macro blocks. These blocks are always placed around the boundary of the chip. The blue area is for the standard cell placement. The spare cells are highlighted in white, and are spread evenly in the standard cell placement area. The spread spare cells are good for further spare cells selection.

Problem Descriptions and Assumptions

Most of spare cells can only provide basic logical functions. The basic logical functions include NOT, AND, NAND, OR and NOR. Some of spare cells can be complex logical functions, such as HA (Half Adder), FA (Full Adder), AOI (And-Or-Inverter) or OAI (Or-And-Inverter). The number of spare cells for each logical function is limited. You can use basic logical spare cells and complex logical
spare cells to implement a required changed function. For example, 2 NAND and 1 AND spares cells can implement an AOI22 function. Figure 2 shows the example of AOI22 implementation by 2 NAND and 1 AND cells.

Note that you CANNOT modify the original netlist since the original netlist was fixed in the last implementation. You can re-synthesize the changed function lists according to the existing resources of spare cells.

![Figure 2: AOI22 can be implemented by 2 NAND and 1 AND cells.](image)

Given functional change requirements, please use these existing spare cells to implement the required functional changes. **The objective of this problem is to minimize the final interconnection cost.**

The interconnection cost of a net $i$ can be defined by the half perimeter of the minimum-bounding rectangle:

$$xspan(i) + yspan(i).$$

Where, $xspan(i)$ and $yspan(i)$ are the horizontal and vertical spans of the minimum-bounding rectangle of net $i$. Figure 3 shows the interconnection cost of a net $i$. There are 4 terminals A, B, C and D for net $i$. The minimum-bounding rectangle is shown in red. The interconnection cost for net $i$ is defined by the half perimeter of the minimum-bounding rectangle.
Figure 3: The minimum-bounding rectangle of net $i$ with 4 terminals A, B, C and D.

The interconnection cost function is taken as:

$$\sum_{i \in \text{nets}} (x\text{span}(i) + y\text{span}(i)).$$

The objective of this problem is to minimize the above function.

**Problem Assumptions**

- The chip is a cell-based design, and no other hard macro blocks exist.
- The input/output terminals of the design are placed on the boundary of the chip. The input/output terminals are shown in red points in Figure 4.
- The original coordinate of the chip is set to the bottom left corner of the chip, i.e. $(0,0)$. See Figure 4.
- The coordinate of the cell $(x,y)$ is relative to the coordinate of the chip. See Figure 4.
- The size of cells and I/O pins are assumed to be zero. Each cell or I/O pin can be represented as a point.

Figure 4: The problem assumptions
Input/Output Files and Formats

Input Files
There are 4 inputs files listed as follows:

- Technology library file in text format (*.lib)
- Original netlist and placement in Cadence DEF 5.4 format (*.def)
- Available spare cell list in text format (*.spare)
- Changed list in Verilog CELL-level format ( *.eco)

The conventional naming extensions of the input files are *.lib, *.def, *.spare, and *.eco for technology library, original netlist/placement, spare cell list file and changed list file, respectively.

The technology library describes the function for each cell. A library cell is specified in the following format:

```
CELL  <cellName>
    OUT <outputPinName> = <logical-function> | UNKNOWN;
    ...
    INPUT <inputPinName>;
    ...
ENDCELL <cellName>
```

- \(<cellName>\) is the name of the cell in the target library.
- \(<outputPinName>\) is the name of the output pin. \(<\text{logical-function}>\) describes the logical function of the output pins. There may exist one or more outputs in a library cell.
- \(<\text{logical-function}>\) is an equation written in conventional algebraic notation using the operators + for OR, nothing (space) for AND, ^ for XOR, ’ for NOT, and parentheses for grouping. The names of the literals in the equation define the input pin names for the cells. The equation terminates with a semicolon. The UNKNOWN is used for some of output pins functions are unknown. In this case, the cell would work as a black box. You **CANNOT apply any modification on these UNKNOWN cells during the spare cells selection.**
- \(<inputPinName>\) is the name of input pins.

For example, the 2-input OR function is represented as \(O=I1+I2\), where \(O\) is the output, \(I1\) and \(I2\) are the inputs, respectively; the 2-input XOR function is represented
The spare cell list is in text format. The syntax of the spare cell list is:

```
spareCellName CellTypeName X-coordinate Y-coordinate
```

The spare cell list example is shown as follows:

```
SPARE_001 NAND2 340 230
SPARE_002 NAND2 350 230
SPARE_003 AND2 445 130
SPARE_004 INV 400 160
SPARE_005 INV 440 160
```

The required change list is in CELL-level Verilog. The change list example is shown as follows:

```
INV ECO1 (.O(newOutputNet1), .I(inputNet1));
AOI22 ECO2 (.O(oldOutputNet2), .A1(inputNet2_1), .A2(.inputNet2_2)
            .B1(inputNet2_3), B2(inputNet2_4));
```

Note that the net name of the original driver has to be renamed, if the output net name exists in the original netlist. In this example, the net name of the original driver of net `oldOutputNet2` has to be renamed after the change is implemented. It can be seen that in the example on p.11, the name `net3` is used again in the new implementation, so it is renamed as `deleted 1`. The new name can be any other label, we suggest the naming rule is `deleted i` if it is the `i`th renamed net.

**Output Files**

There are 2 output files listed as follows:

- Final netlist and placement in Cadence DEF 5.4 format (*.def.eco)
- Final report including spare cell selection results, the final interconnection cost and CPU execution time. (*.rpt)

The conventional naming extensions of the input files are *.def.eco, and *.rpt for final
netlist and placement, spare selection report and final cost and CPU execution, respectively.
The final netlist and placement results should be represented in DEF.
The syntax of the final report is shown as follows:

```
[SELECTION]
  CHANGE_CELL_NAME SPARE_CELL_NAME_1 ...
...
[END SELECTION]
COST = n
CPU = m sec.
```

Example of the report files is shown as follows;

```
[SELECTION]
  ECO1 SPARE_004
  ECO2 SPARE_003 SPARE_002 SPARE_001
[END SELECTION]
COST = 2388
CPU = 134.3 sec.
```

**An Input/Output Files Example**

The given original netlist is shown in Figure 5.

Figure 5: The given original netlist

The description for AND2, OR2, NOT, FA and BLACK in target technology library
format is:

CELL  AND2
    OUTPUT O = I1 I2;
    INPUT I1;
    INPUT I2;
ENDCELL AND2
CELL  OR2
    OUTPUT O = I1+I2;
    INPUT I1;
    INPUT I2;
ENDCELL OR2
CELL  NOT
    OUTPUT O = I';
    INPUT I ;
ENDCELL NOT
CELL  FA
    OUTPUT S = A ^ B ^ CI;
    OUTPUT CO = CI A + CI B + A B;
    INPUT A ;
    INPUT B ;
    INPUT CI ;
ENDCELL FA
CELL  BLACK
    OUTPUT Q = UNKNOWN ;
    INPUT CK ;
    INPUT D ;
ENDCELL BLACK

The original netlist and placement in DEF is:

VERSION 5.4 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[[]]" ;
DESIGN cad ;
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 5000 5000 ) ;
COMPONENTS 4 ;
- U1 AND2 + PLACED ( 2000 4000 ) FN ;
- U3 OR2 + PLACED ( 1000 2000 ) FS ;
- U4 NOT + PLACED ( 3000 1000 ) FS ;
END COMPONENTS

PINS 3 ;
- in1 + NET in1 + DIRECTION INPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 3000 ) W ;
- in2 + NET in2 + DIRECTION INPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 2000 ) W ;
- out + NET out + DIRECTION OUTPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 1000 ) W ;
END PINS

SPECIALNETS 2 ;
- VCC
  ( * VCC )
  + USE POWER ;
- GND
  ( * GND )
  + USE GROUND ;
END SPECIALNETS

NETS 6 ;
- net2
  ( U2 O )
  ( U3 I2 ) ;
- net1
  ( U1 O )
  ( U3 I1 ) ;
- net3
  ( U3 O )
  ( U4 I ) ;
- out
  ( U4 O )
  ( PIN out ) ;
- in2
  ( PIN in2 )
  ( U2 I1 )
Note that the power net VCC and ground net GND are excluded in the final interconnection cost.

The original interconnection cost is:

$$(\text{net1+net2+net3+in1+in2+out}) = 3000+1000+3000+5000+5000+2000 = 19000$$

The available spare cell list is:

- SPARE_001 AND2 3000 3000
- SPARE_002 AND2 1000 1000
- SPARE_003 NOT 1000 4000

The original placement and spare cell placement is shown in Figure 6.
Figure 6: The given original netlist and spare cell placement

The required change list is:

```plaintext
AND2 ECO1(.I1(net1), .I2(net2), .O(net3));
```

In this example, SPARE_001 is selected to implement the required change finally.

The new DEF file would be:

```plaintext
VERSION 5.4 ;
NAMESCASESENSITIVE ON ;
DIVIDERCHAR "/" ;
BUSBITCHARS "[]" ;
DESIGN cad ;
UNITS DISTANCE MICRONS 1000 ;
DIEAREA ( 0 0 ) ( 5000 5000 ) ;

COMPONENTS 5 ;
- U1 AND2 + PLACED ( 2000 4000 ) FN ;
- U3 OR2 + PLACED ( 1000 2000 ) FS ;
- U4 NOT + PLACED ( 3000 1000 ) FS ;
- ECO_SPARE_001 AND2 + PLACED ( 3000 3000 ) N ;
/* Note: The naming rule for spare cells is ECO+_spare_cell_name.
FN, N, FS denote the cell direction. In our formulation, cell size
is assumed to be 0, thus, this label is meaningless. You can always
label it in N */
END COMPONENTS

PINS 3 ;
- in1 + NET in1 + DIRECTION INPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 3000 ) W ;
- in2 + NET in2 + DIRECTION INPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 2000 ) W ;
- out + NET out + DIRECTION OUTPUT + USE SIGNAL
  + LAYER metal3 ( 0 0 ) ( 1 1 ) + PLACED ( 5000 1000 ) W ;
END PINS

SPECIALNETS 2 ;
```
- VCC
  (  *  VCC  )
  + USE POWER ;
- GND
  (  *  GND  )
  + USE GROUND ;
END SPECIALNETS

NETS 7 ;
- net2
  ( U2 O )
  ( U3 I2 )
  ( ECO_SPARE_001 I2 ) ;
- net1
  ( U1 O )
  ( U3 I1 )
  ( ECO_SPARE_001 I1 ) ;
- net3
  ( ECO_SPARE_001 O )
  ( U4 I ) ;
- out
  ( U4 O )
  ( PIN out ) ;
- in2
  ( PIN in2 )
  ( U2 I1 )
  ( U1 I2 ) ;
- in1
  ( PIN in1 )
  ( U2 I2 )
  ( U1 I1 ) ;
- deleted_1
  ( U3 O ) ;
- VCC
  + USE POWER ;
- GND
  + USE GROUND ;
END NETS
END DESIGN
The final interconnection cost is:

\[(\text{net1+net2+net3+in1+in2+out}) = 4000+3000+2000+5000+5000+2000 = 21000\]

Note that the final interconnection cost does not include the net deleted_1 cost. This is because the net deleted_1 will be a one-terminal net, and the original routing of this net will be removed in some router.

The final report is:

```
[SELECTION]
    ECO1    SPARE_001
[END SELECTION]
COST = 21000
CPU = 3.3 sec.
```

**Language and Platform**

Programming language should be either C or C++, and program must be developed under Sun/Solaris, HP/UNIX or x86/LINUX based platform.

**Grading Strategy**

The functional correctness of the final result should be guaranteed; otherwise the score for this test case would be zero. If core dump occurs or CPU time is more than 8 hours while executing a test case, the score for this test case would be zero.

Total score of a test case is calculated as follows:

- The final routing cost (60%)
- CPU time (30%)
- Graphical interface for displaying the selection results (10%)

**References**

- Cadence LEF/DEF 5.4 Language Reference Manual
Available Cadence DEF Parsers on WWW

- Cadence DEF 5.4 parser: http://www.openeda.org