Circuit Partition Algorithm

ECE201A project

Yunhui Ma

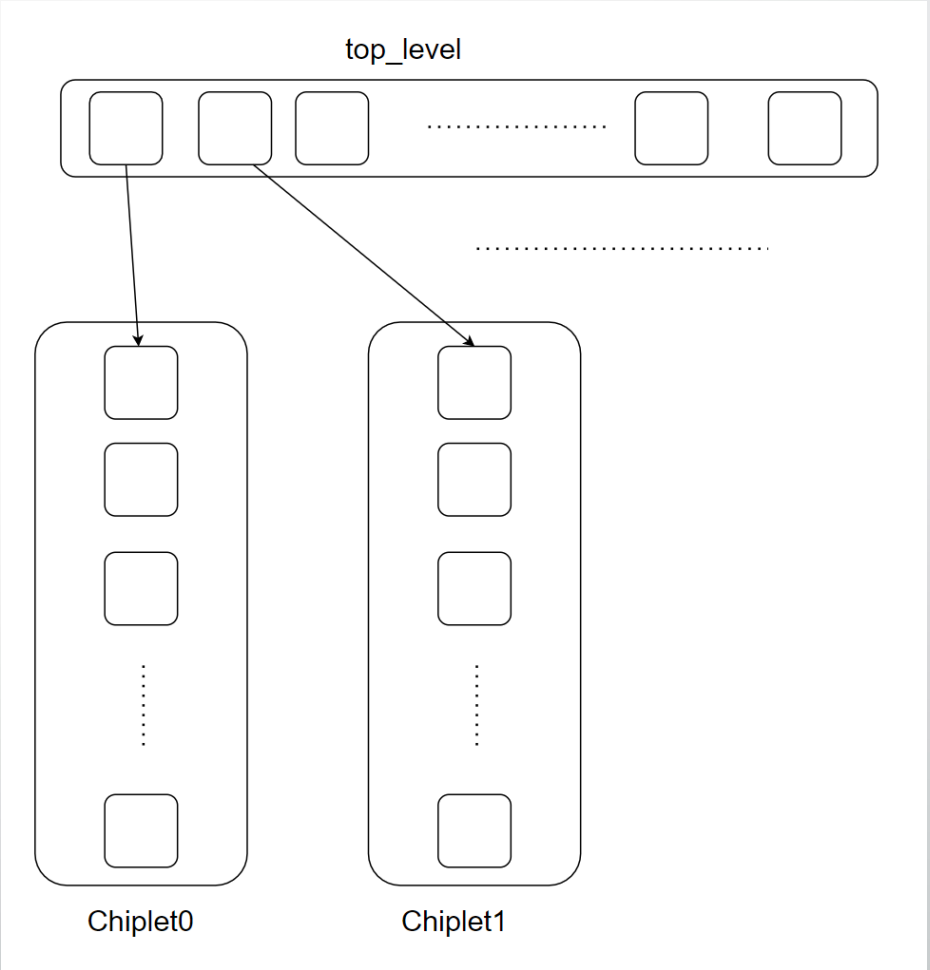
Overall Steps:

- Convert the lef files and top_level.v to oa
- Read the top_level and separate the instances into two chiplets
- Create the new top_level
- Calculate chiplets and new top_level area.
- Convert the oa to verilog
- Generate .txt files
- Run python code and get results.

Partition Algorithm

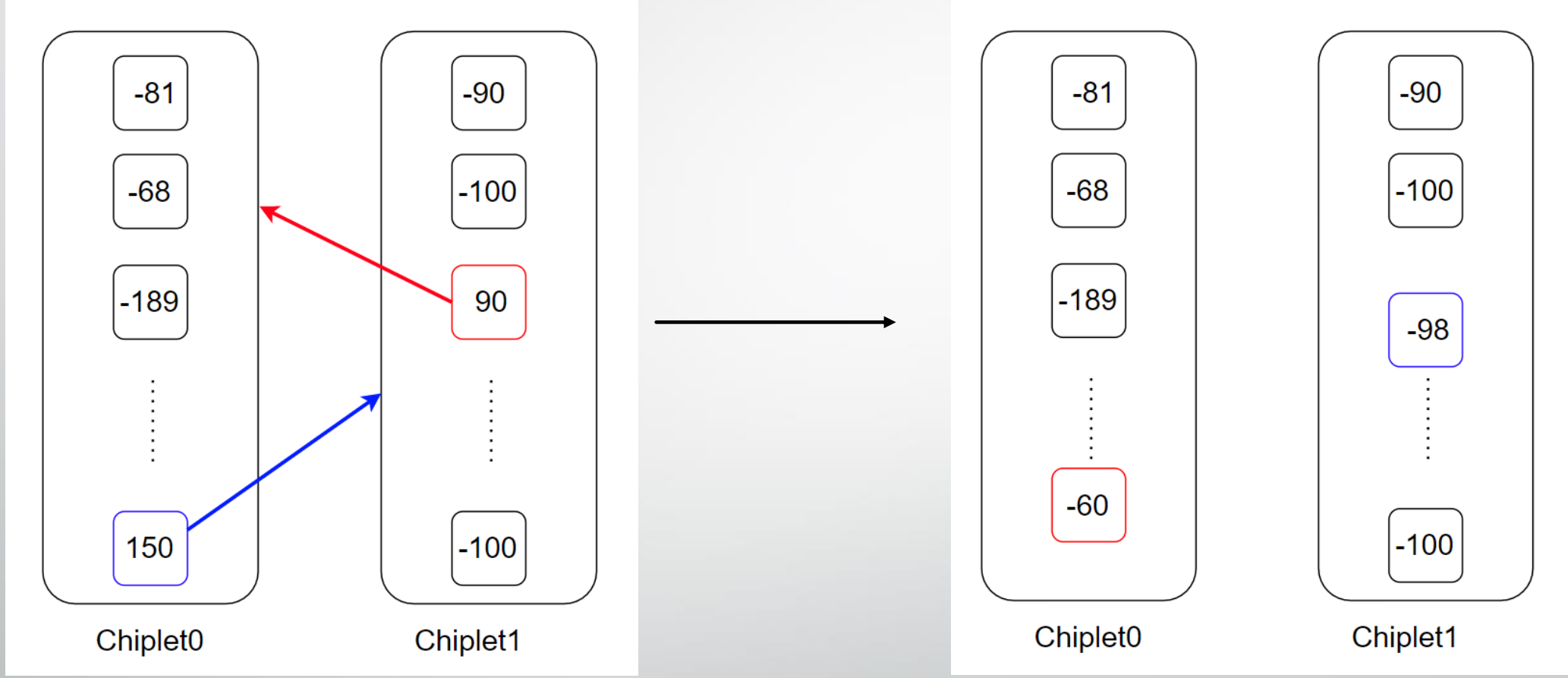
- Separate the instances into two chiplets
- Calculate their FM score
- Move the instances with positive FM score to the opposite chiplet.

```
if(chiplet_map[i].find(connected_instName) != chiplet_map[i].end()){
    //found in the same chiplet
    (it->second).score -= num_bits;
}else{
    //in other chiplets
    (it->second).score += num_bits;
}
```



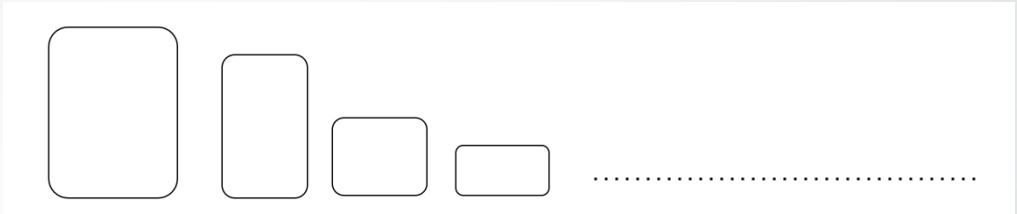
Calculating FM score

```
oaUInt4 oaInstTerm::getNumBits(  ) const
```

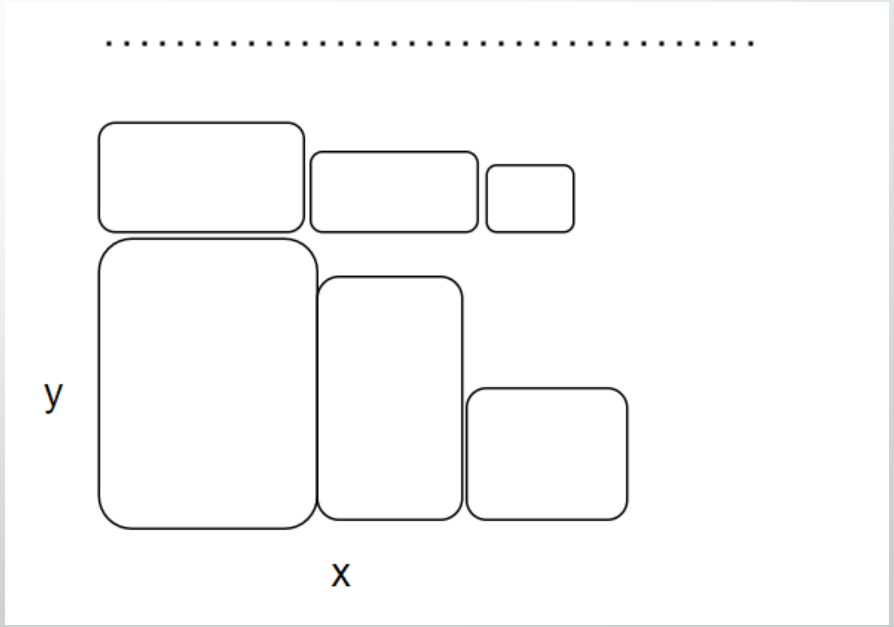


Area Calculation Algorithm

- Sort the instances



- Put the first instance into one position.
- Keep placing the instances on the right side.
- Check If x > 2*y, start a new row.
- Repeat the algorithm unit no more instances




```
600 358
```

```
762 398
```

```
838 831
```

Result

```
431.089234409
1941.005
```

Average: 431 um
Wirelength: 1941.005 um

```
BLOCK_CHIPLET0,block_chiplet0
BLOCK_CHIPLET1,block_chiplet1
top_level_new,top_level_new
```

Original supplied code result

```
(2492.715, 'SDATAR6[20]')
(893.495, 'SDATAW7[21]')
775.856697215
3649.14
```

Conclusion

- The algorithm is not optimized, it only generates two chiplets.
- The FM partition only iterates one time.
- Area calculation is not efficient. It could generate a larger height or width.

Improvement

- Implement the algorithm that allows users choose the number of chiplets that they want to generate under certain constraints.
- Implement efficient FM partition to try to achieve lowest FM score.
- Efficient Area calculation algorithm to produce lowest area but not violate the constraints.

Thank You