ABSTRACT
Current mobile GPU technologies are still relatively immature and require substantial improvements to enable wireless devices to perform the complex graphics-related functions. Traditional way of extracting parallelism requires more processor cores and complex programming. But mobile devices are not designed to hold many processors or execute heavy-computing program. We propose a new combinatorial architecture based on balanced incomplete block designs (BIBD) that is more suitable for mobile GPUs/CPUs, i.e., mobile heterogeneous computing, with limited resources and relative greater performance. This combinatorial architecture that has a unique property—multi-core running on a sequential code. This architecture can be used for both mobile CPUs and GPUs. Some minor adjustments to a regular compiler are needed for loading the instructions.

INADEQUACIES OF EXISTING METHODS
To increase performance, processor manufacturers extract parallelism through multi-core systems. Besides several complications, this requires tremendous efforts for organization special software for parallel processing.

PAIRWISE-BALANCED BLOCK DESIGNS
The block design is a famous combinatorial structure, which is a subset satisfying some conditions. In a processor, partitions designed by these block designs allows concurrent processing of data independent instructions in executable sequential code because the partitioning is done at the compiler-level [1]. A balanced incomplete block design on a set of v elements is a collection of k subsets such that each element appears exactly in r subsets and each pair of elements appears exactly in λ subsets. The balanced incomplete block design is represented by its parameters in standard notation: (b, v, r, k, λ).

Let's consider that there is an interaction between two objects or threads Xi and Xj. We can find both of these elements in PBi. Instead of transporting data from one block to another, the processing can take place instantaneously. As the result, corresponding processing blocks can be updated independently and simultaneously without interfering with others and progress in parallel. Thus this architecture works on sequential code while extracting parallelism. Other possible designs are (13, 13, 4, 4, 1), (21, 21, 5, 5, 1), (26, 13, 6, 3, 1), (35, 15, 9, 3, 1).

EXPERIMENTAL RESULTS
Our uniprocessor software simulation showed the speed increase of over 30% compared to a regular processor. Our implementation of this architecture in FPGA using configurable logic blocks (CLB) showed up to 27% speed increase with the significant reduction in transportation energy [2].

REFERENCES