# FPGA Synthesis for Minimum Area, Delay and Power*

**Kuo-Rueih Ricky Pan and Massoud Pedram**
Department of Electrical Engineering - Systems
University of Southern California, Los Angeles, CA 90089

In this paper, we address the problems of minimizing the area, delay and power during synthesis of field programmable gate arrays (FPGAs). We use Boolean decomposition techniques to minimize the number of configurable logic blocks (CLBs), the depth of the network and the power dissipations. We use OBDDs to represent functions so that our methods can be implemented more effectively. Our mapping algorithm is based on function decomposition which was pioneered by Ashenhurst [1].

**Definition** A function $f(x_0, \ldots, x_{n-1})$ is said to be *decomposable* under *bound set* $B = \{x_0, \ldots, x_{i-1}\}$ and *free set* $\{x_i, \ldots, x_{n-1}\}, 0 < i < n,$ if $f$ can be transformed to $f'(g_0(B), \ldots, g_{j-1}(B), x_i, \ldots, x_{n-1})$, where $0 < j < i$. Each $g_i$ is referred as a *g-function*. The number of the support of $f$ is denoted as $supp(f)$. The variable support reduction for the decomposition, $supp\_red(f, B)$, is equal to $supp(f) - supp(f') = i - j$.

Given a multiple-output function $F = \langle f_0, \ldots, f_{M-1} \rangle$, the proposed algorithm (FGSyn) is carried out recursively as follows.

**Algorithm** $fg\_synthesis$ [2]:
```
1. For each bound set $B$ of size $K$ do /* $K$-LUT */
2.    Partition $F$ into N groups $\langle H_0, \ldots, H_{N-1} \rangle$
         such that each group is decomposable;
3.    For each group $H_n$, generate all its possible
         g-functions and store them in $pgs_n$;
4.    Extract common g-functions according to their
         cost function ($pg$_cost) so as to produce
         a minimum cost valid encoding of all $f_m$'s;
5.    Compute and store the specified cost function
         (decomp_cost) of the final encoding;
6. Pick the bound set $B^*$ with the best decomp_cost;
7. Decompose $F$ with respect to $B^*$ and generate $F'$
     for next iteration.
```

Note that $pg\_cost$ and $decomp\_cost$ are specified according to the objective function being minimized during synthesis. For area minimization:

$$pg\_cost(g_i) = supp(g_i)$$

$$decomp\_cost(F, B) = ave\_supp\_red(F, B).$$

Our algorithm FGSyn, has been implemented in C and incorporated into the SIS environment. We ran FGSyn on a number of benchmarks for Xilinx XC3000 device and compared it with Chortle-crf, ASYL and mis-pga (new). FGSyn does $24.3\%$ better than Chortle-crf, $21.3\%$ better than ASYL and $15.0\%$ better than mis-pga (new). The memory requirement of FGSyn is only $10\%$ more than that of the mis-pga (new) whilst its run time is about $27\%$ lower.

We assume each LUT contributes to a constant delay. Thus, the delay is determined by the maximum number of LUTs on any path from primary inputs to primary outputs. Our delay minimization algorithm FGSyn_d is based on the Huffman algorithm for constructing minimum average code length.

**Definition** Given a multiple-output function $F = \langle f_0, \ldots, f_{n-1} \rangle$ and bound set $B$, we can write:

$$avg\_supp\_red(F, B) = \frac{1}{n} \sum_{i=0}^{n-1} supp\_red(f_i, B).$$

$$avg\_depth(F, B) = \frac{1}{n} \sum_{i=0}^{n-1} depth(f_i, B)$$

$$pg\_cost(g_i) = depth(g_i).$$

Given a collection of bound sets with respect to which $F$ is decomposable, we pick the bound set $B^*$ such that 1) $ave\_depth(F, B^*)$ is minimum and 2) $ave\_supp\_red(F, B^*)$ is maximum.

This minimum delay decomposition scheme has been incorporated into FGSyn as FGSyn_d. Our results show an average 8% reduction in the network depth over the FlowMap-r results.

We show that each $pg$ function contributes to the power consumption of the circuit by:

$$\Delta P(pg_i) = sw(pg_i) nfo(pg_i) + \sum_{x_i \in support(pg_i)} sw(x_i).$$

The number of fanouts of a $pg_i$ is $> 1$ only when it is shared among two or more outputs. At the same time, the larger this number, the smaller the number of $g$-functions required to produce a valid encoding. Therefore, the power contribution of each $pg_i$ is divided by its number of fanouts to yield the cost of a $pg_i$ as:

$$pg\_cost(pg_i) = \frac{\Delta P(pg_i)}{nfo(pg_i)}.$$

We calculate the power cost for decomposing $F$ with respect to $B$ as the difference between the circuit power before and after the decomposition and then pick the bound set $B^*$ with maximum avg_supp_red and minimum power_cost (in that order) and finally decompose $F$ with respect to $B^*$ to generate $F'$.

This low power decomposition algorithm has been incorporated into FGSyn as FGSyn_p. The results show 18% reduction over mis-pga(new) in terms of power consumption.

## References

[1] Robert L. Ashenhurst. The decomposition of switching functions. In *Proceedings of International Symposium on Theory of Switching Functions*, April 1959.

[2] Y-T. Lai, K-R. R. Pan, and M. Pedram. FPGA synthesis using function decomposition. In *Proceedings of the International Conference on Computer Design*, pages 30–35, October 1994.

*This work was supported in part by ARPA under contract no. F33615-95-C-1627 and by SRC under contract no. 94-DJ-559.