Area Time Power Estimation for FPGA Based Designs at a Behavioral Level

Sébastien Bilavarn, Guy Gogniat, Jean Luc Philippe

LESTER - Centre de Recherche 56100 Lorient France
bilavarn@iuplo.univ-ubs.fr

ABSTRACT: A new performance estimation technique for FPGA implementation based designs is presented. The interest and originality of the method is to rapidly test a great number of implementation solutions while staying independent as far as possible of the technology used, and to include power consumption estimation. Thanks to this method, the designer can quickly have realistic information about the performances of a design, starting from a behavioral specification.

1. INTRODUCTION

Fast prototyping of complex applications such as digital signal processing systems over Field Programmable Gate Array technology has become a very important issue due to the increasing number of FPGA components and their recent architecture evolutions. A rapid estimation of FPGA utilization rate, time execution and power consumption (ATP in the following) at the behavioral level is necessary to allow an efficient exploration of a large design space and to select the best implementation solution (algorithm vs. architecture). The methodology described here aims at guiding the design process of heterogeneous architectures such as in the codesign issue or multi FPGA based systems. So the main quality of the performance estimator is to be fast and accurate enough (trade-off precision/ complexity) to guaranty the choice of the best implementation solution for a given functionality. Many techniques are described in the literature estimating the processing unit [1][2](resource estimation, layout estimation), control or memory units [3] for one or two sort of constraints, area and time most of the time. But few techniques targets FPGA and there is no to our knowledge global estimation methods (i.e. power, area, speed) starting from a behavioral level. The work presented in [4] describes a power consumption estimation method based on the activity rate of the design. In [5], the method is set up on the estimation of area and timing values starting from a Register Transfer Level netlist resulting from a High Level Synthesis (HLS) tool.

This approach aims at speed up the HLS flow by getting rid of the Partitioning, Placement, Routing and Timing optimization steps, and estimating the physical design characteristics instead. The method is very accurate but has two main drawbacks caused by the use of a HLS tool: it might be too slow in the case of the exploration of a wide design space such as in codesign, and the estimation process is strongly dependent of the technology used (Xilinx, Altera, Actel, etc.). Our estimation process allows both to be fastest (to the detriment of accuracy) and far less dependent of the technology through the characterization of the target FPGA in a component library. Furthermore, FPGA power consumption estimation is included in order to give a complete characterization of the design.
The outline of this paper is as follow: In section 2, an overview of the resources estimation flow is presented. Then section 3 explains the resources allocation algorithm whereas section 4 describes the dynamic performance characterization. In section 5, we show how control and hierarchy are included in the estimation process and section 6 presents speed area and power mapping. Finally, preliminary results and a conclusion are given.

2. RESOURCES ESTIMATION FLOW

First, the behavioral description given in a high level language (we actually consider the C language) is translated into a HCDFG (Hierarchical Control Data Flow Graph) [6]. The HCDFG model is built on a graph $G = \{V, E\}$, where each node $v_i \in V$ corresponds to a part of the application. There are five different types of nodes (Figure 1):

- v_i^{CDFG} contains hierarchical structures of the graph,
- v_i^{Cont} contains control structures of the graph,
- v_i^{DFG} contains straight forward code,
- v_i^{Data} contains data used in the application,
- v_i^{Proc} contains processing operations implied in the application.

Note that v_i^{Data} and v_i^{Proc} nodes are parts of v_i^{DFG}

nodes and as illustrated in Figure 1 v_i^{CDFG} nodes encapsulate v_i^{DFG} and v_i^{Cont} nodes. An edge $e_{i,j}$ gives the direction of the control flow from node v_i to v_j.

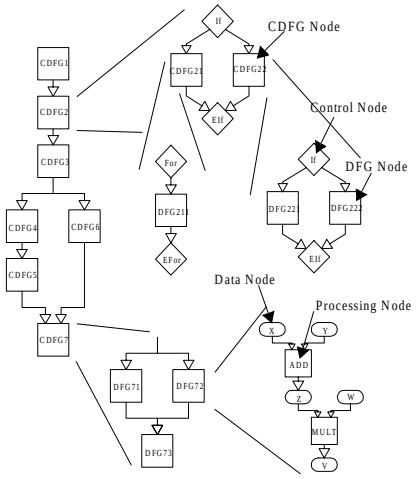

Fig 1 • HCDFG graph of the application

The estimation process explores progressively the entire graph (from DFG nodes to CDFG) in order to estimate the performance of the whole application. To reach this goal, our approach is composed of 4 steps: the resources allocation step (section 3) allocates for each processing node a unique functional unit and computes the clock cycle value. The second step (section 4) estimates each DFG of the graph in term of number of resources vs. number of clock cycles (i.e. dynamic curve). Then, the third step (HCDG estimation) merges the dynamic curves to consider the entire graph (section 5). The goal of that step is to examine all the control structures of the graph in order to estimate the final number of resources needed taking into account resource sharing. Finally, the resulted estimation is mapped on a given FPGA library (section 6).

3. RESOURCES ALLOCATION

The first step in the resources allocation process is to list all the operation types and the data formats within the whole graph. Then, the most suited operator is selected among the component library for each processing node. This step, is based on a Bipartite Matching Resolution, where the cost function who selects the best operator among a set of operators targets to satisfy the timing requirements while minimizing area and power consumption. The component library is described in a characterization file in which the functional units and their ATP characteristics are described. The area and delay of each functional unit is obtained through the synthesis tool (e.g. Foundation for Xilinx FPGAs), and the power consumption is directly measured on the target FPGA for different parameters (data formats, clock frequency, data type, etc.). Detailed explanations about the component library definition can be found in [7]. Once the resources allocation has been performed, the clock period of the design is set as the delay of the fastest functional unit. Each node can then be assigned a number of clock cycle, in order to apply a classical DFG estimation method based on the computation of the as soon as possible (ASAP) and as late as possible (ALAP) dates [1][2], or a scheduling of the graph [8], according to the specification complexity.

4. DFG DYNAMIC ESTIMATION

The ATP performances are characterized in a dynamic way, which means that each DFG is estimated for a variable time constraint. So estimation

```
DFGEstimation(V, R^AB, Ta, Δt, N)
    E_P = {}
    For all nodes v_i^DFG ∈ V^DFG
        E_P = {}
        E_P = DefineExplorationPoints(Ta_i, Δt_i, N)
        For all resources r^AB_i ∈ R^AB / r^AB_i is used in v_i^DFG
            For all timevalue ep_i ∈ E_P
            ComputeEstimationValue(ep_i, r^AB_i)
            End For
        End For
    End For
```

results (dynamic curves) authorize an exploration of several solutions since the varying parallelism of the design can be taken into account.

A global time constraint Ta_i is computed for each DFG of the graph and an exploration zone is defined in order to reduce the dynamic estimation algorithm complexity. The computation of an exploration zone (Figure 2) is performed as follow: Let Δt_i be the temporal exploration zone , and N the number of exploration points around Ta_i. Ta_i, Δt_i and N can be user defined or obtained from a system level analysis (like the method described in [9]). For all resources of the DFG, we compute the number of functional units needed for each time constraint $Ta_i + k.\delta t_i$, with $-N/2 \leq k \leq N/2$ and $\delta t_i = \Delta t_i / N$, as presented in Figure 2.

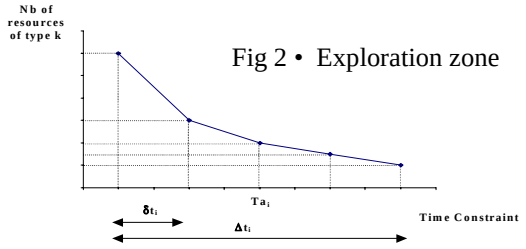

Fig 2 • Exploration zone

5. HIERACHICAL CDFG ESTIMATION

Once each DFG is estimated, we take into account control (loops, branches) and hierarchy in order to gradually estimate the entire design. As explained

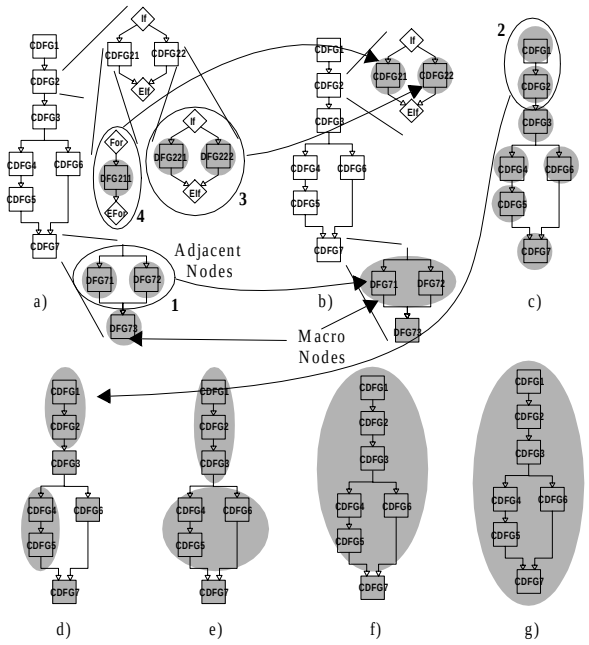

Fig 3 • Hierarchical estimation of the graph

in the following this step is based on the construction and the merging of Macro nodes. Figure 3 shows an example of the progressive hierarchical estimation of the graph through the clustering of the Macro nodes. At the beginning of this step each DFG of the graph is labelled as Macro node. Macro nodes are clustered if they are labelled as adjacent. Then, the combination of the estimated curves of each Macro node depends on the type of dependence between the nodes (i.e. type of adjacent nodes):

case 1 • Two estimated macro nodes are executed in parallel (Figure 3.a.1). In this case, time constraint T for the resulting macro node (i.e. after clustering) is $T=MAX\ (T_1,T_2)$, where T_1 and T_2 are respectively the time constraints for each macro node. Concerning the number of functional unit computation, a distinction based on the operator'size is made:

- coarse analysis (small operators): the number of functional units of type k allocated for a number of clock cycle T, $N_k(T)$ is:

$$N_k^{(mn_1, mn_2)}(T) = N_k^{mn_1}(T_1) + N_k^{mn_2}(T_2)$$

this method leads to an estimation overhead but reduce the estimation algorithm complexity. So, it is used for the estimation of small operators whose area and power consumption cost have a small impact on the final design.

- precise analysis (area and power consuming operators): in this case, the analysis of the temporal use of the operators gives the exact number of operators needed:

$$N_k^{(mn_1, mn_2)}(T) = MAX_{t \in [0;T]} \left(\sum_{t=0}^{T} \left[N_k^{mn_1}(t) + N_k^{mn_2}(t) \right] \right)$$

case 2 • Two estimated nodes are executed in sequence (Figure 3.c.2): resource re-using is performed by taking the maximum number of resources for the two macro nodes:

$$N_k^{(mn_1, mn_2)}(T) = MAX\left(N_k^{mn_1}(T_1), N_k^{mn_2}(T_2) \right)$$

$$T = T_1 + T_2$$

case 3 • Two estimated nodes are parts of a branch structure (Figure 3.a.3): the estimation method is based on the Probability-based Analysis of CFG Models [3]. Each transition arc between the nodes has a probability of execution. Hence, time constraint and resource requirement are computed as follow:

$$T^{branch} = [P(mn_1) \cdot T_1] + [P(mn_2) \cdot T_2]$$

$$N_k(T^{branch}) = MAX\left(N_k^{mn_1}(T_1), N_k^{mn_2}(T_2) \right)$$

case 4 • One estimated node is included in a loop structure (Figure 3.a.4): we suppose that the loop body estimation results $N_k(T)$ are known. There are two ways of proceeding according to the type of the loop:

- deterministic loops: the number of iterations is a known fixed value *N*. For a given loop configuration, let *i* be the number of iterations and *b* the number of loop bodies (*i* * *b* = *N*). The number of resources is:

$$N_k^{loop}(T_{loop}) = N_k(T) \cdot b$$

$$T_{loop} = T \cdot i$$

- non deterministic loops: in this case, the number of loop iterations *N* may be user defined or determined by profiling. There can be only one configuration of one loop body repeated *N* times. In this case:

$$N_k^{loop}(T_{loop}) = N_k(T)$$

$$T_{loop} = T \cdot N$$

Since the HCDFG estimation process leads to a dynamic estimation (to encourage design space exploration), the techniques presented above are applied to each point in the exploration zone. Hence, dy-

namic curves are obtained at the end of each adjacent nodes combination, until the top level of hierarchy is reached. The HCDFG estimation algorithm is presented below.

```
HCDFGEstimation(V)
    MN = {}
    MN = CreateMacroNode(V^DFG)
    While NumberOfMacroNode ≠ 1
        For all couple of MacroNode (mn_i, mn_j) ∈ MN
            if (mn_i, mn_j) are Adjacent MacroNode then
                ComputeMixedEstimation(mn_i, mn_j)
            CreateNewMacroNode(mn_ij)
            RemoveAdjacentMacroNode(mn_i, mn_j)
            If a hierarchical node v_i^CDFG ∈ V^CDFG is composed of only one MacroNode then
                ReplaceHNodewithMNode(v_i^CDFG, mn_ij)
        End For
    End While
ComputeMixedEstimation(mn_i, mn_j)
    Case AdjacentMacroNodeType(mn_i, mn_j)
      When Parallel    ComputeParallelGraph(mn_i, mn_j)
      When Sequential  ComputeSequentialGraph(mn_i, mn_j)
      When Branch      ComputeBranchGraph(mn_i, mn_j)
      When Loop        ComputeLoopGraph(mn_i, mn_j)
    End Case
```

6. SPEED - AREA - POWER MAPPING

At this step of the estimation flow, we know the number of functional units of each type in function of a varying time constraint. The area, delay and power consumption of each functional unit is characterized in the library component. Time execution values are obtained by mapping the clock period value with the time constraint defined in number of clock cycles. We compute the total area of the whole application by adding the contribution of each functional unit, knowing the area of each of them. Concerning power estimation, the temporal use of each functional unit is analyzed in order to determine the number of times they are used. Then, knowing the average power consumption by access (provided by the library) and the clock frequency makes total power consumption easy to compute.

7. RESULTS

The preliminary results presented here shows the estimation results of a filter based on a 12 iteration loop which body is a basic multiplication addition pattern. This example comes from a speech coding application (G.722 recommendation). The target FPGA is a Xilinx of the XC4000 family. Figure 4 presents the results of the processing unit area estimation compared to the synthesis results.

There is a slight difference between the estimated execution time and the results of synthesis, especially for low execution time values. This is due to the fact that at the moment interconnection delays are not taken into account and shows the great influence they have when the design area gets bigger.

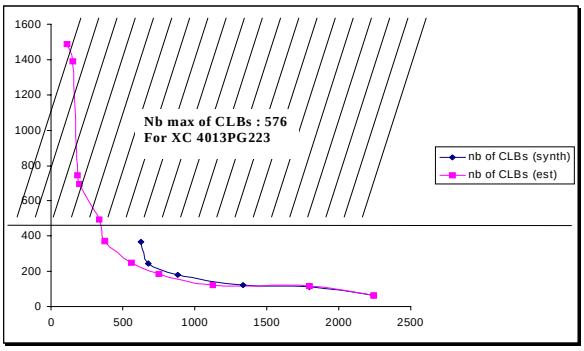

Fig 4 • Example of area estimation result

8. CONCLUSION

We have presented a global estimation technique leading to an exploration of the design space. Actually, the ATP estimation takes into account the processing nodes of the graph. The next step of our work is to extend the method to the estimation of the control and memory units, and also to analyze the interconnection overhead on the design characteristics in order to get accurate estimation values.

9. REFERENCES

[1] J.P. Diguet, "Estimation de complexité et Transformations d'Algorithmes de Traitement du Signal pour la Conception de Circuits VLSI", PhD Thesis, Université de Rennes 1, 1996.

[2] A. Sharma, R. Jain, "Estimating Architectural Resources and Performance for High-Level Synthesis Applications", IEEE Trans. on VLSI Systems, vol 1, No 2, June 1993.

[3] S. Narayan, D.D. Gajski, "Area and Performance Estimation from System-Level Specifications", Technical Report, University of California, 1992.

[4] A. Garcia, W. Burleson, J.L. Danger "Power Consumption Model of Field Programmable Gate Arrays (FPGAs)", FPL'99, Glasgow, Scotland.

[5] M. Xu, F.J. Kurdahi, "Area and Timing Estimation for Lookup Table Based FPGAs", ED&TC, March 1996.

[6] J.P. Diguet & al., "The SPF Model", FDL, Tübingen, Germany, sept 2000.

[7] G. Lenet, "Définition d'une bibliothèque d'opérateurs caractérisés en temps surface consommation sur FPGA", TR, LESTER, may 2000.

[8] D. Gajski, N. Dutt, A. Wu, S. Lin, "High-Level Synthesis", Kluwer Academic Publishers.

[9] H. Thomas, J.P. Diguet, J.L. Philippe, "A Methodology for an Application Profiling at a System Level", SiPS, Taïwan, oct 1999.