View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

FAST PROTOTYPING OF RECONFIGURABLE ARCHITECTURES FROM A C PROGRAM

S. Bilavarn, G. Gogniat, J.L. Philippe, L. Bossuet

LESTER - South Britany University - Lorient, France

ABSTRACT

Rapid evaluation and design space exploration at the algorithmic level are important issues in the design cycle. In this paper we propose an original area vs delay estimation methodology that targets reconfigurable architectures. Two main steps compose the estimation flow: i) the structural estimation which is technological independent and performs an automatic design space exploration and ii) the physical estimation which performs a technologic mapping to the target reconfigurable architecture. Experiments conducted on Xilinx (XC4000, Virtex) and Altera (Flex10K, Apex) components for a 2D DWT and a speech coder lead to an average error of about 10 % for temporal values and 18 % for area estimations.

1. INTRODUCTION

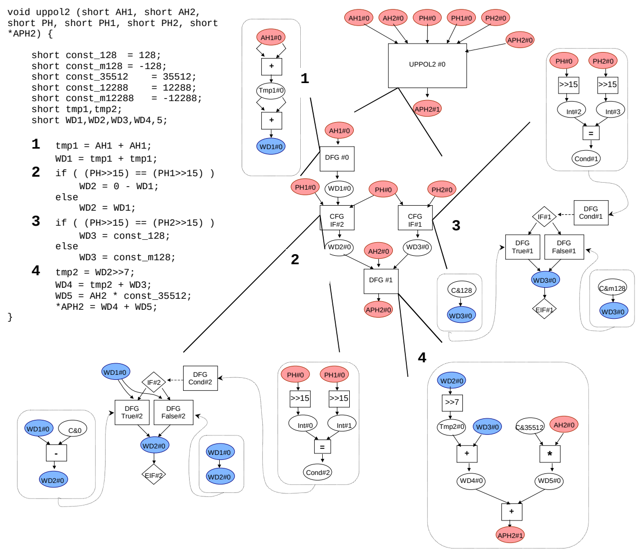
The evolution of telecommunication and multimedia applications towards new standards requires innovative architectures in order to respect always tighter constraints. The recent evolutions of reconfigurable architectures, in terms of capacity and performances, efficient resource integration (like DSP operators and memories), or flexibility through the possibility of run time reconfiguration, offer a very promising issue for reconfigurable system on chip. As a result, the choice of a suitable target component, satisfying both physical (area, performances, . . .) and marketing (final product cost, time to market, . . .) constraints is a complex issue often left to the designer experience. Dealing with such problems as application parallelism exploration and FPGA architecture matching, impose to define new design methodologies in order to find more quickly and surely an integration solution satisfying all the design constraints. Until now, typical hardware design methodologies start, from an algorithmic description of the application, with an architectural synthesis step to obtain a description at the RTL level. Then logic synthesis and place & route steps are performed to obtain the final description of the circuit and precise values of area (FPGA occupation) and performances (execution time). These two steps are very time consuming since the only architectural synthesis step can take from several hours (with a High Level Synthesis tool, HLS in the following) to several months (hand coding) to overcome. Furthermore design space exploration may need several iterations if constraints are not met, what can lead to prohibitive design times.

2. OBJECTIVES & CONTRIBUTION

The purpose of the work presented in this paper is to define an efficient exploration methodology starting from system level specifications that allows: i) to define several architectural solutions and ii) to compute the corresponding estimated area and execution time values. The second point allows the designer to make a choice of a solution, while the first point gives information for the selected solution design. To find an interesting answer to this problem, the following considerations have been addressed: i) Define a method operating at a high level of abstraction, from system level specification including control structures, multidimensional data and hierarchy to deal with complex modern applications. ii) Give realistic cost characterization: estimation takes into account all the different units of the architecture (datapath, control unit, memory unit). iii) The method should explore the application parallelism: several architectural solutions are defined for a given specification. iv) The method should be applicable to several FPGA families. v) Define feasible solutions and give sufficient information for post exploration steps (selected architecture design) and vi) low complexity to enable large design space exploration. The methodology developed can be seen as a global exploration / estimation technique based on the numerous existing works in the field of estimation and HLS (memory size estimation, scheduling techniques, data flow modelling, . . .). Compared to other estimation approaches, the definition of effective architectures have been emphasized: each solution is implementable and corresponds to a given resource allocation, clock period value and scheduling. Their definition relies on a precise architectural model (not only datapath, but also memories and control units) and takes care of modern FPGA architectural specificities. Due to the paper size restriction the complete description of the related work can be found in [1]. Compared to a typical design exploration flow, we do not need to make a complete and precise description of the circuit. For example, we do not need to go until the precise description of the connections between resources, or to build a floorplan. Those steps are only needed to be computed once in the design cycle and are left to the steps following the exploration process (synthesis / refinement / optimization). The reduced complexity allows then to explore quickly the effect of different implementation possibilities (intra loop parallelism exploration, resource allocation, clock period, evaluation of several target FPGAs). Obviously, the solutions defined may be sub-optimal in some cases, but they always correspond to implementable solutions. So estimation values computed (area and execution time) are more representative of the system's feasibility. Moreover, those metrics give a designer usefull information that allow to make an easiest choice for implementation (satisfying both area and execution time constraints). Once a solution selected, application synthesis and solution refinement / optimization can be performed in a classical way with the use of a HLS tool for example, thanks to the rich set of information given by the architecture definition step. This fast system level exploration allows then to evaluate many design possibilities very early in the design cycle, where choices have a great impact on the final system performances. The evaluation of several design possibilities allows moreover to converge more quickly and surely towards an optimal implementation solution.

3. EXPLORATION / ESTIMATION METHODOLOGY

First, the system level specification is given in a high level language (C language), and is then translated into an intermediate representation, the HCDFG model [1]. This model is a hierarchical control and data flow graph allowing efficient algorithm characterization and exploration of complex modern applications including control flow and multidimensional data. As illustrated in figure 1 a C program is decomposed into control structures called CFGs and into linear sequences of operations called DFGs. For example the If-Then-Else construct labeled 2 is composed of three DFGs, one for the evaluation of the condition and two for the True and False sequences of code. Hence, using the HCDFG model, the C program is converted into a hierachical graph. For further information about the HCDFG model please refer to [1]. Starting from this specification and given a target component, the architectural exploration methodology (figure 2) consists in defining several implementation solutions and estimating FPGA resource occupation and algorithm execution time. To perform this estimation, we need to know the target FPGA characteristics which are described in a technology file [1]. Moreover, to give realistic estimation values, we use a specific architectural model and take memory requirements into account (the total memory size needed is estimated). The Exploration / estimation flow is composed of two steps: i) structural estimations and ii) physical estimations. The first step is technological independent and performs architectural exploration based on the considered architectural model [1]. Each solution is characterized for a number of cycle budget N_c by the number $Nop_k(N_c)$ and the type (op_k) of resources required to execute the application for this cycle budget. By changing the number of cycle budget N_c, we explore the design space. Another important characteristic of the architecture is the number of simultaneous read and write accesses to a RAM and read accesses to a ROM which are also computed during the structural estimations. This exploration conducts to several architectural solutions that are characterized for a given cycle budget N_c: i) by the number and type of functional units $(N_{op_k}(N_c))$, ii)



```
void uppol2 (short AH1, short AH2,
short PH, short PH1, short PH2, short
*APH2) {

    short const_128  = 128;
    short const_m128 = -128;
    short const_35512    = 35512;
    short const_12288    = 12288;
    short const_m12288   = -12288;
    short tmp1,tmp2;
    short WD1,WD2,WD3,WD4,5;

    1   tmp1 = AH1 + AH1;
        WD1 = tmp1 + tmp1;
    2   if ( (PH>>15) == (PH1>>15) )
            WD2 = 0 - WD1;
        else
            WD2 = WD1;
    3   if ( (PH>>15) == (PH2>>15) )
            WD3 = const_128;
        else
            WD3 = const_m128;
    4   tmp2 = WD2>>7;
        WD4 = tmp2 + WD3;
        WD5 = AH2 * const_35512;
        *APH2 = WD4 + WD5;
}
```

Figure 1: C to HCDFG format

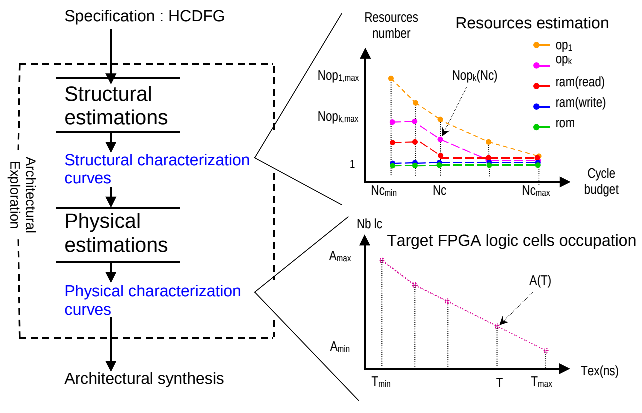


Figure 2: Exploration / estimation flow

by the number of simultaneous read(write) from(to) RAM ($N_{ram\_rd}(N_c)$, $N_{ram\_wr}(N_c)$), iii) by the number of simultaneous read from ROM ($N_{rom\_rd}(N_c)$) and iv) by the number of control states ($N_s(N_c)$). All these results are gathered together in a 2D representation, where the vertical axis corresponds to the number of resources and memory accesses, and the horizontal axis to the number of clock cycles. The second step is technological dependent and targets a specific FPGA technology. During that step, each architectural solution is characterized for a temporal constraint T by the FPGA resources occupation $A(T)$. FPGA resources considered are logic cells (resources that allow the configuration of user defined functions, e.g. slices, logic elements), dedicated cells (e.g. Block SelectRAM, Embedded System Block, those resources allow efficient implementation of specific functionalities like memories, product terms, DSP operators …), tristate buffers and I/O pads. The FPGA description is given in a technology file that contains: i) the characteristics of the target FPGA (number of logic and dedicated cells, I/O pads, tristate buffers), ii) the characteristics of each functional units (area and delay) and iii) the characteristics of the memories (number of bits per logic or dedicated cells, access time). Note that the technology file is derived from the data sheet of the target FPGA and from the synthesis of basic arithmetic and logic operators.

4. EXPERIMENTS & RESULTS

4.1. From specification to synthesis

In this section, the design cycle described above is applied to the example of a half Discrete Wavelet Transform (DWT). Specification is written in the C language for test and simulation, and is then translated into the intermediate representation model (HCDFG) on which the exploration / estimation tool works. The DWT application is composed of 4 filtering / lifting schemes followed by a scaling process and image re-arrange, described by 2^{nd} order nested loops. Figure 3 shows the exploration results for the Xilinx Virtex V400EPQ240-7. We have only represented the logic cells occupation (where the maximum number is 4000 *slices* for Virtex) vs excution time (ns) curves as they represent the most significant FPGA resource occupation for this example. As we can see on the figure, exploration provides 65 architectural solutions, each one corresponding to a different parallelism degree. Let's for example consider the solution highlighted in figures 3 since it corresponds to a good area/speed trade-off. Based on that solution the designer may want to refine the exploration. For example, in this experiment the default clock period value corresponds to the slowest functional unit delay used in the architecture. Hence, the designer can refine the exploration results obtained previously by analyzing the effect of different clock periods and resource allocation. For the solution selected

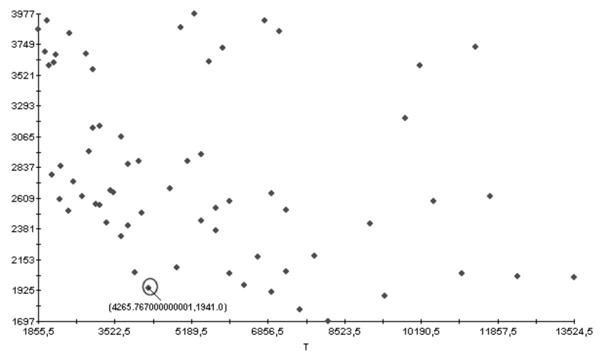


Figure 3: Horizontal DWT exploration results (Virtex) - *slices* vs time

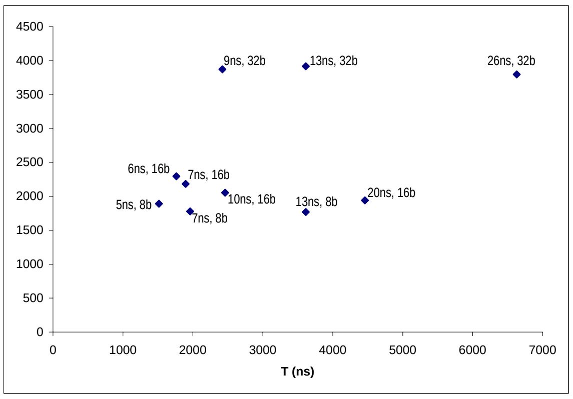


Figure 4: Bitwidth and clock period exploration

before, several clock values and data bitwidths are estimated in figure 4 (labels correspond to a couple clock period value - data bitwidth). Thanks to those information, the designer can quickly evaluate if a solution defined by a parallelism degree, clock value, resource allocation and target FPGA, can meet the design constraints or not. Once a solution have been selected (for example the one with a clock period value equal to 20 ns and a bitwidth equal to 16), details and corresponding structural estimation results for each hierarchy level (each subgraph of the specification) are also available in our tool (table 1). Those partial results fully characterize each architectural solution and give the designer all the necessary information needed for the system design. In the case of our example, we can see that the selected solution is composed of 4 multipliers and 8 adders for a 223 cycles execution, which correspond to a resource occupation of 1941 ($/4000$) *slices*, 12 ($/40$) BRAMs (dedicated resources for memory implementation) and 256 ($/4960$) tristate buffers (used in case of resource sharing or conditional branches) for a $4.5\mu s$ execution time.

GRAPH	Cycles	States	Mul16	Add16	Reg16	Ram(wr)	Ram(rd)	Rom
For12_body	5	5	1	2	--	1	3	1
H1stLftStep	32	32	4	8	--	4	12	4
For22_body	5	5	1	2	--	1	3	1
H1stDLftStep	32	32	4	8	--	4	12	4
For32_body	5	5	1	2	--	1	3	1
H2ndLftStep	32	32	4	8	--	4	12	4
For42_body	5	5	1	2	--	1	3	1
H2ndDLftStep	32	32	4	8	--	4	12	4
For52_body	3	3	2	--	--	2	2	2
Hscaling	66	66	4	--	--	4	4	4
For62_body	2	2	--	--	--	2	2	--
Hreaarange	33	33	--	--	--	8	8	--
Hdwt	223	223	4	8	28	8	12	4
	T_{ex} : 4.5 µs		Slices : 1941		BRAM : 12		3 state : 256	

Table 1: Selected solution details

4.2. Precision & Exploration time

In this section, we discuss the precision of the occupation vs execution time estimations and give values of the exploration time vs logic synthesis time needed. These measures have been performed with two representative of recent FPGA families (Virtex and Apex) for a speech coder (G722) and a 2D DWT (table 2). The corresponding architectures have been synthesized in order to study estimation values precision. Note here that to study this precision, the architectures have been hand coded at the RTL level in order to cope exactly with our architectural model (the use of a HLS tool would have lead to significant estimation errors as it does not generate the same architecture). That's the reason why in the following, exploration times are only compared to the *logic synthesis* times (architectural synthesis times are about hours for a HLS tool and months for hand coding). The Foundation and Quartus synthesis tools have been used to target respectively Virtex and Apex FPGA. The speech coder application is composed of eight functions that correspond for example to filtering and prediction operations. These functions are mainly control and computation oriented. The 2D DWT example is characterized by numerous memory accesses and computations. The average error is about 10 % for temporal values and 18 % for area estimations which represent a good bound for the designer since the application is described at the algorithmic level. Locally more important errors can be noticed which are due: i) to logic optimizations automaticaly performed by the synthesis tools which are not taken into account in our approach or ii) to the considered control unit architectural model that has the charge of setting the address signals. This is particularly true for the 2D DWT example where numerous memory accesses are performed. The exploration / estimation computational time is very fast since in the case of the G722, 16 solutions are estimated in about 1 second and in the case of the 2D DWT, 350 solutions are estimated in 5 minutes on a Pentium III running at 800 MHz. In table 2, solutions for both applications have been manually written at the RTL level and then logic synthesis and place & route steps have been done automatically. As exhibited in the figure, the exploration / estimation approach enables to reduce strongly the design cycle. Hence, the designer can focus on a subset of architectural solutions that presents the best delay vs area trade-offs.

EXAMPLE	Virtex V400EPQ240-7				Apex EP20K200EFC484-2X			
	Precision (%)		Expl vs lgc synth		Precision (%)		Expl vs lgc synth	
	slices	T_{ex}	T_{expl}	T_{synth}	lgc elt	T_{ex}	T_{expl}	T_{synth}
Parrec	-10	+1.4	0.05 sec	1 min	-10.5	+4.9	0.05 sec	1 min
Recons	-10	+1.1	0.05 sec	1 min	-10.5	+4.9	0.05 sec	1 min
Upzero	-14.9	+4.5	0.22 sec	5 min	+52	+18.3	0.06 sec	5 min
Uppol2	-15.2	-2.7	0.11 sec	5 min	+19.4	+18.6	0.11 sec	5 min
Uppol1	-21.5	-9.8	0.11 sec	5 min	-3.8	+19.6	0.11 sec	5 min
Filtep	-8	-13.1	0.05 sec	1 min	-20.1	-8.4	0.05 sec	1 min
Filtez	+2.2	+16.1	0.05 sec	2 min	-2.6	+41.4	0.05 sec	2 min
predic	-10	-2.2	0.05 sec	1 min	-10.5	+4.9	0.06 sec	1 min
G722Predictor	-7.7	-7	0.9 sec	15 min	+3.4	+14.1	0.4	10 min
1stHLftStep	+6.9	+7.1	0.1 sec	5 min	+1.4	+7.6	0.05 sec	8 min
1stHDLftStep	+4	+0.6	0.05 sec	5 min	+2.6	+1.9	0.06 sec	8 min
2ndHLftStep	+5.1	+13.9	0.06 sec	5 min	+2.8	+9.3	0.05 sec	8 min
2ndDHLftStep	+2.5	+9.8	0.06 sec	5 min	-0.2	+1.7	0.05 sec	8 min
Hscaling	+2.7	+3.6	0.1 sec	5 min	+4.9	+3.6	0.1 sec	8 min
Hrearrange	+46.8	-25	0.06 sec	5 min	+67	+9.1	0.05 sec	8 min
1stVLftStep	+7.1	+25.5	0.1 sec	5 min	-0.2	+2.9	0.05 sec	8 min
1stVDLftStep	+5	+16.9	0.05 sec	5 min	-0.6	+5.1	0.06 sec	8 min
2ndVLftStep	+5.1	+18.5	0.06 sec	5 min	+1.1	+2.9	0.05 sec	8 min
2ndDVLftStep	+3.4	+18.3	0.06 sec	5 min	-2.6	+7.7	0.05 sec	8 min
Vscaling	+3.4	+5.5	0.1 sec	5 min	+3.2	+3.8	0.1 sec	8 min
Vrearrange	+50.9	-5.5	0.06 sec	5 min	+61	+3.8	0.05 sec	8 min
DWT 2D	+35.9	+18.2	5 min	1.5 days	+37	+3.1	5 min	2 days

Table 2: Estimation vs Synthesis error and Exploration vs (logic) Synthesis time

5. CONCLUSION & PERSPECTIVES

In this paper we present an automatic exploration / estimation methodology at the algorithmic level. This approach, which has been integrated in the codesign environment *Design Trotter* [2], enables to explore a large design space at an early stage of the design cycle and to characterize each solution in terms of area vs delay. In order to provide the designer useful bounds, the control, datapath and memory units are considered and several FPGA technologies can be targeted. The time saving resulting from this approach is significant and allows to shorten strongly the time to market constraints as well as to converge towards a better application / component matching. Some extensions of this work are currently being studied to consider a separated address generation unit, to take into account some synthesis optimizations to improve local errors and to include power consumption estimation.

6. REFERENCES

[1] S. Bilavarn, *Architectural Exploration from System Level Specification - Application to FPGAs*, PhD, University of South Britanny, Feb 2002.

[2] Y. Moullec, J.P. Diguet and J.L. Philippe, *Design-Trotter: a Multimedia Embedded Systems Design Space Exploration Tool*, IEEE MMSP02 Dec. 9-11, 2002, US Virgin Islands.