RAPID AREA-TIME ESTIMATION TECHNIQUE FOR PORTING C-BASED APPLICATIONS ONTO FPGA PLATFORMS

MY CHUONG LIEU, SIEW KEI LAM, THAMBIPILLAI SRIKANTHAN
CENTER FOR HIGH PERFORMANCE EMBEDDED SYSTEM
NAYANG TECHNOLOGICAL UNIVERSITY, SINGAPORE
{LIEU0003, ASSKLAM, ASTSRIKAN@NTU.EDU.SG}*

Abstract. High-level area-time estimation is an essential step to facilitate rapid design exploration for FPGA implementations. Existing works in high-level area-time estimation usually ignore the physical effects of the design after place and route, which have a notable impact on the maximum achievable speed of the design. In this paper, we propose a framework to rapidly estimate the area-time measures of mapping C-applications onto FPGA. The framework relies on the Trimaran compiler to generate an optimized high-level IR (Intermediate Representation) of the C-applications. Area-time estimation of the IR is then performed using a proposed estimation model that is based on an architecture template with application-specific heterogeneous functional units. In order to accurately predict the delay of the design after place and route, we introduce a new metric for the estimation that models the criticality of the design's interconnectivity. Experimental results based on a set of embedded functions show that the proposed area estimation can achieve comparable results with the synthesis results of a commercial FPGA tool in the order of milliseconds. For the C functions used in our experiments, the proposed delay estimation leads to an average error of about 3% when compared to the post place and route results. In addition, we demonstrate the robustness of the proposed framework which provides consistent results for different FPGA families. The contribution of this paper is a scalable methodology for rapid estimation of cost-benefit metrics of C-based algorithms to be accelerated on FPGA-based high-performance computing platform.

Key words. FPGA, C-based Application, Area Time Estimation, Hardware Accelerator

1. Introduction. FPGAs (Field-Programmable Gate Arrays) have become an attractive solution to meet the technological and market challenges in embedded processing. Traditional hybrid platforms that incorporate ASIC and microprocessors are migrating towards FPGA platforms (e.g. Xilinx Virtex-II Pro [1] and Altera Stratix [2]) to take advantage of the reconfigurable benefits of FPGA. This trend is supported by the availability of efficient EDA (Electronic Design Automation) tools and the increasingly stringent TTM (Time-To-market) requirements. In order to exploit the strengths in both the microprocessor and FPGA, efficient hardware-software partitioning strategies must be incorporated in the emerging design flows. However, commercially available design flows do not enable designers to make design explorations for effective hardware-software partitioning. This is chiefly due to a lack of an essential step that can estimate the performance-cost for mapping a software component to hardware early in the design cycle.

In this paper, we propose a framework that can rapidly and accurately estimate the hardware area-time measures for implementing C-applications onto the FPGA. We have chosen C as the input to our framework as it is widely used for embedded processing. The front-end of the framework relies on the high-level optimization and scheduling capabilities of the Trimaran compiler infrastructure [3]. In order to facilitate effective area-time estimations, we have adopted an architecture template for implementing the applications, which is similar to the one proposed in [4]. The architecture template resembles a VLIW-like architecture that incorporates application-specific heterogeneous functional units and register-files, with dedicated interconnection buses.

High-level estimation is performed using an area-time estimation model, which relies on a set of pre-characterized parameters of the components in the architecture template. Previously reported works in high-level area-time estimation often do not consider the interconnect delay of the design after place and route. We will demonstrate that this oversight will lead to high uncertainties in the estimation results. Our proposed approach overcomes this limitation by incorporating a new metric that models the placement complexity of the design's interconnectivity.

The paper is organized as follows. In the following section, we describe related works in the area of high-level estimation for FPGA implementation. This is followed by an overview of the proposed framework. Section 4 describes the parametric characterization of the architecture template components, and the proposed area-time estimation models. Next, results analysis is provided to demonstrate the benefits of our framework, and we conclude in Section 6.

2. Related Works. Due to the need to expedite the development of complex applications in hardware, a number of commercial tools that synthesizes high-level languages to FPGA have emerged in recent years. These tools differ in several aspects such as high-level language support, high-level optimization features and the target system. For example, Mitrion-C from Mitronics [5] and RCToolbox from DSPLogic [7] supports the Mitrion-C and Matlab programming language respectively, while HandleC from Celoxica [6] and Impulse-C from Impulse Accelerated Technologies [8] support a subset of the ANSI-C language that is extended with constructs for specifying the hardware definitions. These tools cannot be directly employed for most embedded applications that are programmed using ANSI-C. Although the C2H tool from Altera [9] supports ANSI-C applications, the hardware representations that are generated are specific to the Altera FPGA devices only. This limits the generality of the tool across different platforms. Other tools such as Catapult from Mentor Graphics [11] and Trident [10] support ANSI applications and are not device specific. However to the best of our knowledge, all the tools discussed above do not focus on providing accurate high-level estimations to facilitate rapid design exploration for hardware-software partitioning.

The problem of high-level area-time estimation for hardware implementations has received considerable interest in the research community for nearly 20 years. Research efforts in this area are motivated by the need to evaluate the hardware performance-cost indices of various design options early in the design phase, in order to reduce the time-consuming implementation cycles. Figure 1 highlights the major steps in a typical high-level estimation flow, which are 1) Transformation of application written in high-level language to IR; 2) Architecture independent estimation; 3) Architecture independent synthesis; 4) Architecture dependent estimation. It is worth noting that previously reported works typically do not address all these steps in their area-time estimation approach. The first step in the estimation flow typically involves the transformation of a high-level representation of an application (e.g. C, System-C, Matlab, behavioral HDL or JAVA specifications) to an IR (Intermediate Representation). This transformation includes high-level compiler optimizations such as loop transformations to extract the hidden parallelism in the sequential C statements. For example, the SUIF [12] compilers have been widely used to transform C-based application into CFGs (Control Flow Graphs) and DFGs (Data Flow Graphs).

Architecture independent estimation attempts to calculate the hardware resources and latency in terms of clock cycles without performing scheduling. These approaches commonly rely on probabilities or integer linear programming models based on the
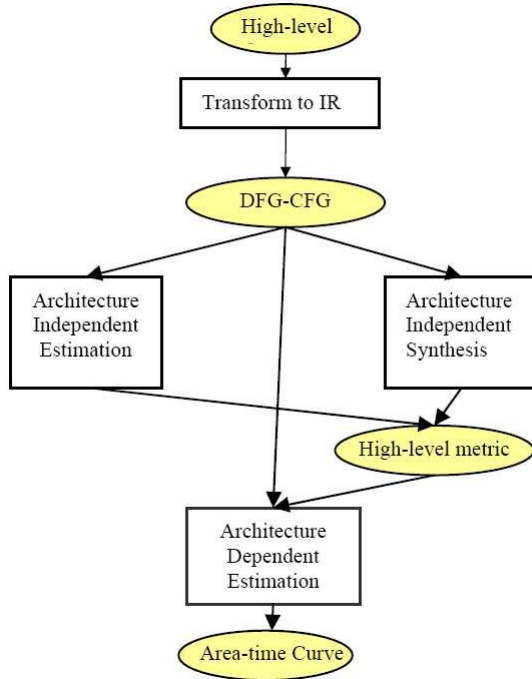
Fɪɢ. 2.1. *High-Level Estimation FLow*

high-level application characteristics in order to predict the number of functional-units or minimum clock cycles. The work presented in [13] performs architecture independent estimation to obtain a lower-bound execution time of a DFG in the presence of hardware constraints to facilitate efficient design space exploration. Similarly, [14] estimates the minimum number of resources that are required to execute a DFG within a control step constraint. Others [15] employ Matlab codes as inputs to estimate the hardware resources by summing the area of the required operators based on the execution probabilities in the application.

Architecture independent synthesis typically performs scheduling of the IR and resource binding to obtain accurate high-level metrics, which include the number of clock cycles and hardware resources. The hardware area-time of the application is then calculated or estimated from these metrics. The architecture independent synthesis approach in [16] considers the effects of various loop transformation techniques. Bilavarn et al [17] presented a method that employs architecture independent synthesis for design exploration. However, maximum clock frequency estimation was based solely on the longest latency of the execution unit and ignores post place and route physical effects. Cardoso proposed a methodology for estimating FPGA implementations of Java byte-codes in [18]. He highlighted the limitations of high-level delay estimation due to the lack of circuit details.

Architecture dependent estimation techniques commonly employ simple hardware models to speed-up the estimation process. The hardware cost is estimated in terms of LUTs (Look-Up Tables), while the performance is often estimated in terms of clock latencies. The technique presented in [19] adopts an analytical approach to

estimate the FPGA area for implementing the DFG. The estimation is based on a set of formulas that models the components and corresponding hardware area of the DFG operations. Their approach reported credible results with a maximum error of 10% and the estimation can be achieved in the order of milliseconds. However, they have not considered delay estimation. In [15] , the number of required flip-flops is estimated by calculating the maximum number of required registers. The Rent Rule and Feuer's formula have been employed to estimate the post place and route interconnect delay. This approach leads to large uncertainty of up to 9 ns. This high uncertainty can become unacceptable for designs that need to be clocked at high frequency (e.g. 100 MHz). It is noteworthy that our proposed area-time estimation accounts for the physical implementation characteristics and is not susceptible to the speed of the design.

3. Overview of Framework. Figure 2 describes an overview of the proposed framework for high-level area-time estimation. The open-source Trimaran compiler infrastructure, which supports state of the art compiler research in ILP (Instruction Level Parallelism) based architectures, is relied upon to expose the hidden parallelism in the sequential C statements, and to perform high-level optimizations and scheduling [3]. This front-end process takes several seconds (typically less than 10s for 1 single C-function compilation) depending on applications and functions sizes. We have adopted the application-specific architecture template that is similar to the one proposed in [4] as shown in Figure 3.2. It is worth mentioning that this architecture template can be adapted for pipelined and non-pipelined data-paths by configuring the application-specific interconnection. The Trimaran machine description is augmented with the heterogeneous functional units in the architecture template. These functional units include a combination of two or more basic operators (e.g. adder, shifter, multiplier, logic operator, comparator and memory-access unit). Only the functional units that are required for a particular application will be incorporated into the architecture.
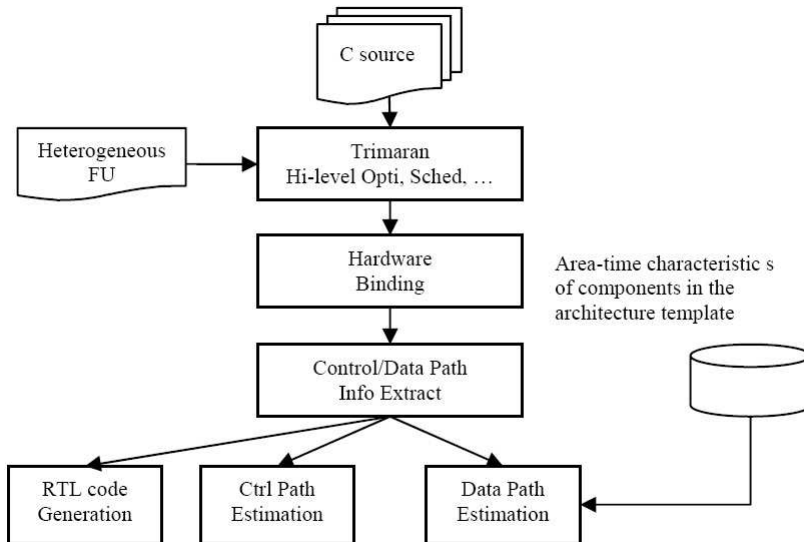

FIG. 3.1. *Propose High-level Estimation Framework*

4

The output of the Trimaran is an ILP schedule of the application (e.g. the type of functional units that will be executed in each clock cycle and the data-dependency between these functional units). Based on this schedule information, we perform a simple hardware binding process that attempts to bind operations with the most common input-outputs to the same functional units. This aims to reduce the complexity of the interconnectivity between registers and functional units. In order to perform area-time estimation, information pertaining to the control-path and data-path are segregated from the ILP schedule after hardware binding. In this paper, we focus on area-time estimation for the data-path only.
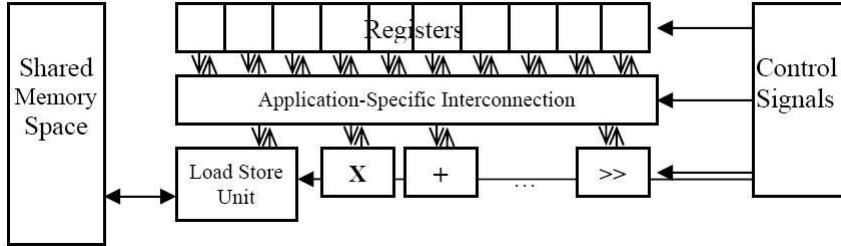

FIG. 3.2. *VLIW-like architechture Template*

A one-time area-time characterization of the components in the architecture template is required to facilitate area-time estimation of the data-path. An estimation model is then employed along with this information to estimate the performance-cost measures of the application by taking into account the physical implementation effects. In order to evaluate the accuracy of our estimation approach, a process to auto-generate the RTL (Register Transfer Level) codes from the control-data path information has been incorporated in the framework. The RTL code can then be subjected to the FPGA implementation tool (i.e. Xilinx ISE) to obtain the actual post place and route report for results comparison with the proposed estimation approach.

4. Area-Time Estimation. In this section, we will provide detailed description of the process to characterize the components and the proposed area-time estimation model.

4.1. Hardware Characterization of Architecture Template's Components. We have used the Xilinx ISE synthesis engine to characterize the hardware components in the architecture template and other relevant information. Table 1 illustrates the hardware components (and other relevant information) and the corresponding area-time measures for the Virtex-II Pro device (xc2vp70-6ff1704). The data-paths are assumed to be 16-bit or 32-bit, as the hardware implementation serves to accelerate the base ISA (Instruction Set Architecture) operations of the microprocessor.

In order to perform interconnect characterization, we have implemented a number of circuits to obtain the average post place and route interconnect delay. The circuits are constructed based on the data-path that is shown in Figure 4, which resembles the architecture template consisting of a single functional unit. A range of designs, each consisting of up to 8 duplicate circuits similar to the one in Figure 4, is subjected to physical implementation using the Xilinx ISE tool. We utilized the Xplore Script provided by Xilinx [20] that iteratively executes the place-and-route process to achieve the maximum clock speed. Figure 5 shows the maximum delay for the range of

5

TABLE 4.1
Characterized Components for Xilinx FPGA Virtex2p-6

Components	Area(LUT)	Delay (ns)
16bit-Addsub	16	2.139
16-bit multiplier by LUT	121	8.165
16-bit left/right/signed-unsigned shifter	76	2.376
16-bit Logic Operator (4 operations)	16	0.313
16-bit Multiplexer 2 to 1	16	0.313
16-bit Multiplexer 4 to 1	32	0.653
16-bit Multiplexer 8 to 1	64	0.972
16-bit Multiplexer 16 to 1	128	1.291
FFCLK2Q (Clock to output delay of FF)		0.234
FFSetup (Setup time of FF)		0.243

designs from which we calculated the average delay after place and route. The average interconnect delay is then computed using Equation 4.1, where Logic Delay is the sum of Clk2q, Mux4to1, functional unit delay, and FF-Set-up time that are listed in Table 1. The average interconnect-delay for the target device is found to be 0.42 ns, and this value will be used by the proposed method for delay estimation.

$$(4.1) \qquad InterconnectDelay = (Avg(MaxDelay) - LogicDelay)/3$$

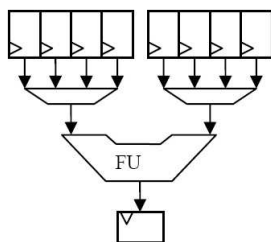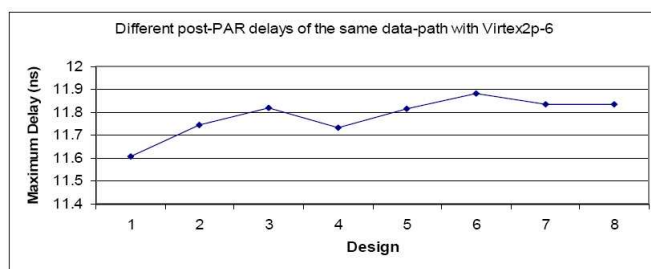Fig. 4.1. *Sample Circuit for Characterizing Average Interconnect Delay*

Fig. 4.2. *Maximum post place and route delays for eight designs comprising of multiple of sample circuits (see Figure 4.1)*

It is noteworthy that the proposed high-level estimation approach can also be adopted for different target FPGA families by performing a one-time hardware characterization for the particular device.

4.2. Area Estimation. In order to perform area estimation, we obtained the number of functional units and the number of registers (flip-flops) from the ILP schedule after hardware binding. The estimated number of LUTs is computed by summing up the number of LUTs for the functional units based on the pre-characterized information. As each slice of Xilinx Virtex2 pro contains 2 Look-up tables (LUT) and 2 flip-flops, we estimate the total slices as in 4.2:

$$(4.2) \qquad Estimated\_number\_of\_slices = (\#LUT + \#FF)/2$$

The estimation for the number of slices assumes that each slice is fully utilized to implement the functional units and registers. Experimental results reveal that the proposed estimation method is very accurate for LUT and Flip-Flop estimation. In addition, the estimation of the slices is comparable with ISE logic synthesis results and results reported in previous works [15].

4.3. Delay Estimation. The difficulty in delay estimation lies in the prediction of the interconnect delay before the physical design steps (i.e. placement and routing). There have been several reported works in the area of interconnect delay estimation such as [21] [22] and we will briefly discuss them before describing the proposed delay estimation approach. It is worth mentioning that these previous works are not integrated as part of a high-level estimation framework, but are used mainly to aid optimization decisions in the CAD flow. The work presented in [21] can achieve very accurate estimations of the interconnect delay by analyzing the physical characteristics of the designs. However, due to the complexity of the approach, the estimation results are achieved in the order of seconds and minutes. Karnik and Kang [23] presented an empirical routing delay model for estimating interconnection delays in FPGA. These methods require low-level metrics of circuit such as net fan-out and routing congestion which is not desirable for efficient high-level estimation [18]. Their method resulted in an estimated delay with 20% errors. Hutton highlighted that that delay estimation based on theoretical models, generally produces inferior results when compared to those computed based on empirical data [24]. Manohararaja et. al. reported an interesting finding that the predictability of FPGA implementation is mainly governed by the placement rather than routing process [25].

The physical characteristics that has been commonly used for post place and route delay estimation includes: 1) Design size [23] [15], 2) Circuit shape [26], and 3) Fan-in/out [23]. The placement and routing effort is mainly influenced by the interconnectivity in the designs rather than the size or shape of the design due to the fine-grained architecture of FPGA. Hence the first two characteristics is not a reliable indicator on the complexity of the place and route process. We have carried out experiments to show that these characteristics do not lead to reliable predictability of the post place and route delay. The fan-in/out of a register is defined as the number of input/output connections of that register. Compared to the first two characteristics, the fan-in/out provides a better indication on the interconnect complexity of the design. However, during place and route, the CAD tools often perform register duplication to mitigate the fan in/out effect. Our experiments show that although register-duplication can lead to improved timing in some cases, it could also increase the routing congestion of the circuit. This was inferred in our experiments for some designs, whereby the final delay after register duplication is higher than the delay obtained from implementations that obviates register duplication. Due to this uncertainty, we have assumed that the applications employed in our experiments have

moderate register reusability and hence, we do not incorporate the fan-in/out characteristic in our delay estimation model.

In this section, we introduce a delay estimation model that takes into account the post place and route characteristics of the design. Our proposed delay estimation model incorporates a new metric that is based on the relative path delays of the design. In contrast to the method in [21], our proposed method can estimate at a higher abstraction level and achieve reasonable results in less than a second. In addition, we will demonstrate that the maximum estimation error of the proposed model is less than 8% for the experiments considered.

4.3.1. Proposed Approach. A path is a connection of a sequence of logic units that begins and ends at a register, as shown in 4.3. Let's define Dpath(i) as the delay of a path i in a RTL design, Dmax as the critical path of that design, and Dmean as the mean delay of the paths in the design. Calculations for Dpath Dmax, and Dmean are defined below, which constitutes to a simple delay estimation model. If place and route effects are ignored, the minimum clock period is approximate as Dmax. We compared the estimated delay of 45 random algorithms using the simple delay model with actual results after place and route for the Xilinx Spartan and Virtex-II Pro device. The estimation error shown in Figure 7 exhibits a consistent error pattern for the two devices. This serves as the motivation for us to use the simple delay model as a basis for post place and route estimation as it can be employed across different devices. The consistency of the delay predictability on the two different devices implies that the inherent characteristics of the design can be used for estimation. The delay of path i (Dpath(i)) for all the paths in the circuit is computed as follows:

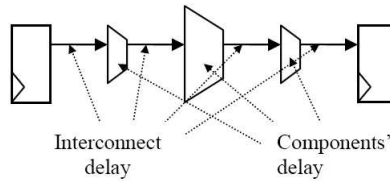Fig. 4.3. *A typical delay path from register to register*

Dpath(i)= FFClk2q + Dint + Dcomp+ FFsetup
Dmax = Max of all Dpath
Dmean = Mean of all Dpath
Dcomp is the characterized delay of the component in Table 4.1 (Usually: mux, Functional Unit, mux)
Dint is the characterized interconnect delay between the components.

As mentioned earlier, our proposed delay estimation model incorporates a new metric that is based on the relative path delays of the design. It has been previously reported that the placement process plays a more important role on the predictability of the final delay (assuming that there is no constraints on the number of FPGA routing resources)[25]. In addition, timing-driven placement relies on the criticality of the nets, and hence the effect of the nets criticality can lead to reliable predictability of the interconnection delay. Based on this, we introduce the lambda metric to compute the relative lengths of nets which capture the complexity of the placement effort for a particular design:
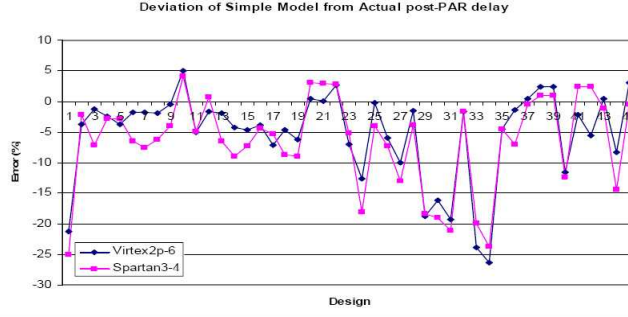
FIG. 4.4. *Estimation error of critical paths for 2 FPGA families*

$$(4.3) \qquad \lambda = \frac{D_{mean}}{D_{max}}$$

λ captures the slack distribution of the nets in the designs. If a circuit has one net that is much longer than the rest (i.e. low-λ), the CAD tool will require lesser effort to place the shorter paths such that they do not exceed the delay of the longest path. On the other hand for a circuit with high-λ, the CAD tool will have less freedom to move the paths around without violating the delay of the longest path. Hence, we expect designs with low-λ design to be more predictable than designs with high-λ. Our strategy is to identify through empirical means, the threshold value that will categorize a design as low-λ or high-λ designs. Let's define this threshold value as Λ. The estimated delay of a design with low-λ is computed as the maximum path length, while a design with high-λ is computed by multiplying the mean path length with a constant factor. The constant factor was empirically found to be close to $1/\Lambda$. The following describes our proposed delay estimation model, where D_{est} is the estimated delay.

$$(4.4) \qquad D_{est} = \begin{cases} D_{max} & \text{if } \lambda \leq \Lambda \\ D_{mean} \times \frac{1}{\Lambda} & \text{otherwise} \end{cases}$$
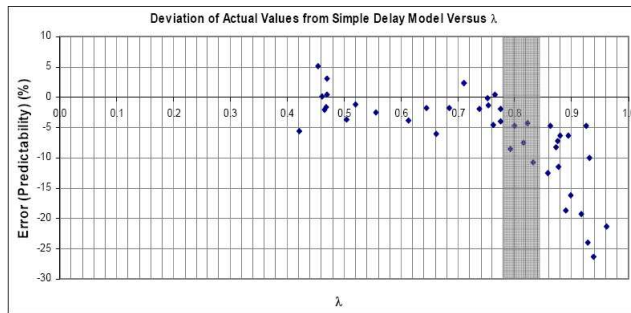

FIG. 4.5. *Estimation Error (Predictability) versus λ*

4.3.2. Determining Value of Λ. We compared the estimated delay using the simple delay model with the actual place and route results for 45 random algorithms.

9

Table 5.1
Connectivity Characteristics of Sample Circuits

Funtions	D_{Mean} (ns)	λ	Max Fanout	Avg Fanout	Max Fanin	Ave Fanin
Random Algorithm[1]	8.66	0.76	6	1.66	2	0.89
Matrix multiplier[2]	4.25	0.44	2	1.06	1	0.59
mpeg2-_bdist1_motion	4.82	0.46	13	2.13	4	1.22
mpeg2-_bdist2_motion	4.84	0.47	13	2.07	4	1.20
mpeg2-_dct_type_estimation	4.87	0.51	5	1.43	2	0.81
mpeg2-_dist1_motion	5.73	0.88	11	2.43	5	1.08
mpeg2-_dist2_motion	5.38	0.50	19	3.25	9	1.92
mpeg2-_idctcol	5.63	0.53	15	1.70	2	0.91
mpeg2-_idctrow	5.88	0.55	15	1.70	2	0.86
sha_transform	4.94	0.84	14	2.60	7	1.65
adpcm_coder	4.84	0.73	10	2.54	8	1.35
adpcm_decoder	4.64	0.67	9	2.16	9	1.22

4.5 shows the estimation error and the corresponding λ of the designs. It can be observed that there exist a high-correlation between the predictability and λ. In particular, it is shown that for low λ, the simple delay model can be applied with about 90% confidence. Large errors or low predictability are found in region where λ is high. We empirically define Λ to be 0.78 from the dataset.

5. Result Analysis. Table 5.1 describes the properties of the C functions that have been used to evaluate the proposed framework for high level area-time estimation. These applications (apart from the random algorithm) are commonly used in embedded applications.

5.1. Area Estimation. Table 5.2 compares the proposed area estimation (Pro) with results obtained from the Xilinx ISE tool after synthesis (Syn) and after place and route (PAR). The last four columns show the estimation errors of our method and that of logic synthesis, when compared with the post place and route values. It is evident that the proposed area estimation achieves up to an average of 98% accuracy, with a worst case error of 8% in terms of LUT comparison. For the estimation of slices, the average error of the proposed method is 12%. It is noteworthy that area estimation in terms of FPGA slices is a difficult task and the majority of the previous works reported their estimation results in terms of LUTs and flip-flops [27] [18] [19]. In general, our proposed area estimation for both LUTs and slices is comparable to the results of the logic synthesis tool. In addition, the proposed estimation can be completed in order of milliseconds, while the compilation time of the commercial synthesis tool takes several minutes (because of the level of abstraction).

5.2. Delay Estimation. The maximum delay of the above mentioned C-functions were estimated using the simple delay model (Sim) and the proposed model (Pro), and compared with actual post place and route values. In addition, we have used the ISE Xilinx tool to synthesize and implement the generated RTL codes in order to obtain the estimated delay after synthesis (Syn) and the actual delay after place and route (PAR). The results show that the proposed approach (Pro) outperforms the simple delay model and the synthesis tool for estimating the post place and route

---

[1]A dummy algorithm which has high parallelism. It computes 1 output from 4 inputs through several operations

[2]Only inner-most loop is considered

TABLE 5.2
Area Estimation Result

	LUT Estimation			Slices Estimation			LUT Error		Slice Error	
	Pro (LUTs)	Syn (LUTs)	PAR (LUTs)	Pro (Slices)	Syn (Slices)	PAR (Slices)	Pro %	Syn %	Pro %	Syn %
Random Algorithm	1636	1586	1,574	1170	1268	1014	3.94	0.76	15.44	25.06
Matrix multiplier	285	284	278	270	312	264	2.52	2.16	2.47	18.22
mpeg2-_bdist1_motion	1898	1959	1,939	1421	1632	1523	-2.11	1.03	6.67	7.16
mpeg2-_bdist2_motion	1971	1962	1,923	1465	1638	1843	2.50	2.03	20.49	11.13
mpeg2-_dct_type_estimation	623	624	616	471	542	482	1.14	1.30	2.28	12.45
mpeg2-_dist1_motion	2200	2183	2,165	1580	1705	1627	1.62	0.83	2.89	4.79
mpeg2-_dist2_motion	2625	2627	2,586	1688	1872	2022	1.51	1.59	16.50	7.42
mpeg2-_idctcol	4646	4531	4,501	3451	3851	3590	3.22	0.67	3.86	7.27
mpeg2-_idctrow	2942	2929	2,899	2271	2530	2229	1.48	1.03	1.91	13.51
sha_transform	2548	2691	2,670	1770	1993	2208	-4.57	0.79	19.82	9.74
adpcm_coder	1380	1399	1,367	978	1093	1308	0.95	2.34	25.20	16.44
adpcm_decoder	1496	1409	1,391	998	1118	1406	7.55	1.29	28.99	20.49
Average Error							1.64	1.32	12.21	12.81

TABLE 5.3
Delay Estimation Result

	Absolute Values				Error Compared to PAR		
	Sim (ns)	Syn (ns)	Pro. (ns)	PAR (ns)	Sim %	Syn %	Pro %
Random Algorithm	11.47	11.22	11.47	11.13	3.07	0.75	3.07
Matrix multiplier	9.61	9.62	9.61	9.46	1.56	1.65	1.56
mpeg2-_bdist1_motion	10.37	10.41	10.37	9.95	4.21	4.61	4.21
mpeg2-_bdist2_motion	10.37	11.54	10.37	10.61	2.27	8.77	2.27
mpeg2-_dct_type_estimation	9.61	9.74	9.61	9.91	3.00	1.65	3.00
mpeg2-_dist1_motion	6.55	6.65	7.35	7.70	15.00	13.69	4.55
mpeg2-_dist2_motion	10.71	10.97	10.71	10.42	2.75	5.20	2.75
mpeg2-_idctcol	10.71	11.22	10.71	10.94	2.13	2.49	2.13
mpeg2-_idctrow	10.71	11.22	10.71	11.10	3.49	1.06	3.49
sha_transform	5.83	5.98	6.33	6.22	6.26	3.88	1.86
adpcm_coder	6.62	6.45	6.62	6.37	3.86	1.32	3.86
adpcm_decoder	6.93	6.81	6.93	6.91	0.39	1.42	0.39
Average Error					4.00	3.87	2.76

delay. In particular, the proposed delay estimation achieves better results in terms of both maximum and average accuracy. The maximum and average estimation error of the proposed approach is only 4.6% and 2.8% respectively. It can be observed from Figure 9 that the simple delay model can provide accurate estimation results for designs with low λ cases. However, in designs with high-λ (i.e. 6 and 10), the simple delay model and the synthesis tool (Syn) incurs very high estimation error (i.e. up to 14%). In these cases, the proposed estimation approach is capable of providing significantly better accuracy due to the inclusion of the new metric λ that can be easily obtained for high-level estimation.
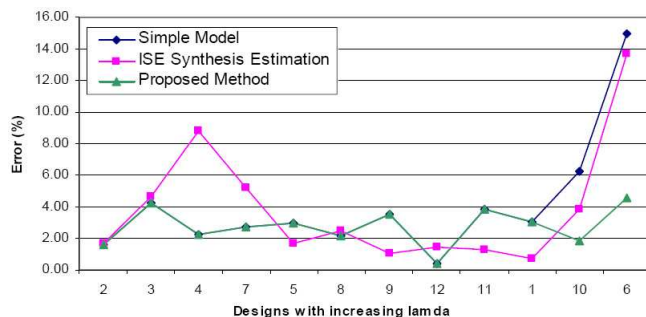

FIG. 5.1. *Estimation error with designs arranged in increasing of λ*

5.3. Estimation Runtime. The experiments were carried out on the Pentium 4 3GHz workstation, and the Xilinx ISE 8.1 tool was used to obtain the synthesis and post-place-and-route results. Table 5.4 compares the estimation time with the execution time of the Xilinx tool for synthesis and PAR. On an average, the proposed estimation process completes in the order of milliseconds except in cases 3, 4, 6, 7. In these cases, parsing of the Trimaran's textual output takes up to 2 seconds to complete, while the actual hardware binding and estimation process is performed in milliseconds. Overall, our technique achieves the estimation results about 350 times faster than the synthesis process, and 3000 times faster than the PAR process.

Table 5.4
Estimation Runtime Compared to Synthesis And Actual PAR time

	Functions	Estimation Time (s)	ISE Syn Time (s)	PAR Time (s)
1	Random Algorier	0.08	31	285
3	mpeg2-_bdist1_motion	2.54	255	676
4	mpeg2-_bdist2_motion	2.59	110	814
5	mpeg2-_dct_type_estimation	0.2	35	613
6	mpeg2-_dist1_motion	2.89	82	3612
7	mpeg2-_dist2_motion	2.71	112	687
8	mpeg2-_idctcol	0.23	141	2620
9	mpeg2-_idctrow	0.17	107	690
10	sha_transform	0.25	152	3600
11	adpcm_coder	0.2	81	780
12	adpcm_decoder	0.15	61	2700

5.4. Evaluation of the Framework for different FPGA Family. In order to evaluate the robustness of the proposed framework, we carried out experiments with the Spartan-3 FPGA. The following processes are repeated with the new target FPGA device: 1) characterization of components and interconnect, 2) identification of the value of λ and 3) area-time estimation. Due to the less sophisticated FPGA routing fabric in Spartan-3, we have obtained $\lambda = 0.72$. Table 5.5 and 5.6 show the quality of estimation compared to actual place and route values. The average errors of delay were found to be 3.4% while synthesis tool's estimation error is 8.5%. The results of the proposed area estimation are reasonably good compared to the results obtain from the synthesis tool. In Figure 5.2, the designs are rearranged in increasing order of . It can be observed that there is a large estimation error obtained using the simple model and ISE synthesis tool for designs with high value of . In contrast, the estimation error incurred with the proposed technique is consistent across the different functions. This implies that the proposed estimation technique leads to a higher degree of predictability when compared to the simple model and synthesis tool.
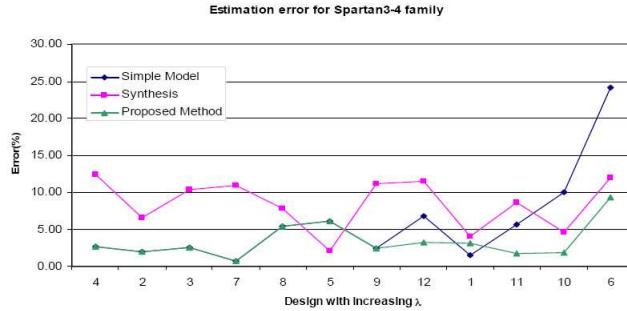

Fig. 5.2. *Estimation error with designs arranged in increasing of λ*

6. Conclusions. FPGA-based high-level area-time estimation that ignores the physical design effects after place and route may lead to very high inaccuracies. In this paper, we have presented a high-level estimation framework that can predict the area-time measures of C-based applications with post place and route effects taken into account. It is worth mentioning that the original C applications can be directly used in the proposed framework without any further modifications. Our area estimation

TABLE 5.5
Area Estimation for Spartan Family

	Pro LUT	Syn LUT	PAR LUT	Pro SLICE	Syn SLICE	PAR SLICE	Pro LUT (%)	Syn LUT (%)	Pro SLICE (%)	Syn SLICE (%)
Random Algorithm	1636	1580	1,568	1122	1170	1093	4.34	0.77	2.65	7.04
Matrix multiplier	285	284	278	239	270	222	2.52	2.16	7.67	21.67
mpeg2-_bdist1_motion	1978	1903	1,854	1349	1384	1564	6.69	2.64	13.72	11.51
mpeg2-_bdist2_motion	1962	1959	1,918	1349	1435	1649	2.29	2.14	18.19	12.98
mpeg2-_dct_type_estimation	623	624	616	440	508	792	1.14	1.30	44.51	35.86
mpeg2-_dist1_motion	2200	2168	2,153	1364	1392	1839	2.18	0.70	25.83	24.31
mpeg2-_dist2_motion	2625	2618	2,576	1585	1669	1567	1.90	1.63	1.15	6.51
mpeg2-_idctcol	4646	4511	4,481	3227	3519	3256	3.68	0.67	0.89	8.08
mpeg2-_idctrow	2942	2925	2,895	2095	2317	2140	1.62	1.04	2.10	8.27
sha_transform	2548	2545	2,530	1578	1693	2267	0.71	0.59	30.38	25.33
adpcm_coder	1688	1498	1,470	988	893	846	14.83	1.90	16.85	5.56
adpcm_decoder	1244	1165	1,148	878	859	807	8.36	1.48	8.80	6.44
Average Error							4.19	1.42	14.39	14.86

TABLE 5.6
Delay estimation for Spartan3-4 Family

	Sim (ns)	Syn (ns)	Pro. (ns)	PAR (ns)	Sim (%)	Syn (%)	Pro (%)
Random Algorithm	16.26	17.17	17.02	16.51	1.51	4.02	3.10
Matrix multiplier	13.54	14.14	13.54	13.28	1.97	6.53	1.97
mpeg2-_bdist1_motion	14.72	15.85	14.72	14.36	2.49	10.33	2.49
mpeg2-_bdist2_motion	15.90	17.42	15.90	15.49	2.62	12.45	2.62
mpeg2-_dct_type_estimation	13.54	14.71	13.54	14.41	6.06	2.04	6.06
mpeg2-_dist1_motion	9.16	10.62	10.94	12.06	24.08	12.00	9.33
mpeg2-_dist2_motion	15.08	16.62	15.08	14.98	0.65	10.93	0.65
mpeg2-_idctcol	15.08	17.17	15.08	15.94	5.37	7.78	5.37
mpeg2-_idctrow	15.08	17.17	15.08	15.45	2.38	11.18	2.38
sha_transform	8.13	9.45	9.21	9.04	9.98	4.55	1.89
adpcm_coder	8.76	10.08	9.45	9.29	5.65	8.57	1.74
adpcm_decoder	8.13	9.73	8.45	8.727	6.81	11.52	3.19
Average Error					5.80	8.49	3.40

has been shown to achieve comparable results with that obtained from a commercial synthesis tool. We have proposed a new metric for our delay estimation model that captures the placement complexity of the circuit. For the experiments considered, when compared to post place and route results obtained from a commercial tool, our proposed delay estimation achieves an average accuracy of 97% with a worst case error of only 4.5%. This result is significantly better than previously reported works in high-level delay estimation and the estimation process can be completed in the order of milliseconds. In addition, we have shown that the proposed framework provide consistent results for devices from the Xilinx Virtex and Spartan families.

REFERENCES

[1] XILINX CORPORATION. *VirtexII Pro Capabilities.* Available at http://www.xilinx.com/products/silicon_solutions/ fpgas/virtex/virtex_ii_pro_fpgas/capabilities/index.htm

[2] ALTERA CORPORATION. *Stratix II FPGA .* Available at http://www.altera.com/products/devices/stratix2/st2-index.jsp.

[3] L. N. CHAKRAPANI, J. GYLLENHAAL, W. MEIW. HWU, S. A.MAHLKE, K. V. PALEM, AND R.M. RABBAH. *Trimaran: An infrastructure for research in instruction-level parallelism.* In Lecture Notes in Computer Science (Langurages and Compilers for High Performance Computing), 2004, pp. 32-41.

[4] R. SCHREIBER, S. ADITYA, S. MAHLKE, V. KATHAIL, B. R. RAU, D. CRONQUIST, AND M. SIVARAMAN. *PICO-NPA: High-level synthesis of nonprogrammable hardware accelerators.* Journal of VLSI Signal Processing, 31(2002), pp. 127-142.

[5] MITRIONICS CORPORATION. *Mitron-C.* Available at http://www.mitrionics.com/

[6] CELOXICA CORPORATION. *Handel-C.* Available at http://www.celoxica.com/

[7] DSPLOGIC CORPORATION. *DSPLogic ToolBox.* Available at http://www.dsplogic.com/home/

[8] IMPULSEC CORPORATION. *Impulse C language.* Available at http://www.impulsec.com/

[9] ALTERA CORPORATION. *C to Hardware Technology.* Available at http://www.altera.com/products/ip/processors/nios2/tools/c2h/ni2-c2h.html.

[10] JUSTIN L. TRIPP, MAYA B. GOKHALE, KRISTOPHER D. PETERSON. *Trident: From High-Level Language to Hardware Circuitry.* Computer, 40(2007), pp. 28-37.

[11] SHAWN MCCLOUD. *Catapult C Synthesis-based Design Flow: Speeding Implementation and increasing Flexibility.* Mentor Graphic White Paper 2003. Available at http://www.mentor.com

[12] BYONGRO SO ET AL. *Using Estimates from Behavioral Synthesis Tools in Compiler Directed Design Space Exploration.* Proceedings of Design Automation Conference 2003, 2(2003), pp. 514-519.

[13] MIN JOONG RIM, RAJIV JAIN. *Lower-bound performance estimation for high-level synthesis scheduling problem.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 13(4-1994), pp. 451 - 458.

[14] CHAUDURI S AND WALKER. *Computing the lower bound of functional unit before scheduling.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol 4 Issue 2(1996), pp. 273-279.

[15] A. NAYAK, M.HALDAR, A.CHOUDHARY, AND P.BANERJEE. *Accurate Area and Delay Estimators for FPGA.* Proceedings of International Conference DATE, 2002, pp. 862-869.

[16] MINJOONG RIM AND RAJIV JAIN. *Estimating performance characteristic of Loop transformation.* IEEE International Symposium on Circuits and Systems, 1(1994), pp. 249-252.

[17] SEBASTIEN BILAVARN ET AT. *Design Space Pruning through estimations of Area/Delay Tradeoff for FPGA.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(2006), pp.1950-1968

[18] JOO M. P. CARDOSO. *On Estimations for Compilation of Software for FPGA.* Proceedings of the16th International Conference on Application-Specific Systems, Architecture and Processors (ASAP'05), 2005, pp. 225-230.

[19] DHANANJAY KULKARNI ET AL. *Compile Time Area estimation for LUT-based FPGAs.* ACM Transactions on Design Automation of Electronic Systems (TODAES), 11(2006), pp. 104-122.

[20] XILINX CORPORATION. *Xplorer Technology.* Available at http://www.xilinx.com/products/design_tools/logic_design/implementation/xplorer.htm.

[21] MIN XU. FADI J. KURDAHI. *Area and timing estimation for lookup table based FPGA.* Proceedings of the 1996 European conference on Design and Test, 1996, pp. 151.

[22] SHANKAR BALACHANDRAN. *A-priori wirelength and interconnect estimation based on circuit characteristics.* Proceedings of the 2003 international workshop on System-level interconnect prediction SLIP, 2003, pp. 77-84.

[23] TANAY KARNIK AND SUNG-MO KANG. *An Empirical Model For Accurate Estimation of Routing Delay in FPGAs.* Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design, 1995, pp. 328-331.

[24] MICHEAL HUTTON. Interconnect Prediction for Programmable Logic Devices. Proceedings of the 2003 international workshop on System-level interconnect prediction, 2003, pp.31-38.

[25] VALAVAN MANOHARARAJAH ET AL. Difficulty of Predicting Interconnect Delay in a timing driven FPGA CAD Flow. In Proceedings of International Proceedings on SLIP, 2006, pp. 3-8.

[26] SEONG Y. OHM ET AL. *A comprehensive estimation technique for High-level Synthesis.* Proceedings of the 8th international symposium on System synthesis, 1995, pp. 122-127.

[27] CARLO BRANDOLESE ET AL. *An area estimation methodology for FPGA based Designs at system C- level.* Proceedings of the 41st annual conference on Design automation, 2004, pp. 129-132.