Improvement of ASIC design processes

Vineet Sahula
Deptt. of ECE
Regional Engg. College, Jaipur
sahula@ieee.org

C. P. Ravikumar
Texas Instruments
Asia Research Center,Bangalore
ravikumar@india.ti.com

D. Nagchoudhuri
Deptt. of EE
Indian Institute of Technology, Delhi
dnag@ee.iitd.ernet.in

Abstract

With device counts on modern-day ASICs crossing the 10 million mark, careful planning of an ASIC design project is necessary to meet time deadlines. Two problems arise in this context. The first is the estimation of man-months for a project, with the knowledge of the ASIC design flow that will be followed for project execution. The second problem is that of making incremental changes to the design flow in order to reduce the time to complete a project. We consider these two problems in a theoretical framework. Starting from a textual description of the design flow, a model known as the hierarchical concurrent flow graph (HCFG) model is constructed to capture the concurrency in the execution of an ASIC design flow and the inherent hierarchy in such a flow. The HCFG model allows us to (a) quickly estimate the project execution time and (b) analyze the effect of introducing AND and OR concurrency in the flow to improve the execution time. We illustrate the use of the powerful estimation technique through two examples. The first example shows the use of AND concurrency in a back-end flow and the second example shows the use of OR concurrency in a software design flow.

1 Introduction

Project managers in ASIC design houses grapple with the problem of estimating the manpower requirement of design projects. Currently, manpower estimation is more of an art than a science. A manager bases the estimate on past experience and the size and complexity of the new project. However, these estimates can be grossly incorrect, throwing the project plan off the balance and creating practical difficulties in resource scheduling. Adding to this complication is the complex nature of modern-day ASIC design: the design flow is iterative, hierarchical, and concurrent. The flow has well-defined steps such as RTL design, Verification, Synthesis, Physical Design, and Physical Verification. Design steps may have to be iterated more than once in order to achieve timing closure or due to constraints on area, test cost, and reliability. Concurrency in the design flow is introduced by the project manager in order to speed up the project execution. For example, the manager may subdivide the design into smaller subdesigns and exploit spatial parallelism. Accurate prediction of the project execution time will require careful calibration of the design process to estimate the parameters associated with the design flow, such as the probability of iteration. In addition to the problem of predicting the manpower requirement for a project, the manager must also consider altering the design flow in order to speed up project execution. A "what if" analysis tool will be invaluable during the project planning phase.

In this paper, we consider the problem of project time prediction and project flow improvement in a theoretical framework. We describe the use of a model known as the hierarchical concurrent flow graph (HCFG) to capture the iterative, hierarchical, and concurrent nature of ASIC design flows. The model can be constructed from a textual description of the process flow and can efficiently predict the project execution time. Using two examples from the VLSI system design domain, we illustrate the power of the HCFG model in the improvement of process flows.

The paper is organized as follows. In the next section, we briefly discuss the HCFG model and the associated analytical techniques. In Section 3, we consider a physical design flow and estimate the improvement in the execution time of the flow when the design is partitioned. Section 4 considers a software design flow and illustrates the use of HCFG in predicting the improvement in the flow execution time when OR concurrency is introduced in the flow. Conclusions are presented in Section 5.

2 HCFG approach

A node in an HCFG corresponds to a task in the ASIC design flow. Two special nodes I and F in the HCFG denote the initial task and the final task in the design flow. A weight T_j is associated with the node j and corresponds to the completion time of the task. Since task completion time can rarely be predicted with point accuracy, we treat T_j as a discrete random variable with a specified distribution. Notation $E[T_j]$ or $\overline{T_j}$ is used to denote the expected value of T_j. A directed edge (i, j) in the HCFG represents a sequencing of task i followed by task j. A weight p_{ij} associated with the edge (i,j) denotes the probability that task j is executed after task i. Mason's flow graph technique, well known in Control theory, is used to compute the *transmittance* $\mathcal{T}_{I,\mathcal{F}}$ of the HCFG [3]. Before computing the transmittance, the weight of edge (i,j) is calculated as z^{T_j}, where z is the variable of the z-Transform. The graph transmittance can be used to obtain various attributes of T_P like $E[T_P]$, $E[T_P^2]$ and variance of T_P as given below [3].

$$E[T_P] = z\frac{d\mathcal{T}_{I,\mathcal{F}}}{dz}\bigg|_{z=1}$$

$$\begin{aligned} E[T_P^2] &= z\frac{d}{dz}\left(z\frac{d\mathcal{T}_{I,\mathcal{F}}}{dz}\right)\bigg|_{z=1} \\ \sigma^2_{T_P} &= E[T_P^2] - (E[T_P])^2 \end{aligned}$$

2.1 Process improvement approaches

There are many options that a project manager will consider to reduce the expected project completion time. These can be summarized as follows. (1) Process model parameters can be tuned to reduce $E[T_P]$. Improving the design parameters will result in an improvement in $E[T_P]$. (2) Shuffling the order of tasks may result in reduction in $E[T_P]$. It is feasible only when the interdependency of the tasks allows for such reshuffling [4]. (3) Decomposing an activity into smaller tasks and deploying them concurrently using a reassignment of manpower. The AND Concurrency construct of HCFG is useful in expressing this type of concurrency. (4) Identifying alternate ways to solve a problem and executing all of these sub-flows concurrently. The best of these solutions will be actually used. The OR concurrency construct in an HCFG is useful in expressing this option.

In this paper, we shall consider the third and the fourth options for design time improvement. We develop a framework for decision making, which uses quantitative analysis of $E[T_P]$. We consider two example design flows to illustrate the process improvement paradigm- *timing driven layout design flow* of Figure 1 and the flow of Figure 4 for a *software design process*. This software design flow may be considered as a subflow of a larger hardware-software codesign flow. We also describe a method for task time parameterization in terms of design and designer characteristics.

3 Improving physical design flow

As the first example, we consider a timing-driven physical design flow and consider how the estimated execution time for the flow can be improved through the use of AND concurrency. Since interconnect delays play a dominant role in determining the performance of deep submicron integrated circuits, it is common to use a timing-driven physical design flow, with the objective of improving the chances

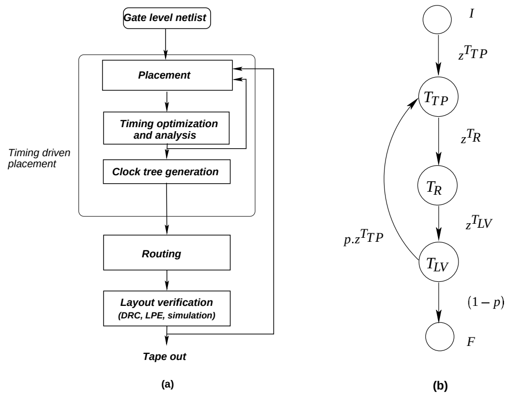


Figure 1. (a) Flow for timing driven Layout design (b) HCFG equivalent

of timing closure by considering timing constraints during the various phases of physical design. Figure 1 shows a design flow that may be useful in the physical design of high-performance chips. The design tasks in the flow are timing-driven placement, routing, and layout verification. Timing-driven placement consists of the three subtasks, namely, initial placement, timing analysis and optimization, and clock-tree generation. Since the complexity of these design tasks increases superlinearly with the size of the netlist, there is a definite benefit in partitioning the circuit into several blocks and carrying out the physical design flow separately on these blocks. The layouts of the different blocks can then be merged together to result in the complete chip layout. Techniques such a buffer insertion, gate resizing, wire resizing, and fanout decomposition can be used to improve the timing of paths that are local to blocks as well as the paths that cut across different blocks if some paths are found to violate the timing constraints.

The parameters that affect the execution time of the different tasks are the number of partitions and the average size of each partition. Figure 1(b) shows the HCFG for the flow of Figure 1(a). We use the notation T_{TP}, T_R, and T_{LV} to denote the completion times of the timing-driven placement task, the routing task, and the layout verification task respectively. Let p be the probability of repeating the entire flow due to timing failure detected during layout verification. Figure 2 shows the HCFG for a flow that uses design partitioning to speed up the original design flow. We assume that the partitioning task subdivides the original circuit into $n > 1$ blocks. In the HCFG of Figure 2, two new tasks, namely, Partitioning and Module Integration & Verification have been added. In addition, two special nodes called AND Concurrency nodes have also been added (nodes marked with a $\odot$). We make n copies of the flow corresponding to the flow of Figure 1 and insert these as subflows between the two AND concurrency nodes. Note that a layout verification step is carried out on the complete chip layout after the merging step. Let p_I indicate the probability of detecting timing violations after this final verification step. Let T_{TP_j}, T_{R_j}, and T_{LV_j} indicate the time taken for timing-driven placement, routing, and layout verification for the block j. Let p_j indicate the probability of detecting timing violations at the block-level in block j. Let T_{part} and T_{IV} indicate the completion time for partitioning and module integration & verification tasks, respectively.

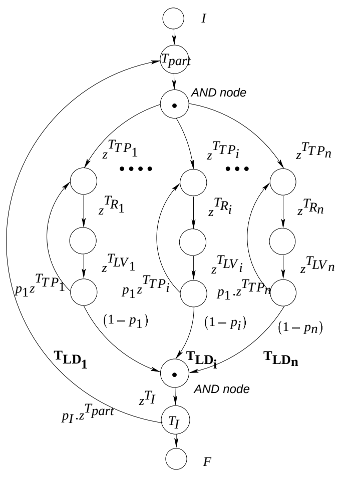


Figure 2. **HCFG for flow of Figure 1 with AND concurrency introduced**

3.1 Process completion time

Using the techniques described in [3], the graph transmittance $\mathcal{T}_{I,\mathcal{F}}$ and the expected run time $\overline{T_P}$ for the flow of Figure 1 can be found as

$$\mathcal{T}_{I,F} = \frac{(1-p)z^{(T_{TP}+T_R+T_{LV})}}{1-pz^{(T_{TP}+T_R+T_{LV})}}$$

$$\overline{T_P} = \frac{T_{TP}+T_R+T_{LV}}{1-p} \tag{1}$$

Let $\mathcal{T}'_{I,\mathcal{F}}$ indicate the graph transmittance for the modified flow depicted in Figure 2, and let $\overline{T'_P}$ indicate the expected run time for the flow given by the following equations.

$$\begin{aligned} \mathcal{T}'_{I,F} &= \frac{(1-p_I)\cdot z^{(T_{part}+T_{AND}+T_{IV})}}{1-p_I\cdot z^{(T_{part}+T_{AND}+T_{IV})}} \\ \overline{T'_P} &= \frac{T_{part}+\overline{T_{AND}}+T_{IV}}{1-p_I} \end{aligned} \tag{2}$$

In the above equations, we use the notation T_{AND} to indicate the expected time for the entire sub-flow that appears between the two AND node pairs of Figure 3. The discrete density function (DDF) of T_{AND} can be found using the technique of [3], and we primarily focus on the expected value $\overline{T_{AND}}$. This quantity is the expected value of the maximum of the random variables T_{LD_j}, where T_{LD_j} denotes the run time of the subflow j. We can approximate $\overline{T_{AND}}$ by the maximum of the expected values of T_{LD_j} (see equation below).

$$\overline{T_{AND}} = E\left[\underset{i=1..n}{MAX}\left\{T_{LD_j}\right\}\right] \sim \underset{i=1..n}{MAX}\left\{E\left[T_{LD_j}\right]\right\}$$

Now, using the earlier result for the expected completion time for the flow of Figure 1, we can write $\overline{T_{LD_j}}$ as

$$\begin{aligned} \overline{T_{LD_j}} &= \frac{T_{TP_j}+T_{R_j}+T_{LV_j}}{1-p_j} \\ &= x_j\cdot\frac{(T_{TP}+T_R+T_{LV})}{1-p_j} = x_j\cdot\left[\frac{1-p}{1-p_j}\right]\cdot\overline{T_P} \end{aligned}$$

Here, x_j denotes the normalized size of block j with respect to the size of the complete circuit. Clearly, the largest sized block will dictate the execution time of the AND subflow. This is expressed by the following equation.

$$\overline{T_{AND}} = x_m\cdot\left[\frac{(1-p)}{(1-p_m)}\right]\cdot\overline{T_P}$$

3.2 Characterizing model parameters

We now consider the tasks Partitioning and Module Integration & Verification and analyze the execution time of these tasks. Our experience with a partitioning tool indicates that the time for partitioning depends mainly on the size of the original netlist. Since this size is a constant for a given problem, we can treat T_{part} as a constant. The time for module integration depends on the number of partitions n and the number of modules N in the original netlist. It also depends on the Rent's coefficient q, which relates the number of IO pins of a block with the number of gates in the block [1]. We noticed that the time for the module integration & verification task can be described using the following equations, which give T_{IV} for different ranges of n.

$$T_{IV} = \begin{cases} N^q\left(\frac{1}{n^q}-\frac{q}{n}\right)\left(\frac{n-1}{N-1}\right)\overline{T_P} & n\le n_1 \\ \left(\frac{T_{IV}|_{n=n_2}-T_{IV}|_{n=n_1}}{n_2-n_1}\right)(n-n_1)+T_{IV}|_{n=n_1} & n_1<n\le n_2 \\ \frac{b}{\overline{T_{P_1}}}\cdot\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}T_{LD_i}.T_{LD_j} & n_2<n\le N \end{cases} \tag{3}$$

The probability p_I is characterized as $p_I=\left(\frac{n}{N}\right)^r\cdot p$, where $r\ge 1$. As this expression indicates, p_I is no greater than the probability p of the original flow shown in Figure 1(b). When n is large, the module integration task essentially boils down to the chip-level layout verification task. Thus, in the limit as n approaches the number of modules N, p_I approaches p. The value of the constant r depends on the circuit and the quality of the partitioning tool; we tuned the value of r to 1.9 in our experimentation. We also assumed that p_j is proportional to the size of the block j; this is based on the intuition that the layout of larger blocks is more likely to involve several timing iterations.

3.3 Results

We conducted experiments in order to compare the execution times of the the flows of Figures 1(b) and 2. In our experimentation, we assumed the following. $T_{TP} = 120$, $T_R = 24$, $T_{LV} = 80$, $T_{part} = 4$. We considered four different values for n, namely, 1,2,4, and 8. We repeated the computation for five different values of p, viz, 0.1, 0.2, 0.3, 0.4 and 0.5. We observe that partitioning load imbalance has a negative effect on $\overline{T'_P}$. In our further discussion we assume a balanced partition, i.e. $x_j = \frac{1}{n}$ for all j. The results of the computation of process completion times T_P and T'_P are shown in Table 1. The value of $n = 1$ pertains to flow of Figure 1. The HCFG analysis provides the discrete density function for the completion times from which various other attributes of completion time can be computed.

Table 1. Layout-Design time for TMS320C30

p	n	T_I	$\overline{T'_P}$	$\sigma_{T'_P}$	*LB and UB on* T'_P
	1	0	448	315	224-1568
0.5	2	1	306	110	229-687
	4	2	273	72	230-622
	8	5	301	139	233-951

HCFG analysis indicates that opting for second flow results in reduced completion time. There is a value of n, denoted by n_{opt}, at which $\overline{T'_P}$ is minimum. $\overline{T'_P}$ increases when n either decreases or increases around n_{opt}. For large p, the second flow is a better alternative for a large range of n ($2 \leq n < n_{opt}$) whereas for small p, the second flow is still a better choice but range of n is restricted i.e. n_{opt} is small. We performed an analysis based partly on HCFG approach to compute n_{opt}. We made use of equations 1-3 to obtain an analytical expression for $\overline{T'_P}$ in the following manner. We substituted T_{IV} from equation 3 and p_I into equation 2 to obtain the expressions for $\overline{T_{AND}}$ and $\overline{T'_P}$ given in equations 4 and 5 as functions of n.

$$\overline{T_{AND}} = \frac{1}{n}\left[\frac{1-p}{1-p_i}\right]\overline{T_P} = \frac{1}{n-p}(T_{TP} + T_R + T_{LV}) \qquad (4)$$

$$\overline{T'_P} = \frac{T_{part} + \left(\frac{1-p}{n-p}\right)\cdot\overline{T_P} + N^q\left(\frac{1}{n^q} - \frac{q}{n}\right)\frac{(n-1)}{(N-1)}\cdot\overline{T_P}}{1 - \left(\frac{n}{N}\right)^r \cdot p} \qquad (5)$$

The optimum value n_{opt} can now be obtained by solving the expression $\frac{d}{dn}\left[\overline{T'_P}(n)\right] = 0$ for n. The optimum value for $\overline{T'_P}$ is obtained from equation 5 for $n = n_{opt}$. The variation of $\overline{T'_P}$ with respect to n, expressed in equation 5, is graphically illustrated in Figure 3, for $N = 50$, $p = 0.5$, $q = 0.77$ and $r = 1.9$. The value of n_{opt} is graphically obtained as 5.

4 Improving a software design flow

For our next illustration, we consider the flow of Figure 4(a) for a software design process for an Application Specific Instruction-set Processor (ASIP) or a DSP [2, 5]. The chief concern here is one of retargetable code generation. The specifications of the SW part of the system, obtained after hardware-software partitioning, are translated into a high level language. The requirements of the application guide the selection of processor architecture and the number of processors required. The code is first partitioned and the scheduling of the processes is carried out on multiple processors. Once the appropriate architectures are chosen, the code assigned to respective architectures is translated into architecture-specific assembly code. It is then followed by a verification step which is done either through SW emulation of the target architecture or by running the code on an actual processor.

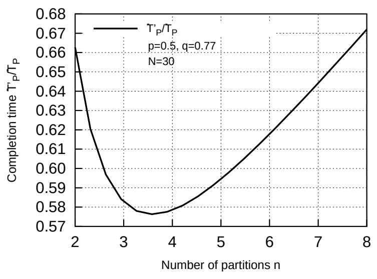


Figure 3. Time and probability variations with n

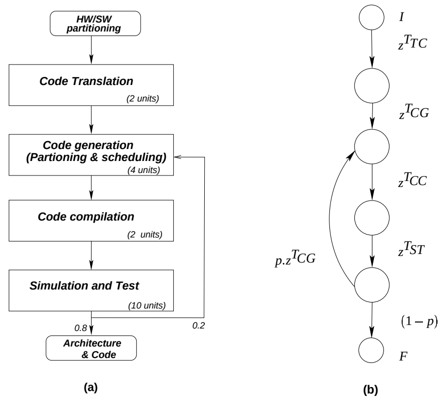


Figure 4. (a) Flow for Software Design of ASIPs (b) HCFG equivalent

4.1 Introducing OR concurrency

For the design flow in Figure 4(a), we consider two alternate execution scenarios. In Scenario 1, the flow gets executed as a purely sequential and iterative design flow. The HCFG equivalent is shown in Figure 4(b). In Scenario 2, a team of n designers is provided with the same specifications to execute the design flow of Figure 4(b). In this situation, we assign the design task to n designers with each of them working concurrently on a separate but identical subflow. In the HCFG equivalent of the flow, we introduce a pair of OR node which encloses all the subflows executed by these individual designers. Such a graph is shown in Figure 5 for $n = 2$. We assume that the i^{th} designer provides a design of quality Q_i. Q_i can be characterized in terms of (a) performance of the system, (b) power, (c) the software execution time and (d) the number of defects in the code.

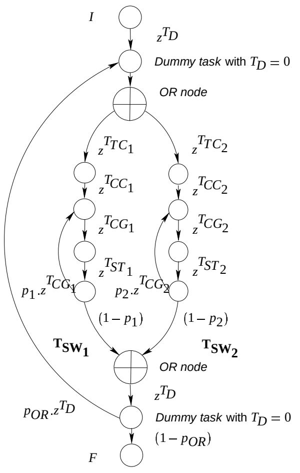


Figure 5. HCFG equivalent for modified software design flow

In Scenario 2, the execution of all the subflows within an OR node-pair is stopped as soon as any one of the n designers provides a design that satisfies the quality criterion. Let T_{TC}, T_{CG}, T_{CC}, and T_{ST} be the completion times for activities *code translation, code generation, code compilation* and *simulation & test* respectively for Scenario 1. Let p be the probability of transition from the *simulation and testing* task to the *code translation* task. Let T_{TC_i}, T_{CG_i}, T_{CC_i}, and T_{ST_i} be the corresponding quantities for the i^{th} designer in Figure 5. Let T_{SW_i} denote the completion time of subflow executed by the i^{th} designer. Similarly, let p_{OR} be the probability of transition to repeat the OR subflow; this iteration is caused if the quality of the product is unacceptable.

4.2 Process completion time

For project 1 $\mathcal{T}_{I,F}$ and $\overline{T_P}$ can be written as

$$\mathcal{T}_{I,F} = (1-p)\frac{z^{(T_{TC}+T_{CC}+T_{CC}+T_{ST})}}{1-p\cdot z^{(T_{CG}+T_{CC}+T_{ST})}}$$

$$\overline{T_P} = T_{TC} + \frac{T_{CG}+T_{CC}+T_{ST}}{1-p}$$

If the graph transmittance of the subflow confined in the OR-node pair is given by $\mathcal{T}_{\mathcal{OR}}$, the graph transmittance $\mathcal{T}'_{I,F}$ for flow of Scenario 2 is given by equation 6. For $T_D = 0$ it reduces to equation 7.

$$\mathcal{T}'_{I,F} = (1-p_{OR})\frac{\mathcal{T}_{OR}\cdot z^{(T_D+T_D)}}{1-p_{OR}\cdot\mathcal{T}_{OR}\cdot z^{(T_D+T_D)}} \tag{6}$$

$$\mathcal{T}'_{I,F} = (1-p_{OR})\frac{\mathcal{T}_{OR}}{1-p_{OR}\cdot\mathcal{T}_{OR}} \tag{7}$$

The process completion time T'_P is given by

$$\overline{T'_P} = \frac{\overline{T_{OR}}}{1-p_{OR}}.$$

The graph transmittance $\mathcal{T}_{OR}$ and completion time T_{OR} for the OR-node pair subflow is obtained using HCFG analysis. $\overline{T_{OR}}$ can be approximated to be equal to the minimum of the expected values of T_{SW_i}, as shown in equation 8.

$$\overline{T_{OR}} \;=\; E\left[\begin{array}{c} MIN \\ i=1..n \end{array}\{T_{SW_i}\}\right] \sim \begin{array}{c} MIN \\ i=1..n \end{array}\{E\left[T_{SW_i}\right]\} \tag{8}$$

The average completion time for the i^{th} designer $\overline{T_{SW_i}}$ is the expected value of completion time of the i^{th} subgraph between the OR nodes in Figure 5. It can be rewritten as

$$\overline{T_{SW_i}} = T_{TC_i} + \frac{T_{CG_i}+T_{CC_i}+T_{ST_i}}{1-p_i}$$

4.3 Characterizing model parameters

To represent the flow of Figure 5 by HCFG, two dummy tasks of zero completion times have to be introduced. Now we discuss the probability p_{OR} in the following sections.

4.3.1 Probability p_{OR}

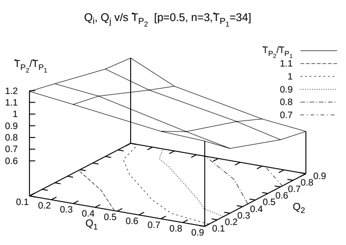
The probability p_{OR} is associated with the quality of design artifact provided by all the n designers relative to the specified quality Q_D. Let $Q_{i_{max}}$ denote the highest quality for the design artifacts provided by n designers. Intuitively, we expect p_{OR} to be larger when $Q_{i_{max}}$ is low and vice-versa. Consistent with this intuitive expectation, we assume

$$p_{OR} = \left(c - d\cdot n\cdot\frac{\frac{Q_{i_{max}}}{Q_D}}{\sum_{i=1}^{n}\frac{Q_D}{Q_i}}\right)\cdot p \tag{9}$$

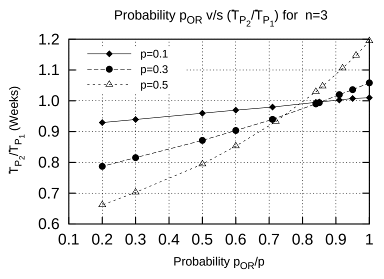
The actual values of c and d can be calibrated using the design history meta-data. We choose $c = 1$, $d = 1$ and $n = 2$. Note that (a) $0 < Q_i \leq 1$, (b) a designer completing the design very early is expected to provide a lower quality product causing p_{OR} to increase. We expect that no designer will exceed the expectation or $Q_i < Q_D$.

4.4 Results

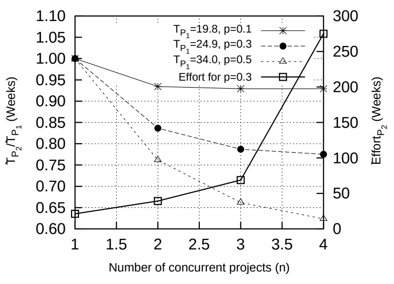
In this section, we discuss the results obtained by comparing the process completion times of Scenario 1 and Scenario 2. We assume $T_{TC} = 2$, $T_{CG} = 4$, $T_{CC} = 2$, $T_{ST} = 10$ and $T_D = 0$ time units. To explore the possibilities of reducing process completion time, the following parameters were considered for variables- n, Q_i and p: $n \in \{1,2,3,4\}$, $Q_i \in \{0.1,0.3,0.7,0.9\}$, $p \in \{0.1,0.2,0.3,0.4,0.5\}$. The results of computation of $\overline{T_P}$ and $\overline{T'_P}$ are shown in Table 2.



(a) **With** Q_1 **and** Q_2



(b) **With** p_{OR}



(c) **With** n

Figure 6. Process completion time T_P' variations

Table 2. Software Design completion time

p	n	$Q_1 = Q_2$	p_{OR}	$\overline{T_P'}$	$\sigma_{T_P'}$	*LB and UB on* T_P'
0.3	1			24.9	12.5	18-66
	2	0.1	0.30	27.9	16.6	18-102
		0.9	0.06	20.8	7.5	18-52

HCFG analysis provides DDF of $\overline{T_P'}$. Also shown are various attributes of $\overline{T_P'}$ computed from its DDF.

We make these observations from the results. (1) Opting for Scenario 2 certainly results in reduced completion time. The region of values of Q_1 and Q_2, which results in $T_P' < T_P$, falls to the right of the contour for "T_P'/T_P= 1" in x-y plane in Figure 6(a). (2) For large p, flow of Scenario 2 is a better alternative for a larger range of Q_i $(1 > Q_i > Q_{opt})$ and Q_{opt} can be less than $\frac{1}{2}$. Q_{opt} is the minimum value of quality at which $T_P' < T_P$. (3) For small p, flow of Scenario 2 would still be a better alternative but for a small range of Q_i, i.e. Q_{opt} has to be larger than $\frac{1}{2}$; additionally all Q_i have to be approximately equal.

When both designers provide quality that are nearly equal and are very close to the desired one, flow 2 entails greater possibility of providing reduced $E[T_P]$ as shown in Figure 6(a). For low values of quality, flow 2 cannot be recommended, see Figure 6(b). Also, given the capability of designers to produce products of certain quality, flow 2 offers better improvement in $E[T_P]$ over flow 1 only when p is large, as illustrated by Figures 6(b) and 6(c). For example, in Figure 6(b), for $p = 0.5$ and relative probability $\frac{p_{OR}}{p} = 0.6$, $E[T_P]$ for flow 2 is 85% of $E[T_P]$ of flow 1. For $p = 0.1$, both the flows yield approximately the same $E[T_P]$.

We infer that (a) for a given team of designers, Flow 2 is advisable only when p is sufficiently large and (b) for small p, Flow 2 provides improved $E[T_P]$ when all designers produce high quality designs. For the SW design projects where a choice of different architectures is available, this paradigm for $E[T_P]$ improvement can be combined with the usual design space exploration, adding one more dimension to the design space. Given constraints on available manpower, a suitable value of n can be found which results in minimum process completion time.

5 Conclusions

We illustrated a methodology for process improvement using the HCFG approach. For DSM chips, the effect of partitioning the circuit or considering elaborate interconnect models on design completion time was explored using the HCFG approach. For a software design flow, for a given size and complexity of the design, the effect of designer expertise on design completion time was explored. There is a penalty involved in using concurrency constructs for reducing the design time. In the use of OR construct, the penalty is in the form of increased design effort, without excessive degradation in utilization factor. Use of AND construct decreases the utilization for a marginal increase in design effort.

Acknowledgements

This work was carried out at IIT, Delhi. The authors wish to thank Motorola, USA for funding this research.

References

[1] W. E. Donath. Placement and average interconnect length of computer logic. *IEEE Trans. on Circuits and Systems*, CAS-26(4):272–277, Apr. 1979.

[2] K. Kucukcakar. An ASIP design methodology for embedded systems. In *Proceedings of Workshops on Hardware/Software codesign*, 1999.

[3] V. Sahula and C. P. Ravikumar. Hierarchical Concurrent Flow Graph approach for modeling and analysis of concurrent design processes. In *Proceedings of* 14th *IEEE International Conference on VLSI Design*, Jan. 2001.

[4] R. P. Smith and S. D. Eppinger. Identifying controlling features of engineering design iterations. *Management Science*, 43(3):276–293, Mar. 1997.

[5] W. Wolf. *Hardware Software codesign: Principles and practice*. 1994.