OPTIMAL MODULE SET AND CLOCK CYCLE SELECTION FOR DSP SYNTHESIS

Liang-Gee Chen and Lih-Gwo Jeng

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R. O. C.

Abstract

In this paper, we present a program to determine the optimal module set and clock cycle time of an application specific digital signal processor at the beginning of synthesis. This method performs a systematically design space exploration in actual time and space domain. The module set can include single-cycle, multi-cycle or pipelined operators. The optimal clock cycle selection is done by scanning all suboptimal clock cycle values in an efficient way. The cost of hardware considers both the cost of the data path and that of the controller. Several examples are illustrated to show the advantages of our approach.

Module Name	Operation	Area mil^2	Delay ns	$A * T$
a2(16bits)	addition	1200	1510	1812000
ax(16bits)		2950	540	1613000
a1(16bits)		2880	530	1526400
a0(16bits)		4200	340	1428000
m0(16bits)	multiplication	49000	375	18375000
m1(16bits)		9800	2950	28910000
m2(16bits)		7100	7370	52327000
s0(16bits)	subtraction	4200	340	1428000
s1(16bits)		2880	530	1526400
s2(16bits)		1200	1510	1812000
r(16bits)	register	500	5	2500

Table 1: Module Library

1 Introduction

In application specific digital signal processor synthesis, the algorithms of DSP application are implemented on a set of hardware modules which are running at a certain clock rate. In this case, the initial selection of modules and the clock cycle time of the data path are important to a final result. The savings in repetitive processing time using the module set and clock cycle selection is a major motivation for this work.

The module set and clock cycle selection procedure is done before scheduling to determine an optimal module set and clock cycle time of an arbitrary data path. This technique is based on several predictive ability of the original data flow and integrated the predicted results using a heuristic globally cost function which both consider the clock cycle T_c, system latency l and the delay and area of individual modules as well as that of a microcode controller.

The Chippe design system [1] and the SPAID system [2] use a user-defined clock time as a basis for scheduling. The Chippe supports multi-cycle operation and chaining but requires the user to explicitly mark function units that are to be chained.

In [3], the module selection is done before scheduling. The clock cycle for a design is chosen as the maximum module delay of the optimal module set. This method promotes chaining but precludes multi-cycle scheduling of operations.

In the context of high-level-synthesis, the module set represents the space domain in the design space. For the same type of operation nodes, fast modules take a larger area than slower modules. As the area is restricted or just one operation node of a certain type in the data flow graph, then slower modules with less area is preferable. At the beginning of synthesis, hardware balance can be estimated, based on a evaluation of all critical paths, the operation nodes $V=\{N_{op_1}, ...N_{op_n}\}$ of each types and the system timing constraint on latency or throughput. The clock cycle represents the time domain in the design space, and the balancing between number of the system control steps and the length of a control step. A module set which contains various module delays, $\{d_{op_1}, ..d_{op_n}\}$, then choosing gcd of these delays as the clock cycle is the most saving the execution time. The trade-off is more control steps that complete the design, because the clock scheme of each operation would be multi-cycled. The object of module and clock cycle selection is to provide a reasonable initial solution of the module set and the clock cycle time for the task of synthesis based on a heuristic cost function.

2 Methodology

Starting point for our consideration is a DFG which consists of a set of operation nodes V and a set of directed edges E which describe the data dependencies. There are a predefined module library, as shown in table 1, which can provide module set s_i and the clock cycle t_{c_i} to minimize the global cost function F_{cost}.

$$F_{cost} = cost(hardware) * time(execution) \qquad (1)$$

2.1 Timing Consideration

The normal operation during one control step is reading the input data from register, processing the data and then storing the result, as shown in figure 1. Thus, the timing model of the clock cycle T_c is defined as

$$T_c = t_{read} + t_p + t_{store} \qquad (2)$$

t_{read}, t_{store} : *the data transfer delay.*
t_p : *the data processing delay.*

The multi-cycle operation which reads data in t_{read} of the first cycle and writes data in t_{store} of the last cycle and the arithmetic logic executed between them. And the timing consideration is also suitable for chaining operations.

2.2 Candidate Module Set

The architectural model of operational module is shown in figure 2, which formulated both pipelined and nonpipelined modules. The operation delay of pipeline modules is specified as stage delay [5].

For the multicycle clock scheme is allowable, the fast modules in module set will not decrease its speed performance due to synchronization with slower modules. The slower modules with less area than the fast one will gain the area performance. In this case, the module is not *redundant* if it gains a speed performance or an area performance than other module. In table 1, ax is a redundant module of $a1$. *For redundant module we mean that a module whose area and delay is larger than another module.* In table 1, the module ax gains a better speed performance than that of $a2$ and gains a better area performance than that of $a0$, but ax is redundant module of $a1$. Since $a1$ gains both a better area and a better speed performance than ax, which a design using ax is less performance than using $a1$.

Every not *redundant* module generates the candidate module sets. Once the module sets are generated, the associated local optimal clock cycles also can be generated, which is depicted in the next section. If there are constraints on the design, then the performance evaluation is under this constraint.

2.3 Finding Optimal Clock Cycle

Traditionally, the clock cycle T_c is chosen by the maximum value among all the delays of the different functional units of a module set. Thus, the natural upper bound for T_c is dependent on the delay of each kind of module in the optimal module set. And the absolutely lower bound, $T_{c_{min}}$, which is a technology dependent parameter for the implementation of controller; therefore: $T_c > MAX(T_{c_{min}}, t_{read} + t_{store})$. Although the range of T_c is specified, the enumeration of the clock cycle time is still a problem. Based on the following survey, the enumeration of the clock cycle can be performed in an efficient manner.

In a synchronous system, by employing a fast T_c which is shorter than the slowest functional unit delay, the slowest module will be synchronized as a multi-cycled module, $c_i * T_c$, with a synchronized redundant delay, r_i, which is expressed as:

$$c_i * T_c \geq dop_i \ \ or \ \ c_i * T_c = dop_i + r_i$$

Because finding the gcd of module delays as the clock cycle time is not practical, the local optimal values of these module delays might be the good choice points of the clock cycle time. The local optimal clock cycle for a candidate module set is depicted below:

Definition of local optimal clock cycle: Suppose there are n types of operations op_i with delay dop_i in a candidate module set. For a clock cycle t_c, these operations are synchronized as $c_1, c_2, .., c_n$ cycled operation with the synchronized redundant delay $r_1, r_2, .., r_n$, and then the common factor of $c_1, c_2, .., c_n$ is 1 and the local optimal clock is existed at least one $r_i = 0$ for the clock scheme $c_1, c_2, .., c_n$.

The clock cycle time selection procedure is to scan all possible cycle time between upper and lower bounds, where upper bound means the longest operation delay in a candidate module set. And, lower bound is the speed limitation of the controller. The clock scheme is defined as the number of cycles needed to complete the operation. One example illustrated the local optimal clock cycle is shown in figure 3, which uses one module set consisting of an adder($16bits$, $40ns$, $2000mil^2$), a multiplier($16bits$, $125ns$, $6200mil^2$) and a register ($16bits$, $5ns$, $500mil^2$) and with the $T_{c_{min}}$ absolutely lower bound $25ns$ on the 8-tap FIR[4].

By scanning the local optimal values of all the module delays, the range of the continuous clock cycle design domain is searched and exploited efficiently. There are four choices of the clock cycle time in figure 3-(a), $T_c = 68ns$ with one cycle addition and two cycle multiplication minimizing the cost function. But under different constraint, as in figure 3-(b) $T_c = 50ns$ is for a minimum system latency. area design. In figure 3-(c), $T_c = 135ns$ is for a minimum area design.

2.4 Hardware Cost Consideration

2.4.1 Data Path Consideration

For a possible data path design, the number of control steps is related to the clock cycle time and the system latency. This relationship is given by:

$$S_c = \lfloor \frac{l}{T_c} \rfloor \tag{3}$$

l : *system latency*, $\frac{1}{throughput}$, *ns*.
T_c : *the clock cycle, ns*.
S_c : *number of control steps*.

In order to enhance the performance of the fast and pipelined functional unit, multicycle and pipelined clock schemes are allowable. In the following two equations which formulate the cost model of the single-cycle, multi-cycle and pipeline operators.

$$m_{op_i} = \lceil (\frac{N_{op_i}}{\lfloor (S_c / \lceil \frac{dop_i}{T_c} \rceil) \rfloor}) \rceil \tag{4}$$

dop_i : *delay of functional unit op_i or minimum available latency of pipelined functional unit*[5].

In equation (4), $\lceil dop_i / T_c \rceil$ is the number of synchronized cycle needed by module op_i. For the slower functional unit which could be multicycled, i.e. the $\lceil dop_i / T_c \rceil$ could be greater than 1, and the number of control steps $S_c (= \lfloor l / T_c \rfloor)$ is also increased by employing a fast T_c. But, for a fast functional unit which is still single cycled and due to the number of control steps S_c increased by a fast T_c, the necessary number of this fast functional unit to implement the data path is reduced, $m_{op_i} = \lceil N_{op_i} / S_c \rceil$. For the fast functional unit always occupies a larger area, the fast T_c not only speeds up the system throughput but also decreases the area of a data path. Then the total operator area of a data path is:

$$Area(operator) = \sum_{op_i=0}^{n} m_{op_i} * aop_i \tag{5}$$

aop_i : *area of functional unit op_i*.

Refer Equation 3 to Equation 4, and then taking into the above equation, which is shown as:

$$Area(operator) = \sum_{op_i=0}^{n} \{ \lceil (\frac{N_{op_i}}{\lfloor (\lfloor \frac{l}{T_c} \rfloor / \lceil \frac{dop_i}{T_c} \rceil) \rfloor}) \rceil \} * aop_i \quad (6)$$

2.4.2 Controller Consideration

However, the clock cycle T_c affects both the data path and the complexity of the controller. The number of control steps S_c is increased while T_c is decreased. In this way, the size and complexity of controller must also be taken into account. Assume that each control step corresponds to a microword in the controller. In each microword there are three control field to implement each control step; s bits are required to specify the next control step (the next state), o bits are required to specify the microoperation to be performed, and d bits are required to control the data path. The total amount of microwords multiplied by the width of each microword:

$$\mu C = S_c * (s + o + d) \quad (7)$$

Now we examine the details of each field of the μC. The next state field represents the number of control steps of the controller. The simple relationship is given by:

$$s = \lceil \log_2 S_c \rceil$$

The number of bits, d, in the data path field is required to control the data path. Since the combinational functional units are "free running", no microcoded control is necessary. Instead, the number of bits to control the data path comes solely from the number of registers and multiplexers. The estimation of the number of registers and multiplexers for a pipelined design has been proposed by [6]. The microoperation field, o, which is dependent on the application, for example, 4 bits can specify sixteen various kinds of microoperations. If each bit in the μC is $W(c)$ mil^2, then the total area of controller is shown as:

$$Area(controller) = \lfloor \frac{l}{T_c} \rfloor * (s + o + d) * W(c) \quad (8)$$

The hardware cost of the module selection considering the above factor is $cost = area(datapath) + area(controller)$. The goal is to minimum the total chip area. In the data path the area of operators and registers is taken into account. The overall hardware cost for our module selection is shown below:

$$cost = \sum_{op_i=0}^{n} \lceil (\frac{N_{op_i}}{\lfloor (\lfloor \frac{l}{T_c} \rfloor / \lceil \frac{dop_i}{T_c} \rceil) \rfloor}) \rceil * aop_i + r * r_{area} + \lfloor \frac{l}{T_c} \rfloor * (s+o+d) W(c) \quad (9)$$

The objective cost function also can be multiplied by an overhead factor to account for interconnect and layout overhead.

design	type & number of operation	the critical paths
FIR	(+) : 15. (*) : 8.	8(+) + 1(*)
random DFG	(+) : 26. (*) : 8. (−) : 4.	11(+) + 3(*)
		10(+) + 2(*) + 1(−)

Table 2: Description of Design Examples

2.4.3 Algorithm

```
INPUT:1.a data flow graph :
        2.a design library :
        3.constraints: throughput, latency, area or $F_{cost}$.
OUTPUT:optimal module set and $T_c$ for synthesis.
for n various types of operation nodes
  find all possible candidate module sets: $s_0$, $s_1$ ..,$s_m$ ;
for $s_i$ in ($s_0$, $s_1$ ..,$s_m$) {
  find local optimal $T_c$ of $s_i$ {
    calculate the number FU of $op_1..op_n$;
    calculate the numberofcycle FU of $op_1..op_n$;
    calculate the number of registers;
    calculate the number of multiplexor inputs;
    calculate the cost function;
    if (constraints)
      record the minimum cost module set;
  }
}
```

The complexity of this algorithm is $O(m*K)$, for K denotes the number of the local optimal clock cycle for module set s_i.

3 Experiment and Results

The described algorithm is implemented in C language on sun3/110. The results of two examples are reported. The first is the FIR design [4], the second is the random data flow graph in figure 4 which shows a similar topology as the fifth order elliptic filter design [2]. Table 3 describes these examples in terms of operation types, number of nodes of each types and the critical paths.

The module library is shown in table 1, there are four adders, three multipliers and three subtractors in the library. In the FIR example, only addition and multiplication operation nodes in DFG, there are four by three combinations of module sets. But, the adder ax is a redundance of a1 and the module set with ax can be replace by a1 resulting in a better performance. In this way, there are three by three combinations of candidated module sets. By scanning of the local optimal values of the clock cycle times, a selection of a optimal clock cycle and the clock schemes for the optimal module set are derived. Figure 5 shows the comparison of the flexible clock cycle time and the single cycle operation curves. The flexible selection of the clock cycle time shows a much better design space exploration than the design curve that all operation are single cycled. Figure 6 shows the result of the random data flow graph example. In this example, there are three types of operation, addition, multiplication and subtraction. The most critical paths list in table 2 and show in figure 4 in heavy lines. The two examples are illustrated this program can provide a good design space exploration and a feasible estimation of design performance.

4 Conclusion

The algorithm presented in this paper select an optimal module set with an optimal clock cycle time by a heuristic cost function which both considers the area of the data path and that of the controller, and the flexible choice of the clock scheme also improve the exploration of the design space. The clock schemes of single-cycle, multi-cycle and structural pipeline modules are formulated. The work can be appled on both ordinary or functional pipelining synthesis. Before actual synthesis, this program can also provide a good estimation of the design performance.

References

[1] Forrest D. Brewer and Daniel D. Gajski, "Knowledge Based Control in Micro-Architecture Design," *Proc. 24th Design Automation Conference*, pp 274-277, June 1987.

[2] Baher S. Haroun and Mohamed I. Elmasry, "Architectural Synthesis for DSP Silicon Compilers," *IEEE Trans. CAD.*, pp 413-447, April 1989.

[3] R. Jain, A. C. Parker, and N. Park, "Module Selection for Pipelined Synthesis," *Proc. 25th Design Automation Conference*, June 1988.

[4] N. Park and A. C. Parker, "Sehwa: a Software Package for Synthesis of Pipelines from Behavior Specifications," *IEEE Trans. CAD.*, Vol. 7, No. 3, March 1988.

[5] P. M. Kogge, *The Architecture of Pipelined Computers*, New York, McGram-Hill, 1981.

[6] R. Jain, A. C. Parker and N. Park, "Predicting Area-Time Tradeoffs for Pipelined Design," *Proc. 24th Design Automation Conference*, June 1987.

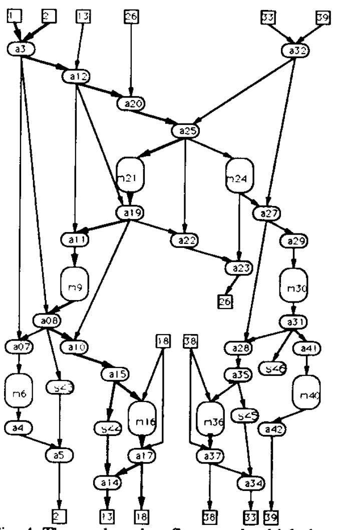


Fig. 4. The random data flow graph which the most critical path is in heavy line. The node index 'a' represents'addition', 's' represents 'subtraction', and 'm' represents 'multiplication'.

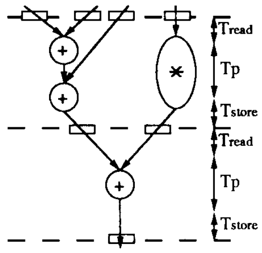


Fig. 1 The control step's timing of a fragment of the DFG.

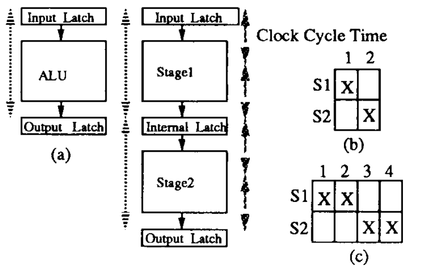
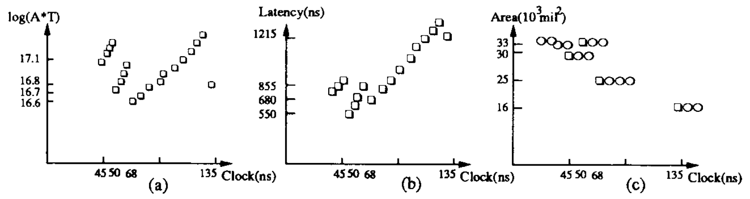


Fig. 2 Hardware model example. (a) Non-pipelined. (b) Pipelined single cycle latency. (c) Pipelined multi-cycle latency.



Module Library:

Module Name	Operation	Area *mil* ²	Delay *nS*
A1	Addition	2000	40
M1	Multiplication	6200	125
Reg	Register	500	10
Cont	PLA-controller	30 per uC	min20

Local Optimal Value of Clock Cycle Time:

Latency	Step	Clock	Cycle(A)	Cycle(M)	Num(A)	Num(M)	Num(Reg)	Area	A(data)	A(cont)	log(A*T)
550	11	50	1	3	2	3	10	33540	27600	5940	16.7304
1215	9	135	1	1	2	1	6	16980	13200	3780	16.8423
680	10	68	1	2	2	2	8	25200	20400	4800	16.6567
855	19	45	2	3	2	2	8	30090	20400	9690	17.0630

Fig. 3. The effect of the Clock Cycle time selection of a candidate module set on area-performance product.

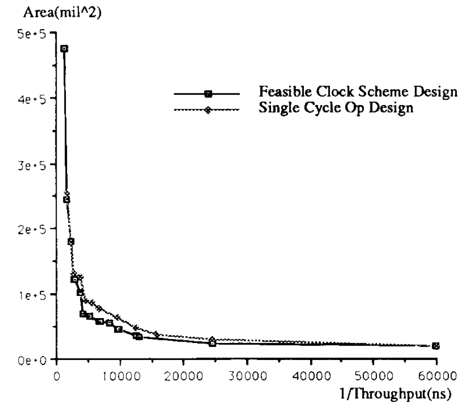


Fig. 5. The results of module and clock cycle selection for FIR filter . The feasible clock scheme design vs. single cycle op design curves.

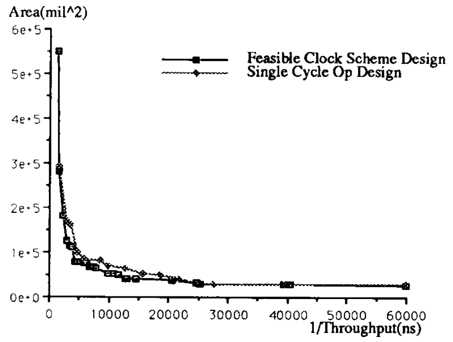


Fig. 6. The results of module and clock cycle selection for the random data flow graph example.The feasible clock scheme design vs. single cycle op design curves.