THE EFFECT OF REGISTER-TRANSFER DESIGN TRADEOFFS ON CHIP AREA AND PERFORMANCE

John J. Granacki
Alice C. Parker

DIGITAL INTEGRATED SYSTEMS CENTER REPORT
DISC/82-10

DEPARTMENT OF ELECTRICAL ENGINEERING SYSTEMS
UNIVERSITY OF SOUTHERN CALIFORNIA
LOS ANGELES, CALIFORNIA 90089-0781

December 1982

This research was supported by Army Research Office Grant DAAG29-80-K-0083 and by a Hughes Aircraft Corp. Fellowship.

THE EFFECT OF REGISTER-TRANSFER DESIGN TRADEOFFS ON CHIP AREA AND PERFORMANCE

John J. Granacki
Alice C. Parker

DIGITAL INTEGRATED SYSTEMS CENTER REPORT

DISC/82-10

DEPARTMENT OF ELECTRICAL ENGINEERING SYSTEMS
UNIVERSITY OF SOUTHERN CALIFORNIA
LOS ANGELES, CALIFORNIA 90089-0781

December 1982

This research was supported by Army Research Office Grant DAAG29-80-K-0083 and by a Hughes Aircraft Corp. Fellowship.

Table of Contents

1. Abstract

This paper describes an experiment to determine how register-transfer tradeoffs affect the resultant silicon area and performance of layouts. Six register-transfer level designs, each performing the same function with different register transfer level structure, were implemented using a library of CMOS/SOS standard cells, and an automatic placement and routing program. The consumption of area and the critical path timing were calculated for each design. The results show that register- transfer design variations do produce changes in area and performance at the layout level. However, optimistic timing analysis at the RT level, variations in relative storage timing across technologies, and omission of fanout and path length delays altered the resultant area and timing from that predicted at the RT level.

2. Introduction

One of the major differences between designing digital circuits with discrete components and designing an integrated circuit layout is the difficulty of predicting the impact of each design decision at the functional level on the resultant performance and "cost"(which we will measure in area) of the final design. While some area estimators exist for regular structures [Thompson 80], and channel width estimators have been researched ([Syed 82], [Donath 81]), a model of area consumption as a function of design tradeoffs has not been proposed.

In order to automatically synthesize register-transfer(RT) designs from higher-level specifications, estimates of area consumption and critical path timing are needed. Such estimates must take into account routing area, the effect of signal buffering on area and timing, and the effects of fanout and of path lengths on performance.

This paper describes an experiment designed to provide data for predicting area consumption and performance at the register-transfer level. Six register-transfer level designs were used in the experiment.[1] All six of the designs represented data paths designed to perform the same function, but the structure of each design varied. The RT designs were synthesized automatically ([Hafer 83], [Hafer 81]), and estimates of "cost" and performance were done by the synthesis program.

Each of the RT designs was manually implemented using a library of CMOS/SOS standard cells, and then automatically placed and routed. Area consumption and critical path timing were then computed and compared to the RT level estimates.

The experiment is described in Section 3 and the results are discussed in Section 4. Section 5 gives an analysis of the data, and conclusions are drawn in Section 6.

3. The Experiment

There are virtually an infinite number of register-transfer structures which can be chosen to implement a given function, most of which are inferior designs. However, some data on register-transfer implementations have already been produced by Hafer, and six RT implementations of the same function have been shown to be non-inferior using Hafer's estimates of cost and speed. Thus, we chose these six as examples of register-transfer tradeoffs. The function implemented by these is shown in Figure 1, reproduced from [Hafer 83].

Figure 2 (a) through (e) show five of the six RT implementations. Design #5 had to be discarded due to the timing assumption used during RT synthesis which allowed inputs to be changed before an operator had finished computing

[1] One later had to be discarded due to timing problems which will be described in section 3.

results based on the inputs. This overly optimistic assumption allowed design 5 to be produced, but it could not be made to function correctly in practice.

Design #1 uses a single ALU to do all operations, while #2 uses an adder to do the adds and an ALU for the subtracts. The output of the ALU is connected to the input of the adder through a multiplexer, so that no storage of intermediate values is necessary here. #3 is similar, except that the operators are not chained, and so intermediate storage is necessary. In design #4, the adder does both adds, while the subtracts are done by a subtracter and the ALU. Design #6 contains two adders and two subtracters, which perform the two adds and subtracts. In all designs, the outputs are stored in registers. The RT synthesis program assumed that the inputs were available for 100 nanoseconds. This time had to be lengthened due to the implementation technology. Finally, the RT designs assumed 16 bit data widths, but this was reduced to 4 bits for the layout of each design.

Each of the designs was manually implemented by replacing each RT element with a set of standard cells and a specification of their interconnections. Multiplexers were implemented with transmission gates. The ALU is essentially an add/subtract unit constructed in the same manner as the adder and subtracter, and therefore had virtually the same performance but used slightly more area than the adder and subtracter. (Logic functions, unused in this design, were omitted from the ALU, changing its characteristics from the ALU data available to the RT synthesis program).

Signals were buffered as necessary, and all implementations used the same pad cells. The choice of cells to implement each register or functional unit was uniform across the designs in order to isolate the effects of the RT tradeoffs from implementation tradeoffs.

The designs were then placed and routed using the MP2D program run on a VAX 11/780. I/O pads were allowed to be placed uniformly around the periphery but there were no other constraints placed on their location. Furthermore, no initial placement information was provided for any of the internal cells, and no critical timing paths were specified to the program. This was done to avoid biasing the implementations in any way.

4. Results

The results of the experiment can be divided into the area consumption of each design, and the critical path delay. Because the placement and routing package supplies detailed area statistics, the area used for each RT element, for routing, for I/O pads and unused area can be tabularized directly from the program outputs.

Table 1 shows the area statistics. The internal overhead refers to buffering of signals. The wiring category includes all routing of signals, including feedthroughs, and power and ground routed to the cell rows. The subtotals for each design represent all area which could be accounted for, except for the scribe line.

Some trends in the data are immediately obvious. The operator area increases from design #1 to #6, except for a minor dip at #3, and the wiring area follows the same trend. Total used area, total chip area and unused area all increase in the same way. An interesting category is the multiplexer area, which varies randomly over the designs. This is an interesting result, since multiplexing is not even considered in the RT level cost analysis.

Performance results are given in Tables 2 and 3. Table 2 shows operator delays across the five designs. The uniformity of the delays should be noted. Table 3 gives critical path delays for the five designs. The first entry is the delay predicted by the RT synthesis program. The second entry is the critical path delay for each design if fixed gate delays are considered. No allowance is made here for variable delays due to capacitance of loads. The third entry is the path delay with capacitive loading considered. This was computed for each gate in the critical path by using the total capacitance values supplied by the placement and routing software for each signal output, along with the delay per picofarad for each gate and its associated propagation delay. Due to the variable nature of the loads as a function of fanout and path lengths, multiple potential critical paths had to be examined to locate the worst case delays for each design. The fourth entry is the number of operations in the critical path, as predicted by the RT synthesis software. The fifth entry shows the actual number of operations in the critical path when the critical paths in the layouts were examined. The sixth and final entry shows the number of register stores in the critical paths. The number of operations in the critical paths varied from that predicted at the RT level because the optimistic timing assumption referred to earlier allowed the RT analysis to consider much greater overlapping of functional units than that actually possible in the implementation. Partially offsetting this difference is the fact that designs #4 and #6 have operator-operator connections. This configuration gives a critical path delay through both operators which is less than the sum of the delay through each, since the second operator can begin as soon as the lowest order bits feeding it have been produced. Thus operator overlap occurs, but the reasoning behind it is different from that at the RT level.

5. Analysis

This section touches on the relationship between performance and silicon area, and factors affecting each. In addition, we discuss the relationship of these results to predictions made by the RT synthesis software. Finally, reasons for the discrepancies between the predictions and the measured data are enumerated.

Figure 3 shows the relationship between area and performance for the 5 designs. With the exception of design #3, we see a clear inverse relationship between area and performance. Design #3 performed better than the area figures would have predicted. We see that the sharing of the ALU in design #1 produces the smallest design, and the slowest, since there is no parallelism. Adding a second operator in design #2 allows some parallelism. Delay decreases about as fast as total area increases here. Note that a rearrangement of interconnections (and the substitution of a subtracter for the ALU, which does not significantly change the statistics) to allow more parallelism decreases delay another 25% without increasing area consumption. Design #4 adds a third operator to speed up the operation but, due to the data precedence relationships, this operator has no effect. In fact, the adder cannot start its second operation until the ALU is finished with its subtract, and so the net effect is that of somewhat more than two operations in series. In fact, the delay would look very similar to design #2 except for the fact that the ALU can start as soon as the least significant result bits from the adder and subtracter are available, overlapping operation significantly. The sixth implementation shows about 15% increase in total area over #4 and 33% over design #3, but the performance gain is about 33%. This is due entirely to the fact that the chaining of operations now allows parallelism to exist on the bit level, so that the first and second adds and subtracts overlap significantly. In fact, due to data precedence relationships, the designs go from serial (#1) to parallel (#3), and the only further performance gains are due to bit-level parallelism.

The analysis shows that the operators affect area and delay far more than the registers or the multiplexers. Moreover, design #3 is heavily multiplexed, but is a non-inferior design. Due to the CMOS technology implementation of multiplexers, performance did not degrade and cost did not increase with increases in switching as was expected.

Buffer overhead seems to be a relatively constant area factor, and I/O pad area is constant. Wiring area is proportional to functional area, and to total area. Unused(unuseable?) area increases as used area increases.

Figure 3 also shows the (normalized) cost-speed curve produced by the RT predicter. While the two curves are very similar, the reasons for their shape are different. First of all, the optimistic timing assumption the RT software uses allows operator overlap to a greater extent than is realizable. Thus, design #2 and design #4 appear slightly faster than they actually are.

The implementation of 4 bit data paths instead of 16 bits has exaggerated the effect of interconnection delays and minimized the effects of carry propagation through the operators. Thus, fanout delays and path length delays which show up with some significance in design #6 might be less significant in the full 16 bit implementation.

Finally, the data the RT predicter used had proportionately higher storage delays, which made design #3 look slower than it really was, due to the two stores in the critical path. The RT predicter also had ALU data which indicated a cheaper, slower ALU than the one implemented.

Figure 4 shows the variation in the ratio of functional area to wiring area as functional area increases. Wiring consumes a larger portion of used area as functional area increases. Figure 5 shows normalized CPU times to place and route the five designs as a function of total chip area.

6. Conclusions

Because the entire experiment is based on CMOS standard cells, all of the conclusions given here are predicated on the use of these cells, and the associated placement and routing software. Any other layout methodology could not be predicted to manifest the RT tradeoffs in the same manner.

From the results shown, we can see that there is a tradeoff between area and delay. As the amount of parallelism increases, the area also increases.

However, there is a point where adding more hardware increases performance only because parallelism on the bit level can then be realized. This tradeoff could be considered to be as much an implementation tradeoff as an RT tradeoff.

The effects of routing and switching on total area appear to be well-behaved. We can hypothesize here that routing area can be predicted to increase as functional area increases, and at a slightly faster rate. We can also hypothesize that switching area can be considered to be a second-order effect.

The timing model used by the RT predicter, with the exception of the assumption about early removal of inputs, proved to be a reasonable predicter of delay. Fanout and path delays in design #6 became significant, however, and in VLSI designs would not be able to be ignored.

7. Acknowledgements

the authors are indebted to Lou Hafer for providing the RT designs.

REFERENCES

[Donath 81] Donath, W. and Mikhail, W.
Wiring Space Estimation for Rectangular Gate Arrays.
In *Proceedings VLSI81*, pages 301-312. 1981.

[Hafer 81] Hafer, L.
Automated Data-Memory Synthesis : A Formal Model for the Specification, Analysis and Design of Register-Transfer Level Digital Logic.
PhD thesis, Dept of Electrical Engineering, Carnegie Mellon University, Pittsburgh, Pa., May, 1981.

[Hafer 83] Hafer, L., and Parker, A.
A Formal Method for the Specification Analysis, and Design of Register-Transfer Level Digital Logic.
January, 1983.
To appear in IEEE Transactions on Computer-Aided Design.

[Syed 82] Syed, Z., El Gamal, A., and Breuer, M.
On routing for Custom Integrated Circuits.
In *Proceedings of the 19th Design Automation Conference*, pages 887-893. June, 1982.

[Thompson 80] Thompson, C.
A Complexity Theory for VLSI.
PhD thesis, Computer Science Department, Carnegie-Mellon University, 1980.

```
Criss.Cross :=

BEGIN

  ** Carriers **
  t1<0:15> ,
  t2<0:15> ,
   a<0:15> ,
   b<0:15> ,

  ** Activity **
  action :=
  ( t1 _a+b next
    t2 _a-b next
    a _t1+t2 next
    b _t1-t2 )

END
```

Figure 1(a): Criss.Cross ISPS Behavioral Description

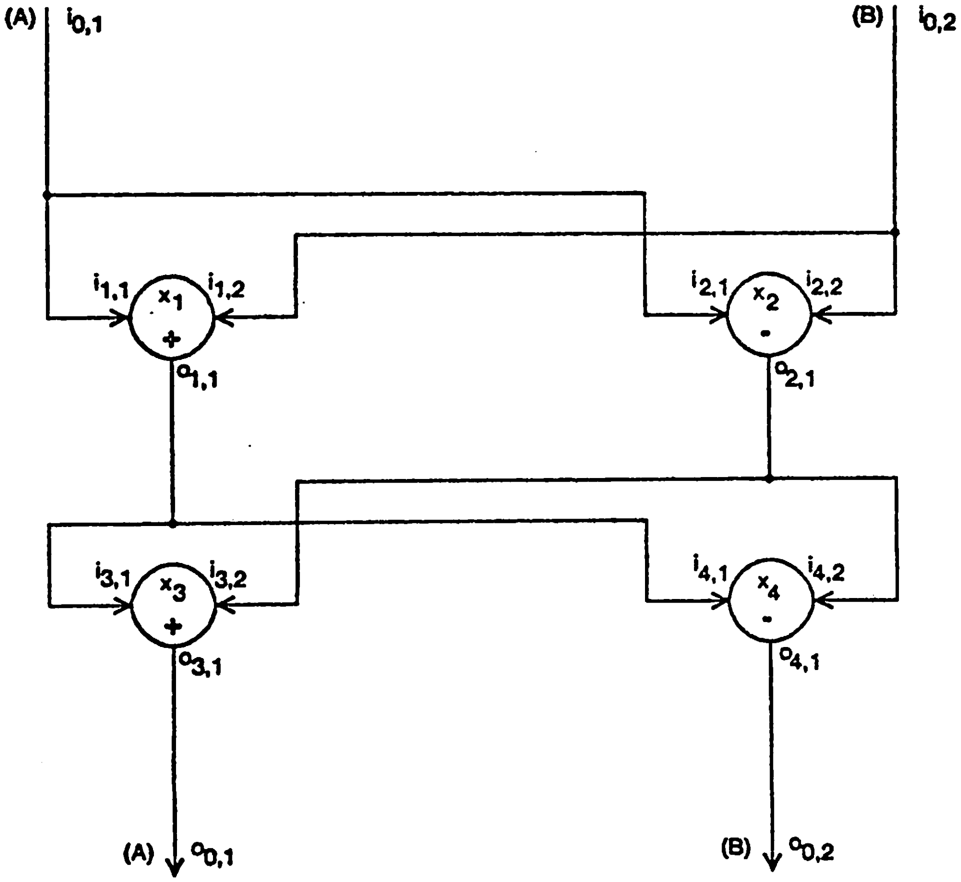

Figure 1 (b): Criss.Cross Data Flow Representation

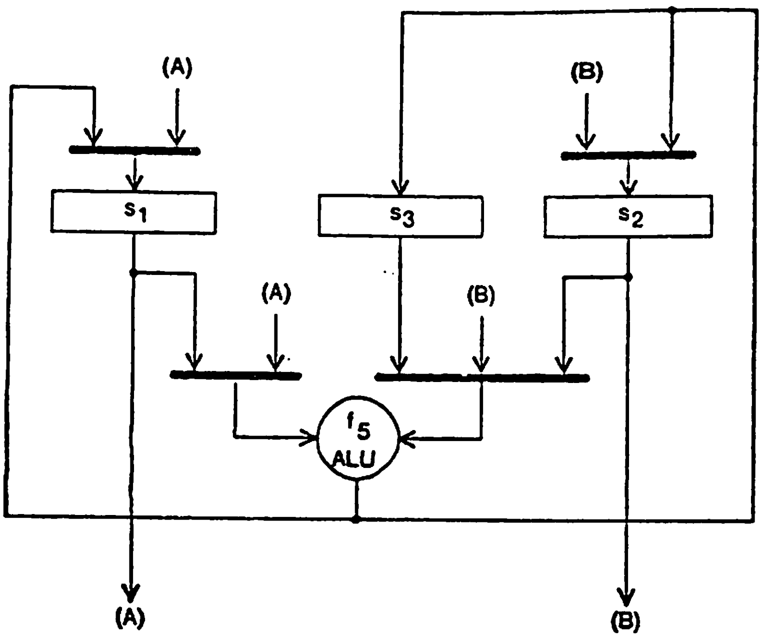

Figure 2(a) : Implementation #1 for Criss.Cross

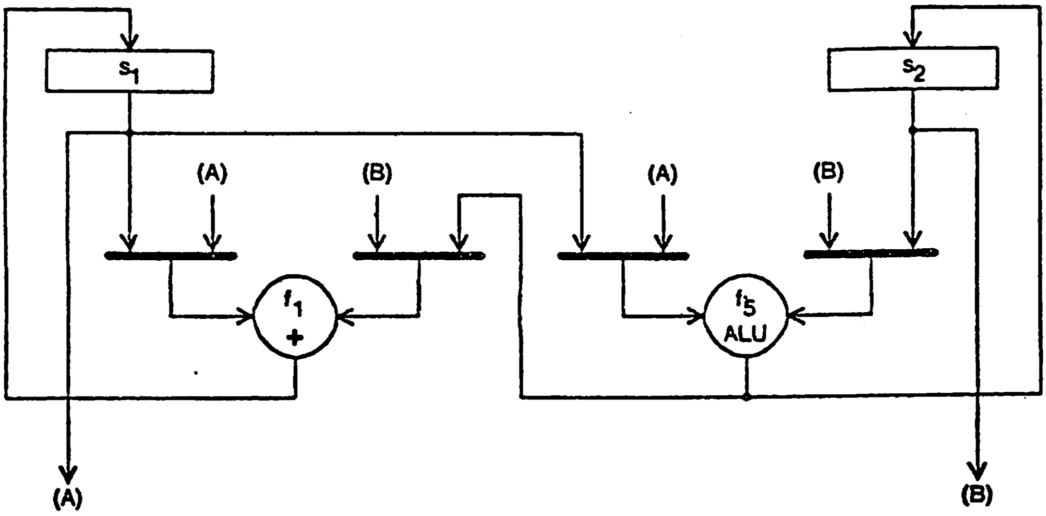

Figure 2(b) : Implementation #2 for Criss.Cross

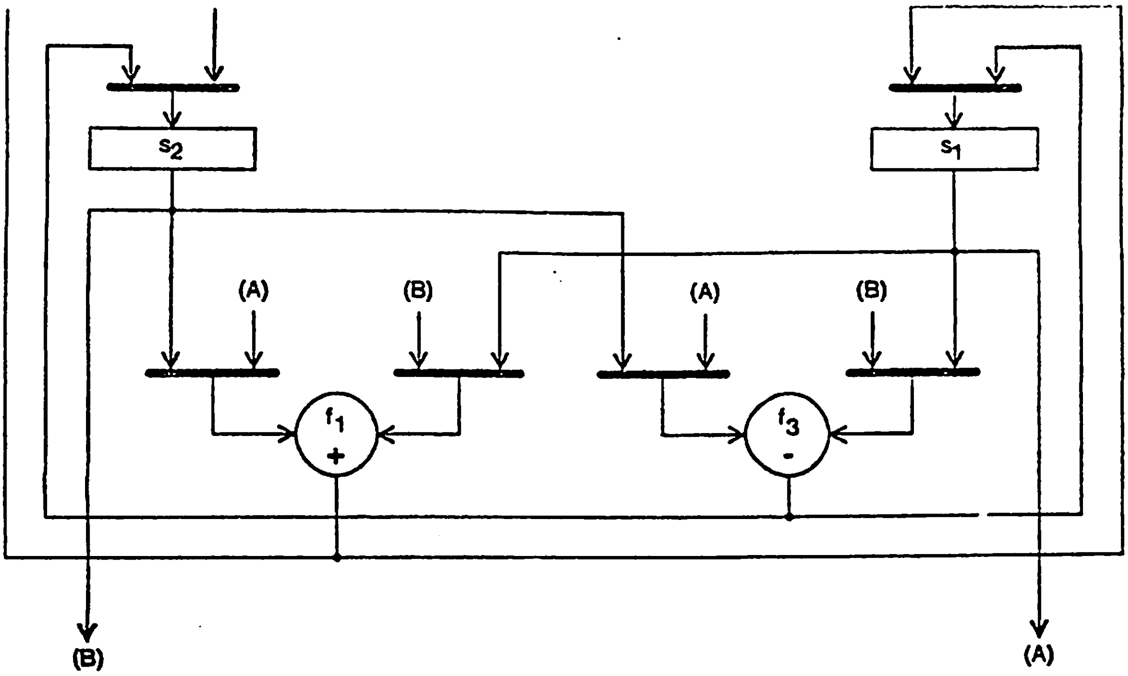

Figure 2(c) : Implementation # 3 for Criss.Cross

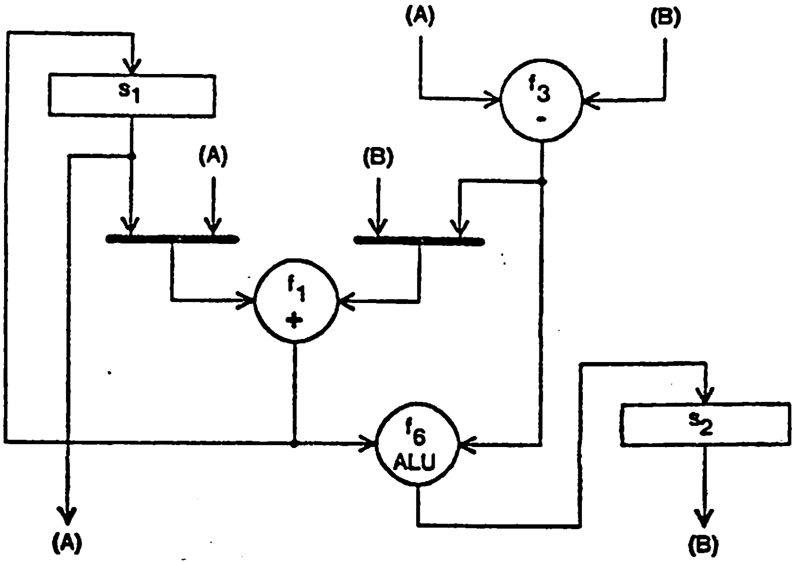

Figure 2(d) : Implementation # 4 for Criss.Cross

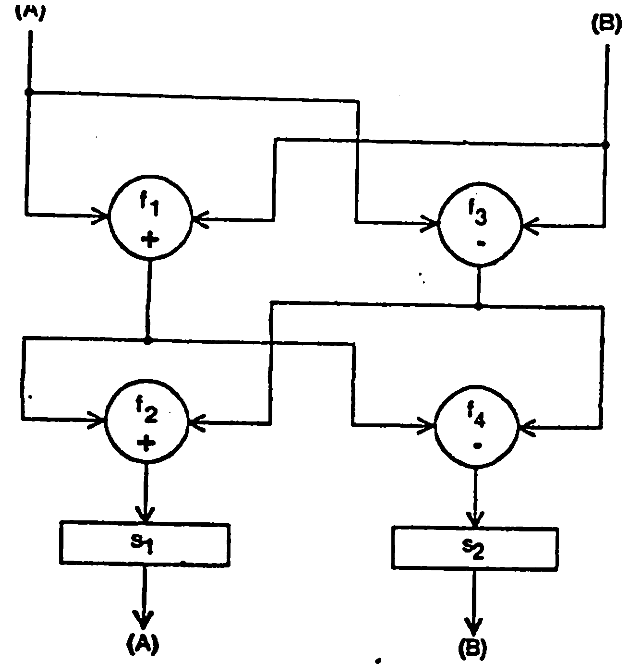

Figure 2(e): Implementation #6 for Criss.Cross

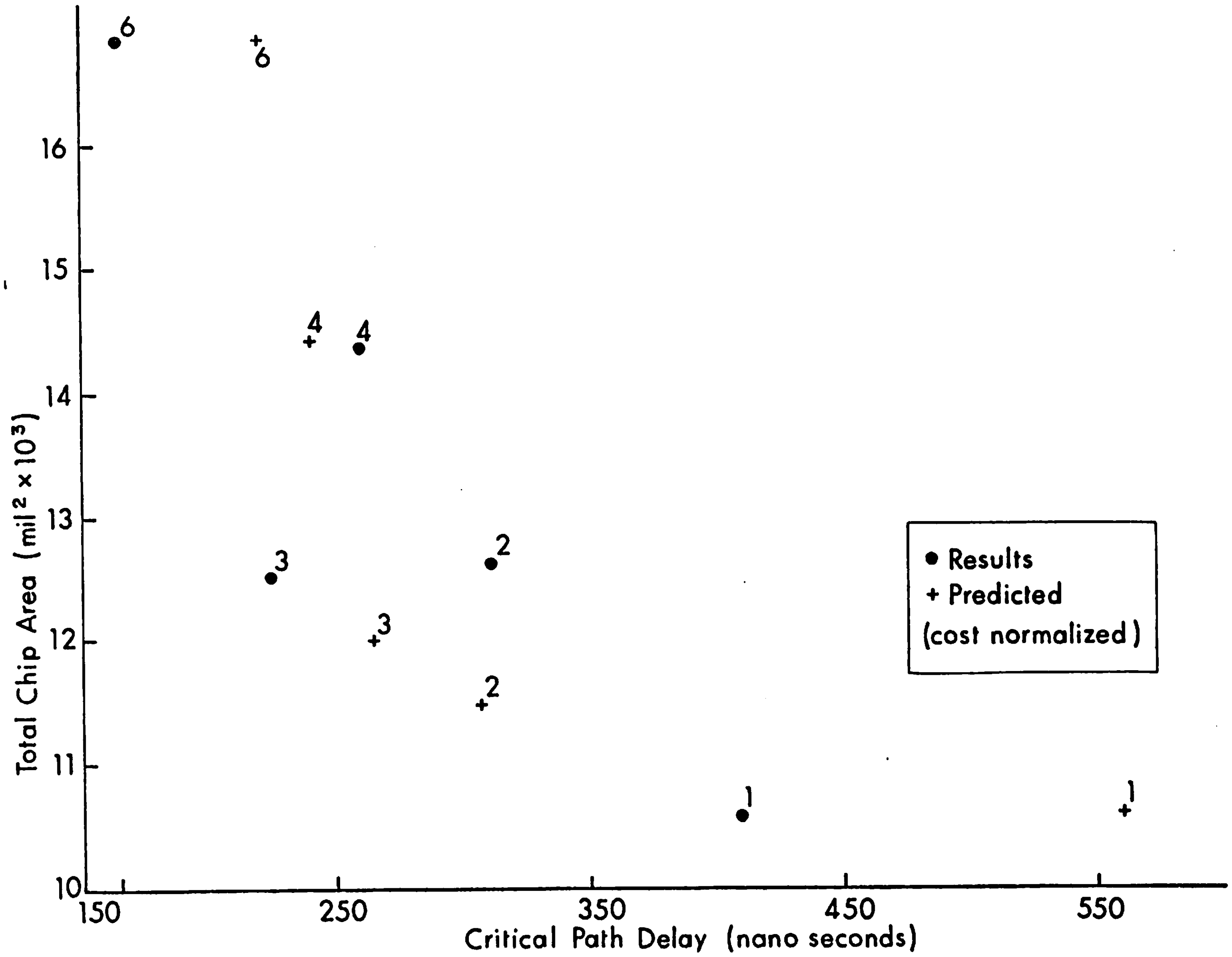

Figure 3. Design Space of Implementations and Predicted Values.

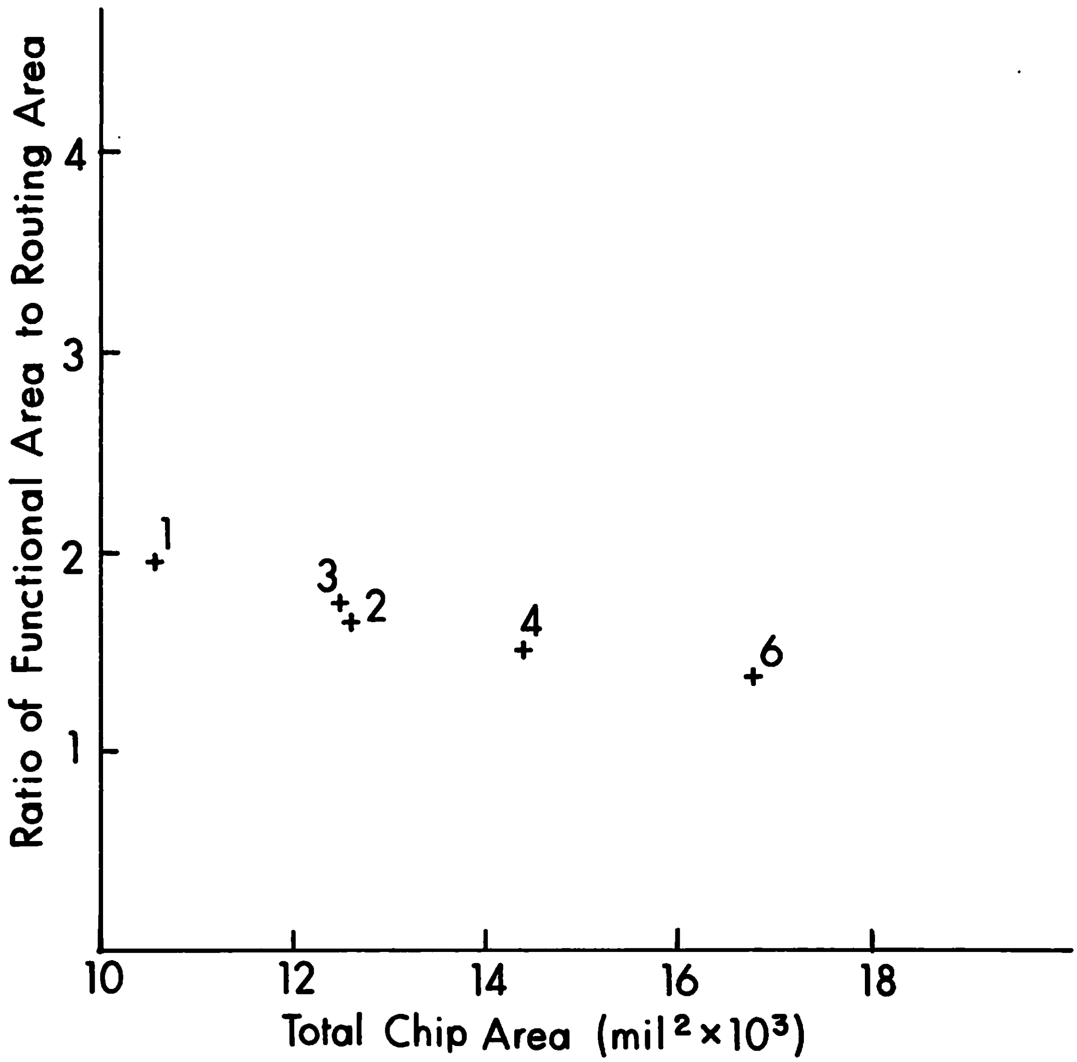

Figure 4. The Change in Proportion of Routing Area to Functional Area.

Crisscross	#1	#2	#3	#4	#6
Registers	665.3	443.5	443.5	443.5	443.5
Operators	987.8	1854.6	1733.6	2721.4	3467.2
Muxes	625.0	443.5	675.4	221.8	0.0
Internal Overhead	403.2	524.1	524.1	524.1	806.4
Input/ Output	2232.0	2232.0	2232.0	2232.0	2232.0
Wiring	2486.7	3304.1	3175.2	3985.0	5047.5
Sub Total	7400.0	8801.8	8783.0	10126.0	11996.6
Unused *	2188.0	2683.2	2657.0	3074.2	4867.4
Total	10593.0	12584.0	12535.0	14375.0	16864.0

* does not include area for scribe line

Table 1 : Area statistics for the 5 designs

Crisscross	#1	#2	#3	#4	#6
Hafer's RT Design	562	308	265	240	220
CMOS Design (no loads)	200	144	112	144	100
CMOS Design (gate loads &parasitic)	410.7	313.2	225.7	261.4	165.3
Ops in CP for Hafer's RT Design	4	3	2	2	2
Effective number of ops in CP	4	3	2	2.3	1.3
Storage elements in CP	2	1	2	1	1

Table 2 : Critical path analysis (delays in nanoseconds)

Crisscross	#1	#2	#3	#4	#6
f5(ALU) Muxes off Subtract	82.2	80.3			
f5(ALU) Muxes off Add	82.2				
f1(ADD)		83.5		89.5	
f3(Subtract)			79.8		89.0 88.1

Table 3 : Delays for operators in the critical paths

Crisscross	#1	#2	#3	#4	#6
Hafer's RT Design	562	308	265	240	220
CMOS Design (no loads)	200	144	112	144	100
CMOS Design (gate loads &parasitic)	410.7	313.2	225.7	261.4	165.3
Ops in CP for Hafer's RT Design	4	3	2	2	2
Effective number of ops in CP	4	3	2	2.3	1.3
Storage elements in CP	2	1	2	1	1

Table 2 : Critical path analysis (delays in nanoseconds)

Crisscross	#1	#2	#3	#4	#6
f5(ALU) Muxes off Subtract	82.2	80.3			
f5(ALU) Muxes off Add	82.2				
f1(ADD)		83.5		89.5	
f3(Subtract)			79.8		89.0 88.1

Table 3 : Delays for operators in the critical paths

THE EFFECT OF REGISTER-TRANSFER DESIGN TRADEOFFS ON CHIP AREA AND PERFORMANCE

John J. Granacki
Alice C. Parker

DIGITAL INTEGRATED SYSTEMS CENTER REPORT

DISC/82-10

DEPARTMENT OF ELECTRICAL ENGINEERING SYSTEMS
UNIVERSITY OF SOUTHERN CALIFORNIA
LOS ANGELES, CALIFORNIA 90089-0781

December 1982

This research was supported by Army Research Office Grant DAAG29-80-K-0083 and by a Hughes Aircraft Corp. Fellowship.

Table of Contents

1. Abstract

This paper describes an experiment to determine how register-transfer tradeoffs affect the resultant silicon area and performance of layouts. Six register-transfer level designs, each performing the same function with different register transfer level structure, were implemented using a library of CMOS/SOS standard cells, and an automatic placement and routing program. The consumption of area and the critical path timing were calculated for each design. The results show that register- transfer design variations do produce changes in area and performance at the layout level. However, optimistic timing analysis at the RT level, variations in relative storage timing across technologies, and omission of fanout and path length delays altered the resultant area and timing from that predicted at the RT level.

2. Introduction

One of the major differences between designing digital circuits with discrete components and designing an integrated circuit layout is the difficulty of predicting the impact of each design decision at the functional level on the resultant performance and "cost"(which we will measure in area) of the final design. While some area estimators exist for regular structures [Thompson 80], and channel width estimators have been researched ([Syed 82], [Donath 81]), a model of area consumption as a function of design tradeoffs has not been proposed.

In order to automatically synthesize register-transfer(RT) designs from higher-level specifications, estimates of area consumption and critical path timing are needed. Such estimates must take into account routing area, the effect of signal buffering on area and timing, and the effects of fanout and of path lengths on performance.

This paper describes an experiment designed to provide data for predicting area consumption and performance at the register-transfer level. Six register-transfer level designs were used in the experiment.[1] All six of the designs represented data paths designed to perform the same function, but the structure of each design varied. The RT designs were synthesized automatically ([Hafer 83], [Hafer 81]), and estimates of "cost" and performance were done by the synthesis program.

Each of the RT designs was manually implemented using a library of CMOS/SOS standard cells, and then automatically placed and routed. Area consumption and critical path timing were then computed and compared to the RT level estimates.

The experiment is described in Section 3 and the results are discussed in Section 4. Section 5 gives an analysis of the data, and conclusions are drawn in Section 6.

3. The Experiment

There are virtually an infinite number of register-transfer structures which can be chosen to implement a given function, most of which are inferior designs. However, some data on register-transfer implementations have already been produced by Hafer, and six RT implementations of the same function have been shown to be non-inferior using Hafer's estimates of cost and speed. Thus, we chose these six as examples of register-transfer tradeoffs. The function implemented by these is shown in Figure 1, reproduced from [Hafer 83].

Figure 2 (a) through (e) show five of the six RT implementations. Design #5 had to be discarded due to the timing assumption used during RT synthesis which allowed inputs to be changed before an operator had finished computing

[1] One later had to be discarded due to timing problems which will be described in section 3.

results based on the inputs. This overly optimistic assumption allowed design 5 to be produced, but it could not be made to function correctly in practice.

Design #1 uses a single ALU to do all operations, while #2 uses an adder to do the adds and an ALU for the subtracts. The output of the ALU is connected to the input of the adder through a multiplexer, so that no storage of intermediate values is necessary here. #3 is similar, except that the operators are not chained, and so intermediate storage is necessary. In design #4, the adder does both adds, while the subtracts are done by a subtracter and the ALU. Design #6 contains two adders and two subtracters, which perform the two adds and subtracts. In all designs, the outputs are stored in registers. The RT synthesis program assumed that the inputs were available for 100 nanoseconds. This time had to be lengthened due to the implementation technology. Finally, the RT designs assumed 16 bit data widths, but this was reduced to 4 bits for the layout of each design.

Each of the designs was manually implemented by replacing each RT element with a set of standard cells and a specification of their interconnections. Multiplexers were implemented with transmission gates. The ALU is essentially an add/subtract unit constructed in the same manner as the adder and subtracter, and therefore had virtually the same performance but used slightly more area than the adder and subtracter. (Logic functions, unused in this design, were omitted from the ALU, changing its characteristics from the ALU data available to the RT synthesis program).

Signals were buffered as necessary, and all implementations used the same pad cells. The choice of cells to implement each register or functional unit was uniform across the designs in order to isolate the effects of the RT tradeoffs from implementation tradeoffs.

The designs were then placed and routed using the MP2D program run on a VAX 11/780. I/O pads were allowed to be placed uniformly around the periphery but there were no other constraints placed on their location. Furthermore, no initial placement information was provided for any of the internal cells, and no critical timing paths were specified to the program. This was done to avoid biasing the implementations in any way.

4. Results

The results of the experiment can be divided into the area consumption of each design, and the critical path delay. Because the placement and routing package supplies detailed area statistics, the area used for each RT element, for routing, for I/O pads and unused area can be tabularized directly from the program outputs.

Table 1 shows the area statistics. The internal overhead refers to buffering of signals. The wiring category includes all routing of signals, including feedthroughs, and power and ground routed to the cell rows. The subtotals for each design represent all area which could be accounted for, except for the scribe line.

Some trends in the data are immediately obvious. The operator area increases from design #1 to #6, except for a minor dip at #3, and the wiring area follows the same trend. Total used area, total chip area and unused area all increase in the same way. An interesting category is the multiplexer area, which varies randomly over the designs. This is an interesting result, since multiplexing is not even considered in the RT level cost analysis.

Performance results are given in Tables 2 and 3. Table 2 shows operator delays across the five designs. The uniformity of the delays should be noted. Table 3 gives critical path delays for the five designs. The first entry is the delay predicted by the RT synthesis program. The second entry is the critical path delay for each design if fixed gate delays are considered. No allowance is made here for variable delays due to capacitance of loads. The third entry is the path delay with capacitive loading considered. This was computed for each gate in the critical path by using the total capacitance values supplied by the placement and routing software for each signal output, along with the delay per picofarad for each gate and its associated propagation delay. Due to the variable nature of the loads as a function of fanout and path lengths, multiple potential critical paths had to be examined to locate the worst case delays for each design. The fourth entry is the number of operations in the critical path, as predicted by the RT synthesis software. The fifth entry shows the actual number of operations in the critical path when the critical paths in the layouts were examined. The sixth and final entry shows the number of register stores in the critical paths. The number of operations in the critical paths varied from that predicted at the RT level because the optimistic timing assumption referred to earlier allowed the RT analysis to consider much greater overlapping of functional units than that actually possible in the implementation. Partially offsetting this difference is the fact that designs #4 and #6 have operator-operator connections. This configuration gives a critical path delay through both operators which is less than the sum of the delay through each, since the second operator can begin as soon as the lowest order bits feeding it have been produced. Thus operator overlap occurs, but the reasoning behind it is different from that at the RT level.

5. Analysis

This section touches on the relationship between performance and silicon area, and factors affecting each. In addition, we discuss the relationship of these results to predictions made by the RT synthesis software. Finally, reasons for the discrepancies between the predictions and the measured data are enumerated.

Figure 3 shows the relationship between area and performance for the 5 designs. With the exception of design #3, we see a clear inverse relationship