Algorithm for Hardware Allocation in Data Path Synthesis

- *SOURCES:*
  - **Srinivas Devadas**
  - Duerat
  - **Richard Newton**

Abstract:

- In this paper, authors present simulated-annealing-based algorithm for simultaneous cost/resource constrained allocation of registers ,ALU, and interconnect in data path synthesis
- The authors also present how they developed this algorithm, also compare this algorithm with other algorithms. Examples and result are presented
- Limitation and Future work is discussed.
- Conclusion of this paper is that the new algorithm can achieved excellent solution in data path synthesis and can be extended to synthesis pipelined data path.

Data Path Synthesis by Tseng and Siewiorek

- Goal of data path synthesis step in behavioral synthesis is to produce RT level hardware design
- In hard allocation,both the schedule of operations and numbers of computational /storage unit need to be decided
- Other approaches

1.DAA, for general purpose computer data path

2. FACET,by Tseng and Siewiorek, is automatic data path synthesis program, but the entire design space is not explored

3. USC MAHA, by Gircyzc,force-directed scheduling, local minimum solution problem

4.CATHERAL and SEHWA systems. Concern mostly on scheduling issue, not hardware allocation

5. CMU,HERCULES,BECOME and BRIDGE are mentioned

- What is the deference between the new algorithm with others

  1.All the sub problems are handled simultaneously

  2.The optimization is completed global in nature

  3.Probabilistic hill-climbing algorithm is used, Min problem is avoided

- Main idea of the new algorithm

  To synthesis the data path corresponding to input data flow specification such as F(T,C). Find a near-optimal placement of microinstruction, thus find the data path configuration

Conditional Resource Sharing

Discuss modification to incorporate conditional resource sharing

- Introduction F(T,C)
- Input Description. Parallelism, sequentially and disjonitness
- Basic Allocation problem
- A sub problem. PLA multiple folding problem
- Formulation of Entire Data path problem

  C=p1*(#alu)+p2*(exe_time)+p3*(#reg)+p4*(#bus)

  T?

- The cost table
- Conditional Resource Sharing

  Mutual exclusion

Simulated-Annealing-based Solution

- Introduction. Hill climbing moves

  It can be used for combinatorial optimizations by specified a finite set of states and cost function, it contains many local search algorithm that seek to find optimal solution for our demands by allowing the algorithm to visit inferior solution to optimal/near optimal solution

- Generating New States

  Generates a new state/configuration, the new state is accept or rejected according to random acceptance rule by the parameters analogue to temperature in the physics process

  How is generated? 3 different ways

- Interchanging 2 code operation
- displacing a code operation from one location to another
- Interchanging the Var in symmetric operation

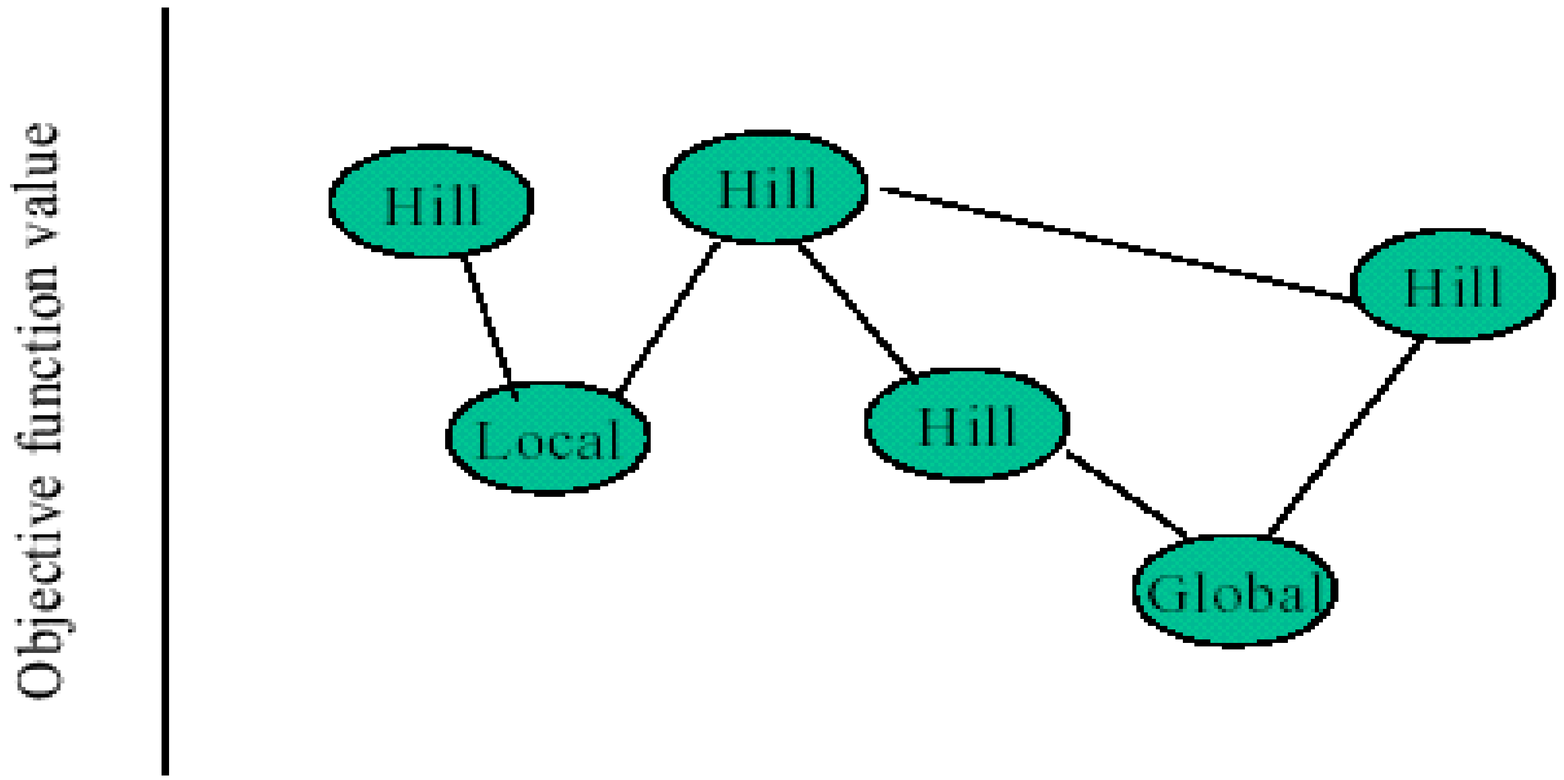


Figure 2.1: Locals, Hills and Globals

http://scholar.lib.vt.edu/theses/available/
etd-08042000-14590003/unrestricted/dianeetd.pdf

- The cost Function
  - The cost of ALU , registers
  - The cost of interconnect is estimate by estimating bus and links
- Hardware resource constraints

  hardware resource constraints is be incorporated by penalizing configuration which violated any of the constraints
- Execution Time Constraints. More in section 5

  Check the bound on time required by the data path to execute the code sequence. More detail in Section 6
- Stopping and Inner Loop Criteria

  stop when F(T,C) has not changed in certain number trying

Further Extensions

- Input description have equal delay, real life is not, but it is *possible to handle different delay.*
- Loops
  - Treat each loop as single operation

    Problem of this approach is all the iterations of loop are ways scheduled serially on single ALU
  - Full unwinding. All iteration expended into numbers of operation

    Too many iterations and not always feasible
  - Dynamic partial unwinding. All loops are represented as basic operation and its delays computed and ***<u>tagged and may be split up into sub-problems</u>***
  - Trade off delay and cost for single operation

    E.g.. Fast components to Slow components, vice versa

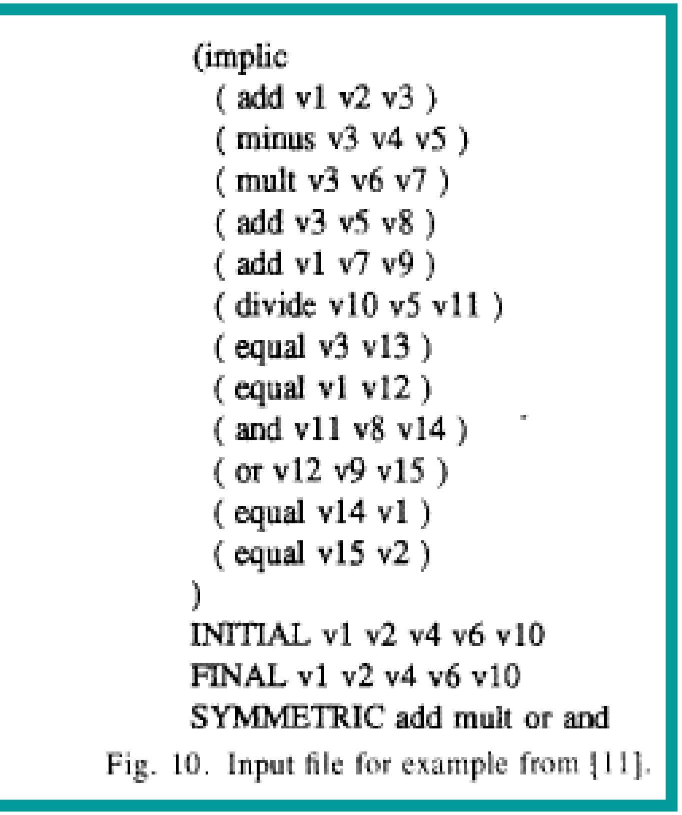
Example and results

- Examples

  No1.

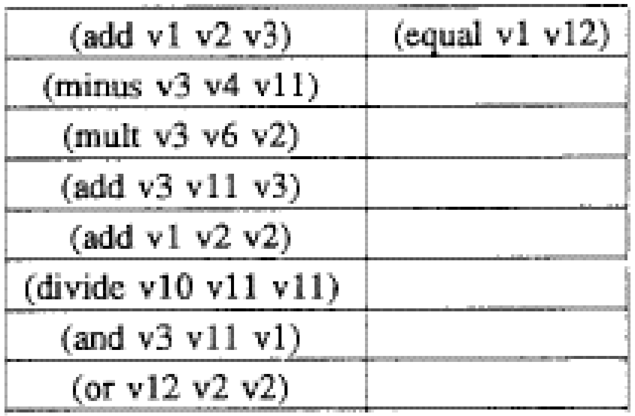
Input description

results after 2D placement



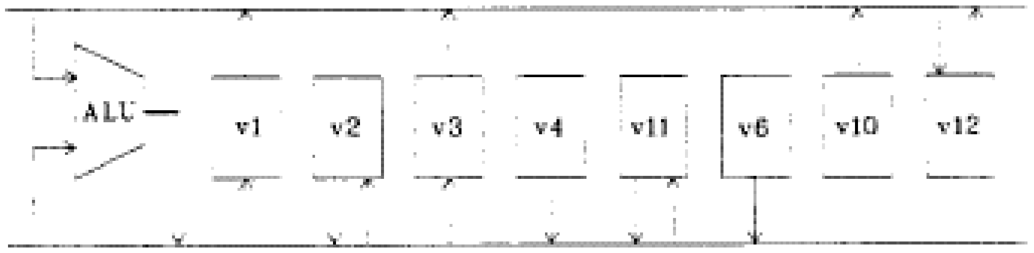
```
(implic
  ( add v1 v2 v3 )
  ( minus v3 v4 v5 )
  ( mult v3 v6 v7 )
  ( add v3 v5 v8 )
  ( add v1 v7 v9 )
  ( divide v10 v5 v11 )
  ( equal v3 v13 )
  ( equal v1 v12 )
  ( and v11 v8 v14 )
  ( or v12 v9 v15 )
  ( equal v14 v1 )
  ( equal v15 v2 )
)
INITIAL v1 v2 v4 v6 v10
FINAL v1 v2 v4 v6 v10
SYMMETRIC add mult or and
```

Fig. 10. Input file for example from [11].



(add v1 v2 v3)	(equal v1 v12)
(minus v3 v4 v11)	
(mult v3 v6 v2)	
(add v3 v11 v3)	
(add v1 v2 v2)	
(divide v10 v11 v11)	
(and v3 v11 v1)	
(or v12 v2 v2)	

(a)



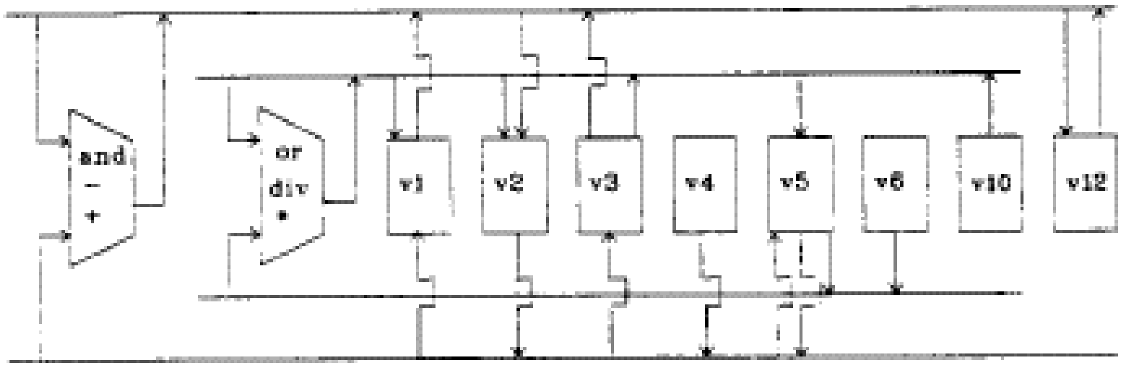
(b)

Fig. 11. (a) Code sequence after two-dimensional placement. (b) Synthesized bus-style data path.

- 4 bus solution

(add v1 v2 v3)	(equal v1 v12)	
(minus v3 v4 v5)		(mult v3 v6 v2)
(add v3 v5 v3)		(divide v10 v5 v5)
(and v1 v2 v2)		(or v3 v5 v1)
(add v12 v2 v2)		

(a)



(b)

```
0 s0 | s0 NOP NOP 0000000000000000000000
1 s0 | s1 NOP NOP 0000000000000000000000
− s1 | s2 AND NOP 0100000001001100000010
− s2 | s3 SUB MUL 0010110000000001100111
− s3 | s4 ADD DIV 0000101100001001101 1
− s4 | s5 AND OR  1101010000010000101 1
− s5 | s6 ADD NOP 0001000001010000001 0
```

(c)

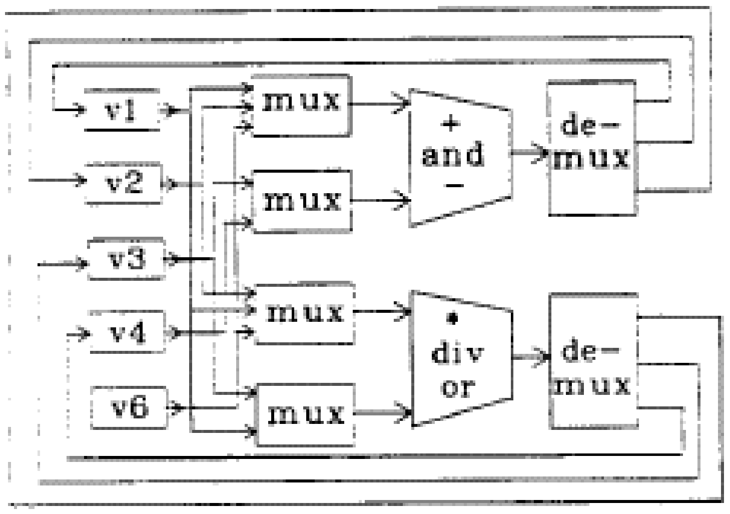
Fig. 12. (a) Code sequence after two-dimensional placement. (b) Synthesized bus-style data path. (c) Finite state machine controller.

No2 which contains conditional clauses in the input description

```
(serial
    (parallel
        (add v2 v3 v1) (divide v2 v3 v4)
    )
    (disjoint
        (add v1 v4 v6) (minus v1 v4 v6)
    )
    (disjoint
        (mult v6 v3 v7)
        (serial (divide v6 v3 v8) (mult v8 v2 v7))
    )
    (parallel
        (and v7 v4 v9) (or v7 v1 v10)
    ) )
```

(add v2 v3 v1)	(divide v2 v3 v4)
[(add v1 v4 v6) (minus v1 v4 v6)]	
	[(mult v6 v3 v6) (divide v3 v6 v3)]
	(mult v2 v3 v6)
(and v6 v4 v2)	(or v6 v1 v3)

(b)



(c)

(a) Input description. (b) Two-dimensional placement. (c) Multiplexer-style data path.

No4. Elliptic filter .Using HAL T=17 cycles,3 Multiplier and 3 adder are needed,.

New algorithm T=17

2 3

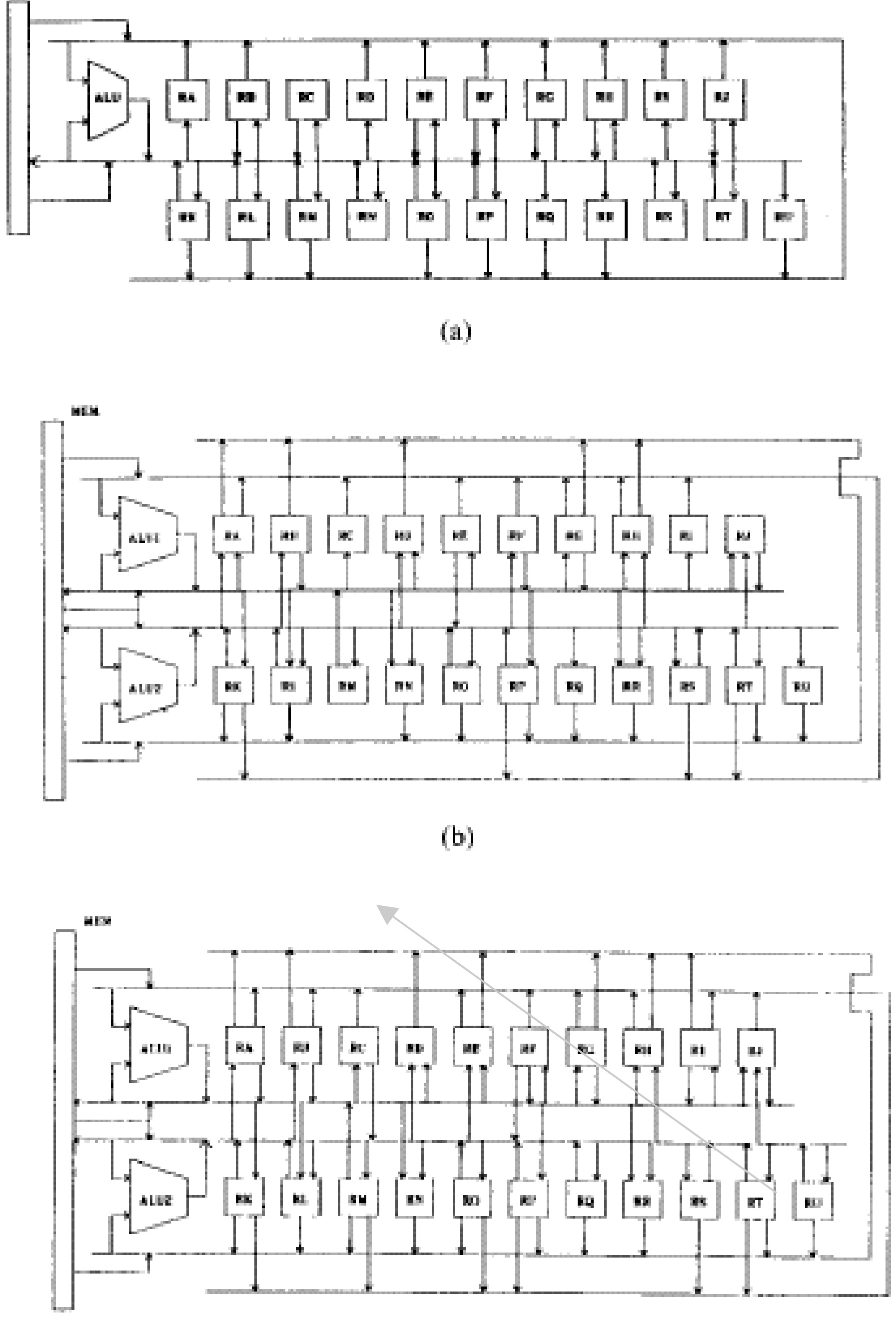
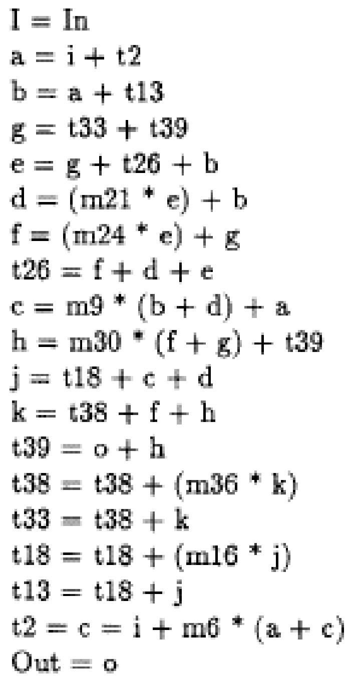


Fig. 14. (a) Data path 1. (b) Data path 2. (c) Data path 3.

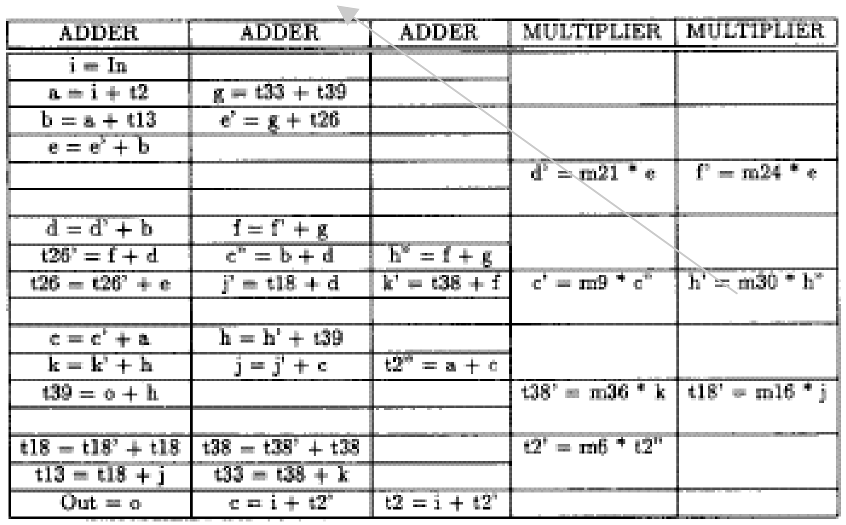
No4. Elliptic filter .Using HAL T=17 cycles,3 Multiplier and 3 adder are needed,.

New algorithm T=17 2 3



```
I = In
a = i + t2
b = a + t13
g = t33 + t39
e = g + t26 + b
d = (m21 * e) + b
f = (m24 * e) + g
t26 = f + d + e
c = m9 * (b + d) + a
h = m30 * (f + g) + t39
j = t18 + c + d
k = t38 + f + h
t39 = o + h
t38 = t38 + (m36 * k)
t33 = t38 + k
t18 = t18 + (m16 * j)
t13 = t18 + j
t2 = c = i + m6 * (a + c)
Out = o
```

(a)



ADDER	ADDER	ADDER	MULTIPLIER	MULTIPLIER
i = In				
a = i + t2	g = t33 + t39			
b = a + t13	e' = g + t26			
e = e' + b				
			d' = m21 * e	f' = m24 * e
d = d' + b	f = f' + g			
t26' = f + d	c'' = b + d	h'' = f + g		
t26 = t26' + e	j' = t18 + d	k' = t38 + f	c' = m9 * c''	h' = m30 * h''
c = c' + a	h = h' + t39			
k = k' + h	j = j' + c	t2'' = a + c		
t39 = o + h			t38' = m36 * k	t18' = m16 * j
t18 = t18' + t18	t38 = t38' + t38		t2' = m6 * t2''	
t13 = t18 + j	t33 = t38 + k			
Out = o	c = i + t2'	t2 = i + t2'		

- ## Result

Comparison with Heuristics algorithm

Run time of algorithm relies on cost functions, however, good cost function may be not be amenable to quick evaluation, it can be kept down to a manageable level by using quickly evaluation cost function at high temperature, and optimality of solution can be obtain by using the good cost function at low temperature

Synthesis Pipelined Data path

- Introduction

  Multiple tasks, SEHWA,

- The pipeline synthesis problem

  Hardware resource can not be shared across pipeline stage, involve partitioning the input data flow description into a number of pipeline stages and finding the placement of micro-operation within each stage so to fulfill the requirement, gave Max delay for each stage. Then using F(C,T)

- Extension for pipeline synthesis

  1. Algorithm begins with a serial pipeline schedule which not violate the MAX delay for each stage

  2.Move, on the condition of the Max delay is not violate

  3.Throughput E proportional of 1/(1+(max(di)*K-1)*P)
  P=resynchronization rate

- Example

- 1. Input description Synthesized pipeline schedule

v1 = x1 + x2 v2 = x3 + x4 v3 = x5 * x6 v4 = x7 * x8
w1 = v1 + x3 w2 = v2 + x2 w3 = v3 + x7 w4 = v4 + x6
y1 = w1 + v3 y2 = w2 + v4 y3 = w3 + v1 y4 = w4 + v2
z1 = y1 + y3 z2 = y1 * y3 z3 = y2 + y4 z4 = y2 * y4
a1 = z1 + x5 a2 = z2 + x6 a3 = z3 + x7 a4 = z4 + x8

$+_s$ (1.0,40) $+_f$ (1.5,25) /* cost delay tradeoff for + */
$*_s$ (2.0,80) $*_f$ (3.0,50) /* cost delay tradeoff for * */

(a)

v1 = x1 $+_s$ x2 v3 = x5 $*_s$ x6
v1 = x3 $+_s$ x4

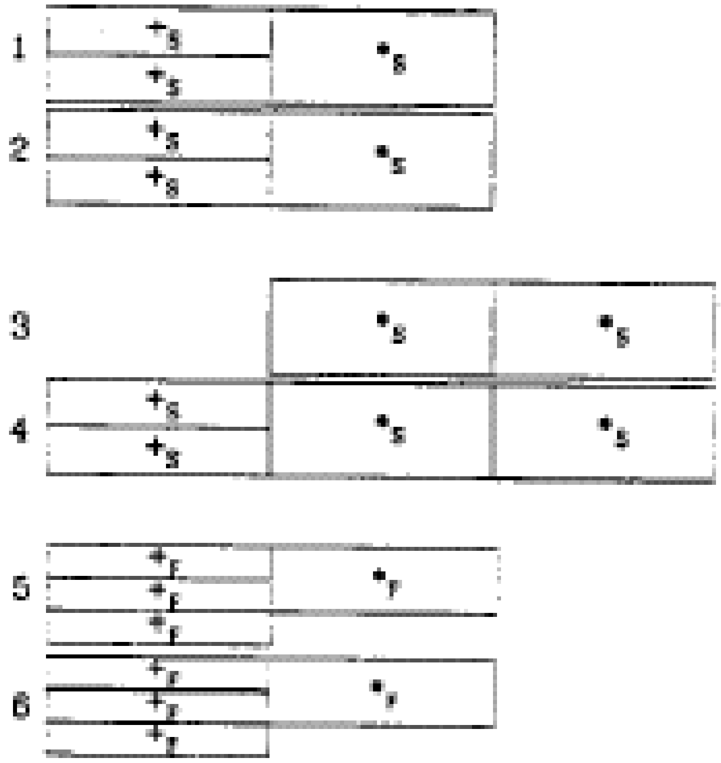
w1 = v1 $+_s$ x3 v4 = x7 $*_s$ x8
w2 = v2 $+_s$ x2

w3 = v3 $*_s$ x7 w4 = v4 $*_s$ x6

y3 = w3 $+_s$ v1 y1 = w1 $*_s$ v3 y2 = w2 $*_s$ v4
y4 = w4 $+_s$ v2

z1 = y1 $+_f$ y3 z2 = y1 $*_f$ y3
z3 = y2 $+_f$ y4
a1 = z1 $+_f$ x5

a2 = z2 $+_f$ x6 z4 = y2 $*_f$ y4
a3 = z3 $+_f$ x7
a4 = z4 $+_f$ x8



(b)

- 2.MOSFET model

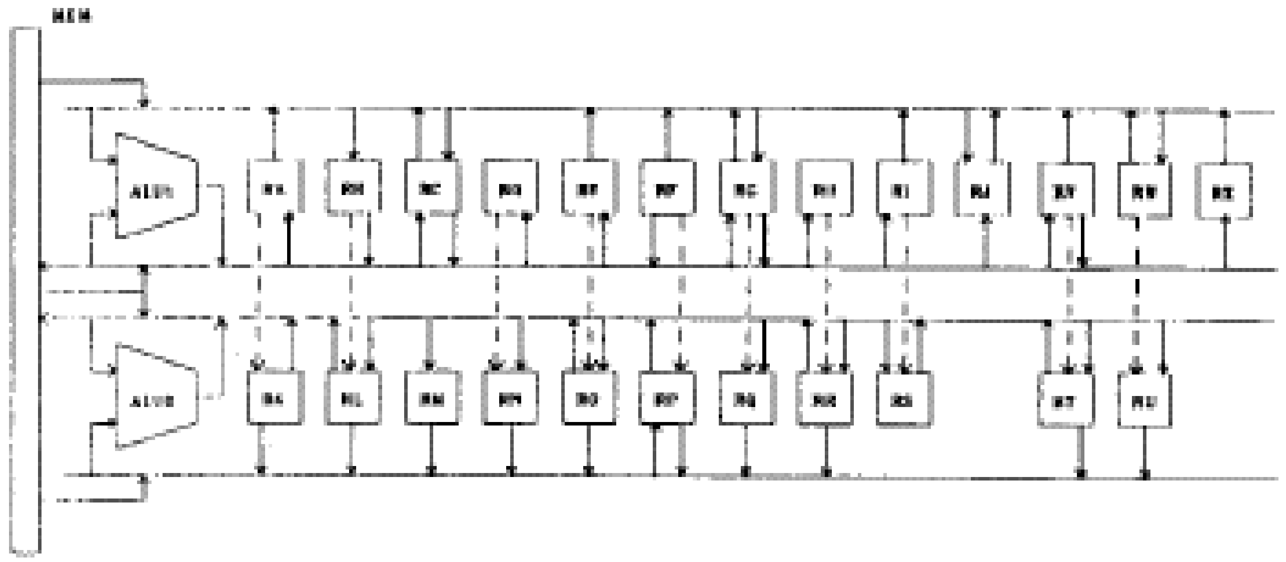


Fig. 17. Data path 4.

SERIAL, PARALLEL, AND PIPELINED DATA PATH STATISTICS

DP	execution time	#reg	#bus	#link	estimated area	CPU time
1	1.0	21	2 + 1*	54	1.0	10.1m
2	0.65	21	4 + 1*	66	1.7	9.2m
3	0.54	21	4 + 1*	77	2.5	11.2m

* memory bus

Limitation and future work

- Interconnect area estimation
- Complex constraints would require a complicated analysis, decrease efficiency.

Conclusion

- The novel method for synthesizing data path from behavioal description The entire allocation process in data path can be formulated as two-D placement problem of microinstruction in space and time.it allows simultanous cost-costraints ALU, bus, registers while trade off execution speed. It can extended to pipelined data path synthesis.
- It works.