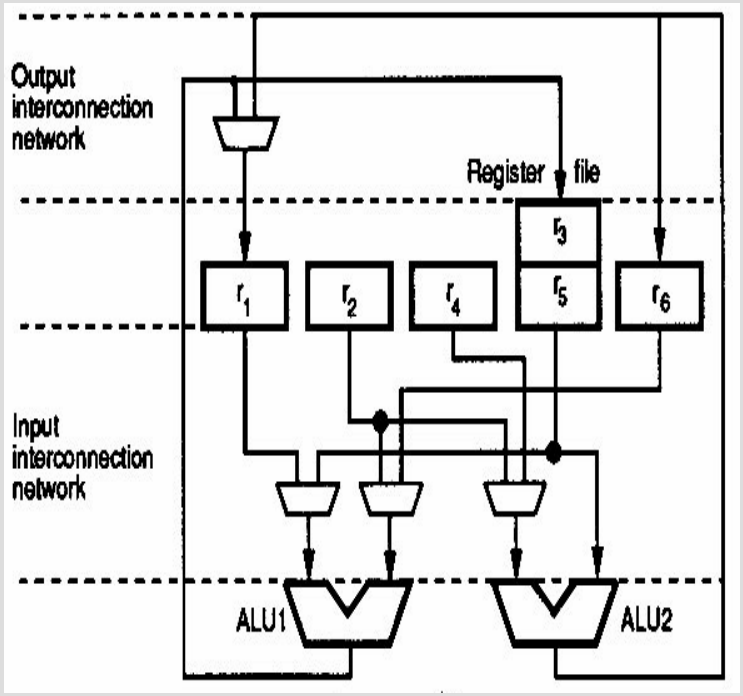
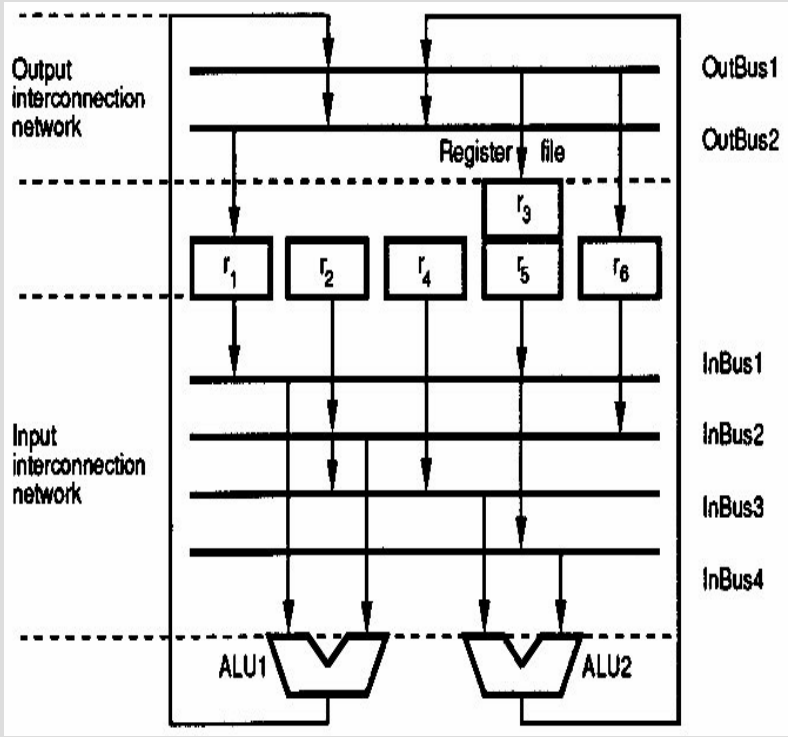
Datapath interconnections

- Multiplexer-oriented datapath

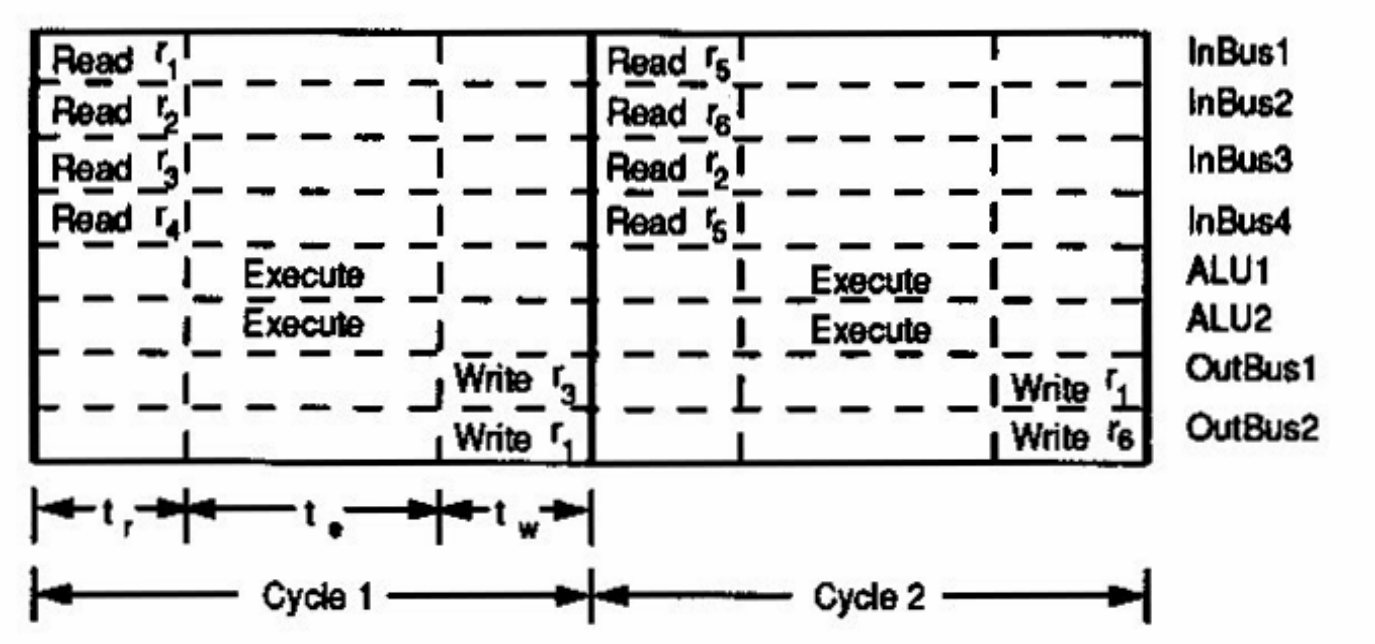


- Bus-oriented datapath



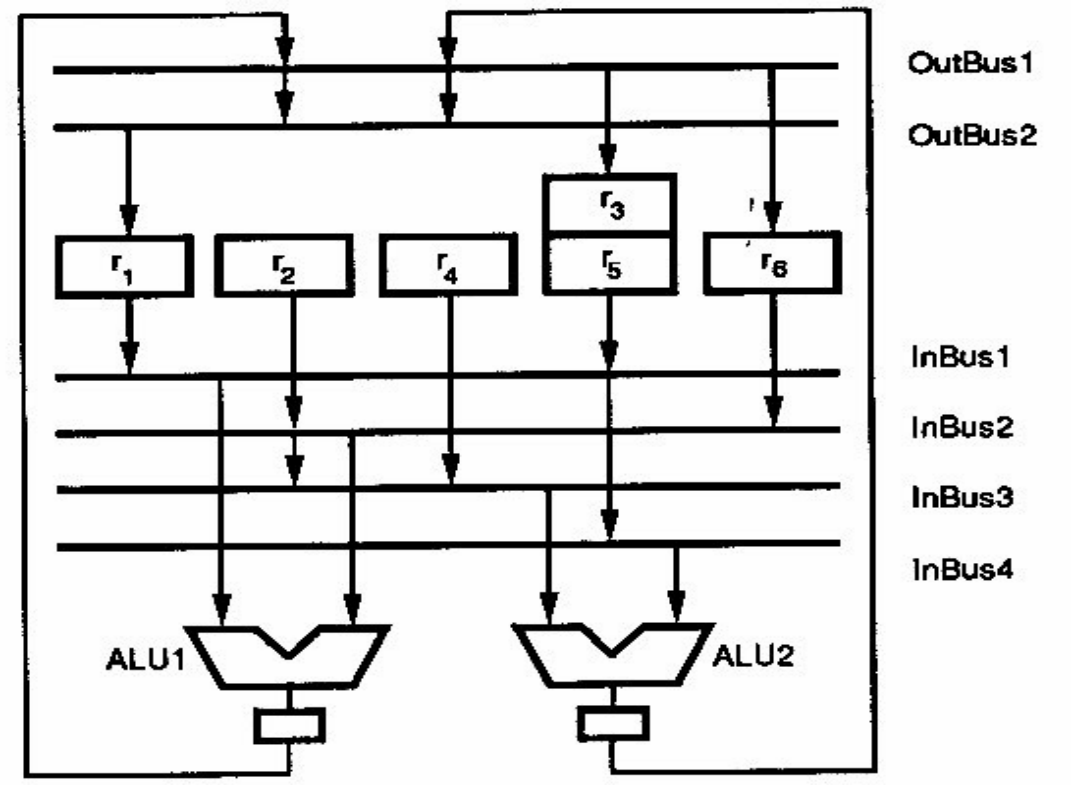
Sequential Execution

- Example of three micro-operations in the same clock period



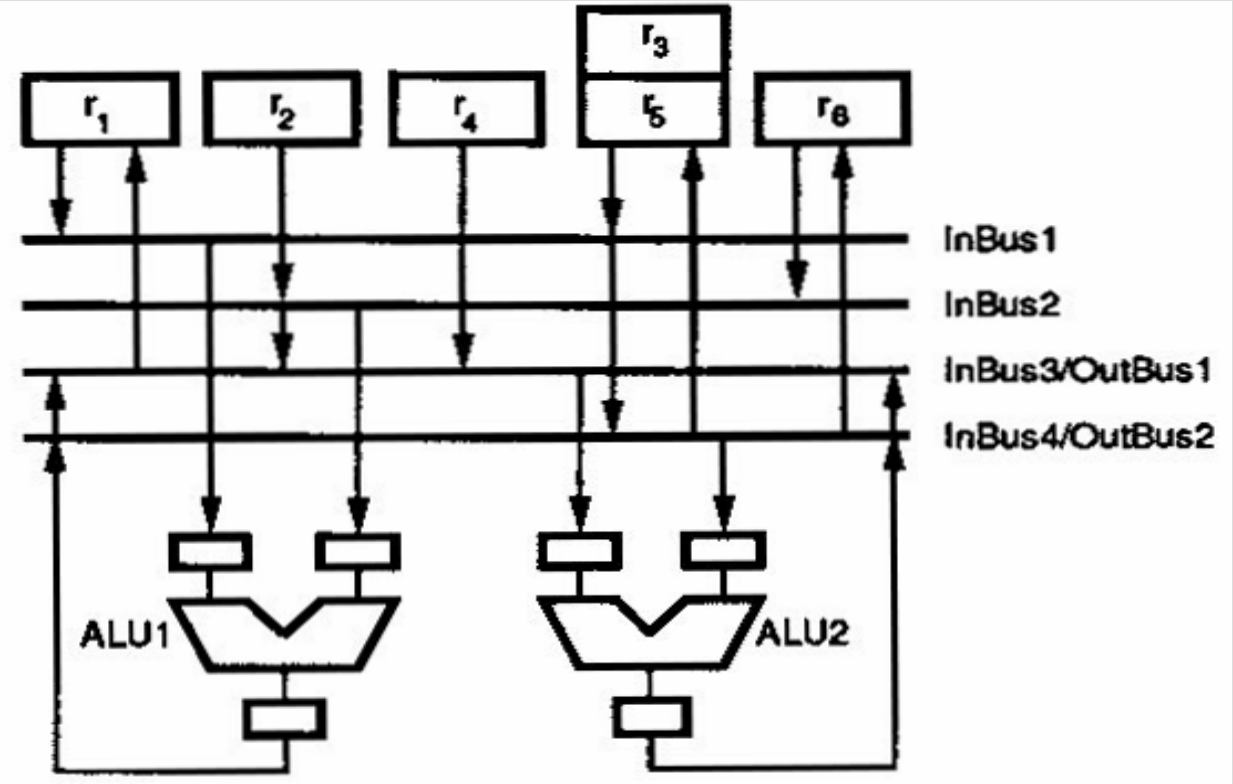
Insertion of Latch (out)

- Insertion of latches at the output ports of the functional units



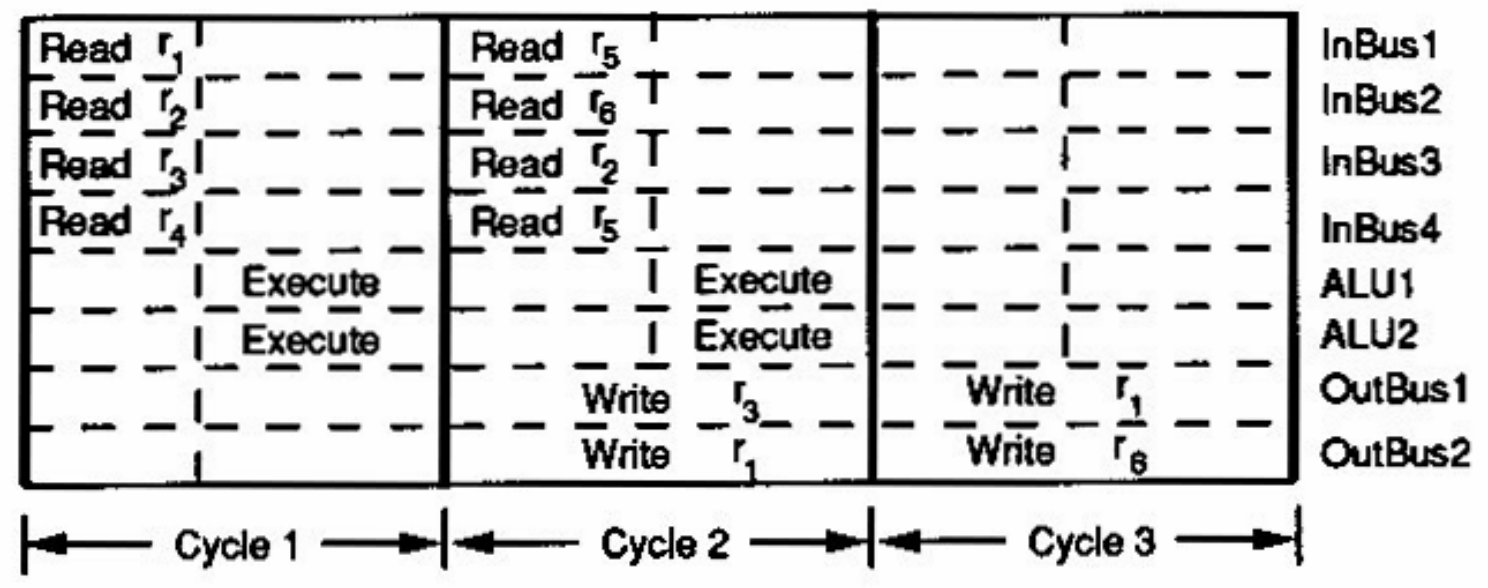
Insertion of Latch (in/out)

- Insertion of latches at both the input and output ports of the functional units



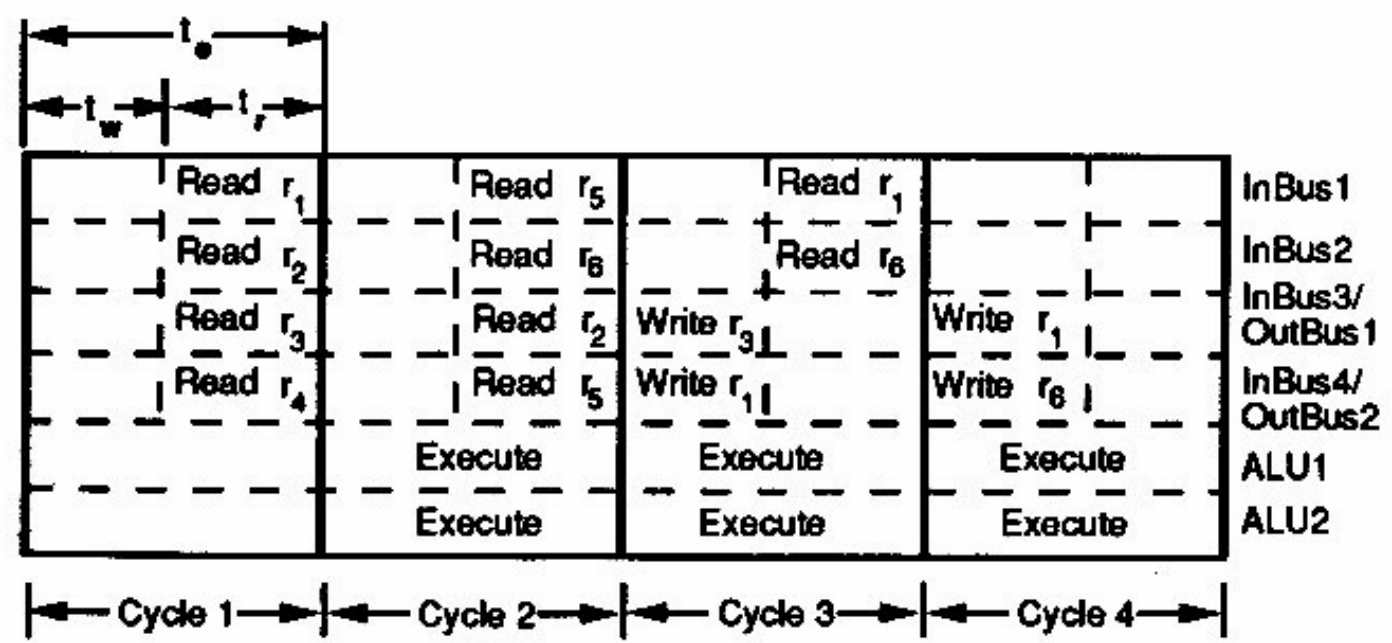
Overlapping Data Transfer(in)

- Overlapping read and write data transfers



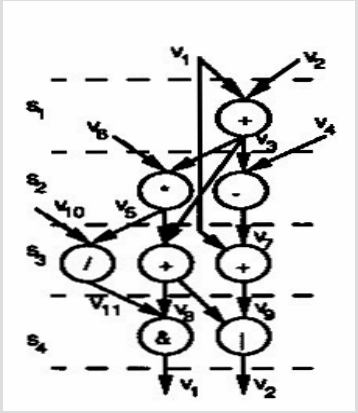
Overlapping of Data Transfer (in/out)

- Overlapping data transfer with functional-unit execution

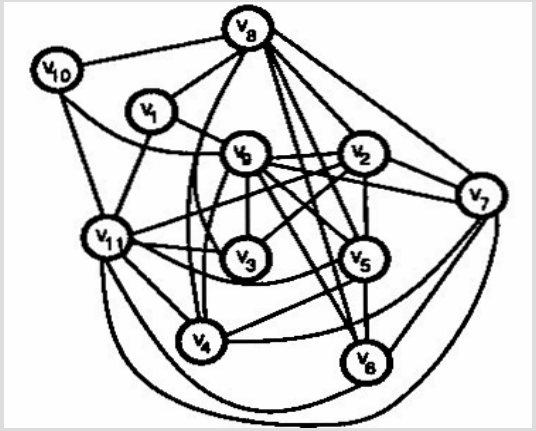


Register Allocation Using Clique Partitioning

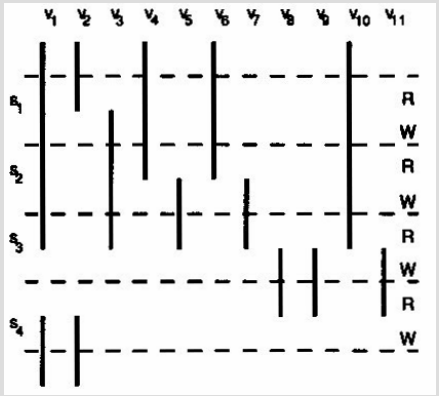
- Scheduled DFG



- Graph model



- Lifetime intervals of variable



- Clique-partitioning solution

Cliques:

$r_1 = \{v_1, v_8\}$

$r_2 = \{v_2, v_3, v_9\}$

$r_3 = \{v_4, v_5, v_{11}\}$

$r_4 = \{v_6, v_7\}$

$r_5 = \{v_{10}\}$

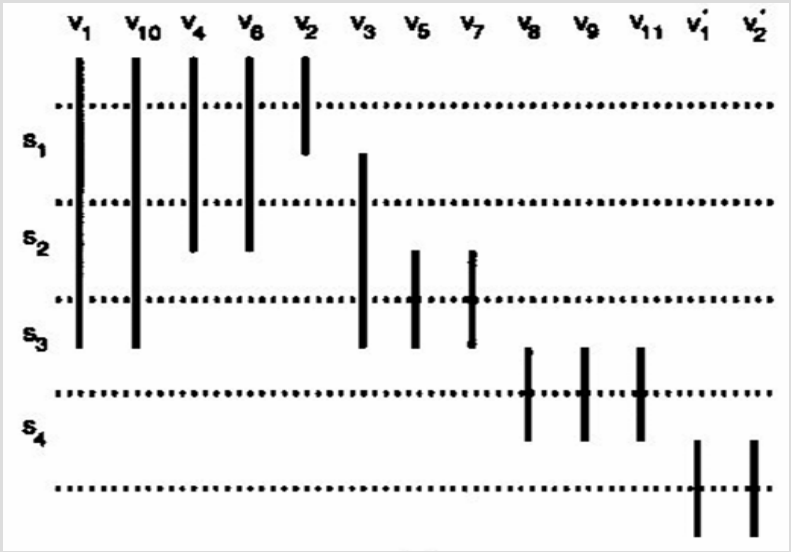
Left-Edge Algorithm

- Register allocation using Left-Edge Algorithm

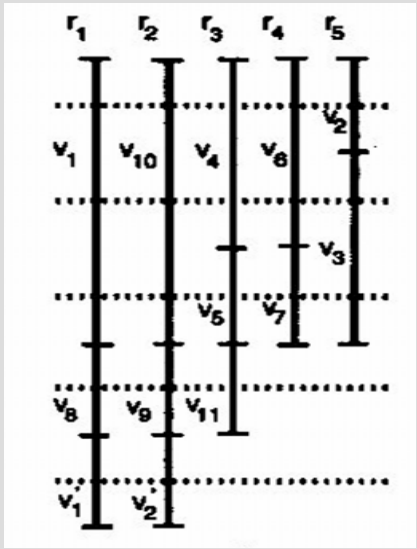
```
for all v ∈ L do  MAP[v] = 0;  endfor
SORT(L);

reg_index = 0;
while L ≠ φ do
    reg_index = reg_index + 1;
    curr_var = FIRST(L);
    last = 0;

    while curr_var ≠ null do
        if Start(curr_var) ≥ last then
            MAP[curr_var] = r_reg_index;
            last = End(curr_var);
            temp_var = curr_var;
            curr_var = NEXT(L, curr_var);
            DELETE(L, temp_var);
        else
            curr_var = NEXT(L, curr_var);
        endif
    endwhile

endwhile
```

Register Allocation: Left-Edge Algorithm

- Sorted variable lifetime intervals



- Five-register allocation result



Register Allocation

Allocation : bind registers and functional modules to variables and operations in the CDFG and specify the interconnection among modules and registers in terms of MUX or BUS.

Reduce capacitance during allocation by minimizing the number of functional modules, registers, and multiplexers.

Register allocation and variable assignment can have a profound impact on spurious switching activity in a circuit

Judicious variable assignment is a key factor in the elimination of spurious operations

Effect of Register Sharing

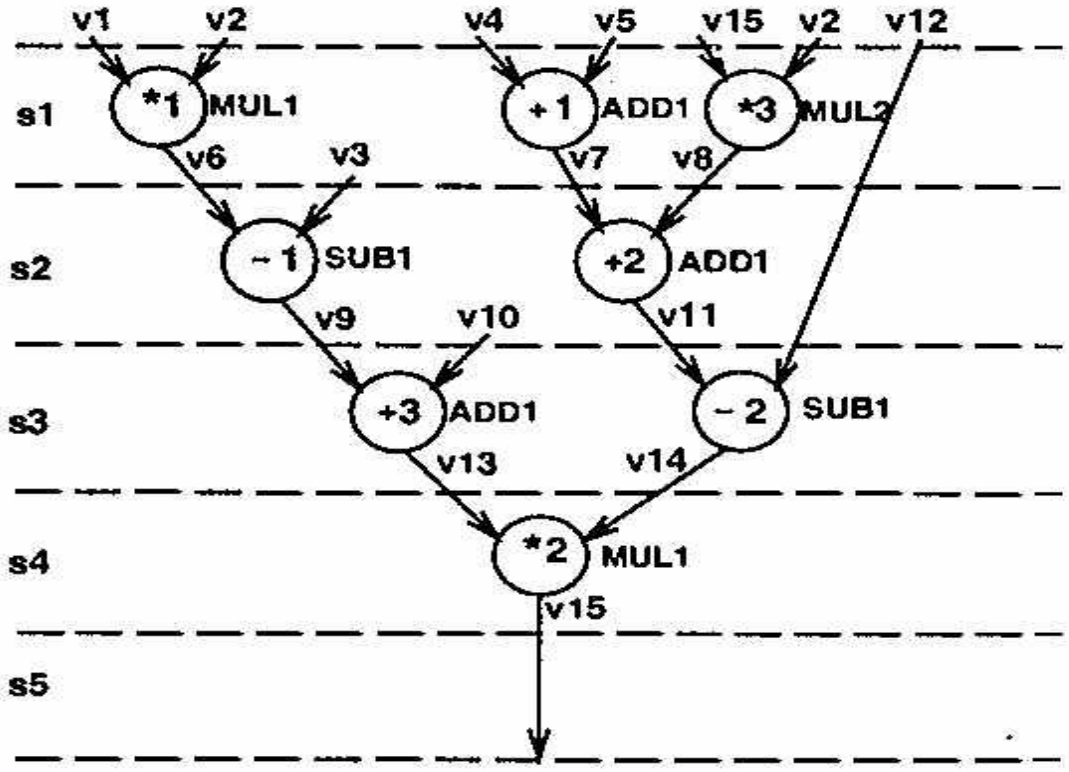


Fig. 2. A scheduled DFG to illustrate execution of spurious operations.

Two Variable Assignments

TABLE I

TWO VARIABLE ASSIGNMENTS FOR THE SCHEDULED DFG SHOWN IN FIG. 2

Register	*Assignment 1*	*Assignment 2*
R1	*v1, v7, v11, v13*	*v1, v13*
R2	*v2, v8, v10, v14*	*v2*
R3	*v3, v5, v9*	*v4, v8, v10*
R4	*v4, v6*	*v5, v7, v9*
R5	*v12*	*v12*
R6	*v15*	*v3*
R7	-	*v6, v11*
R8	-	*v14*
R9	-	*v15*

Spurious Computing

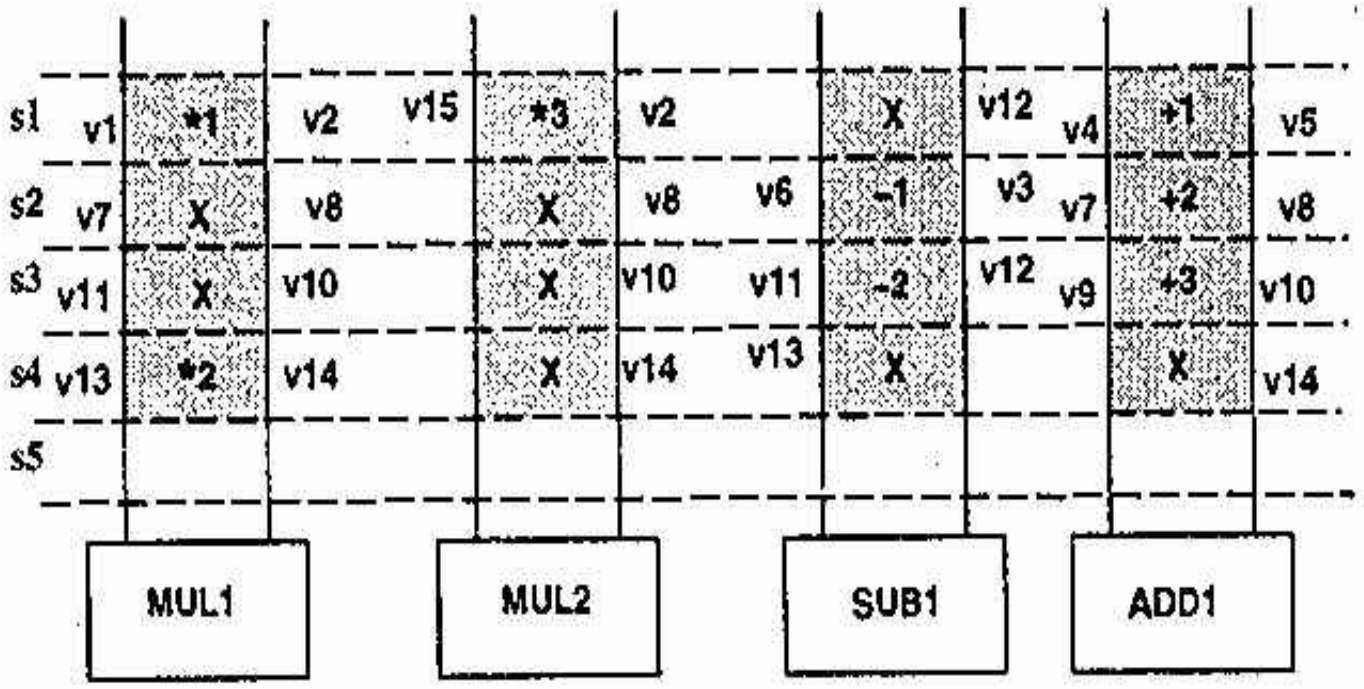


Fig. 3. Switching activity in the functional units of *Design 1*.

Minimizing Spurious Computing

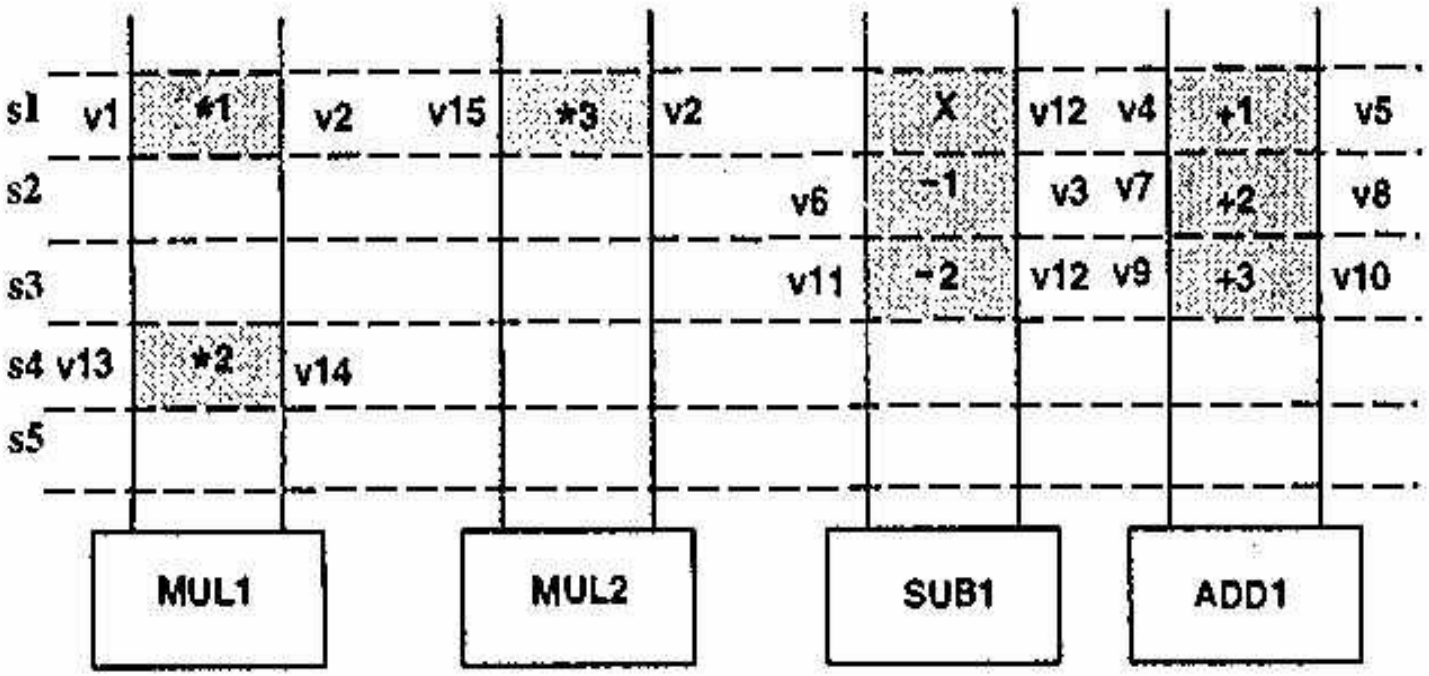


Fig. 4. Switching activity in the functional units of *Design 2*.

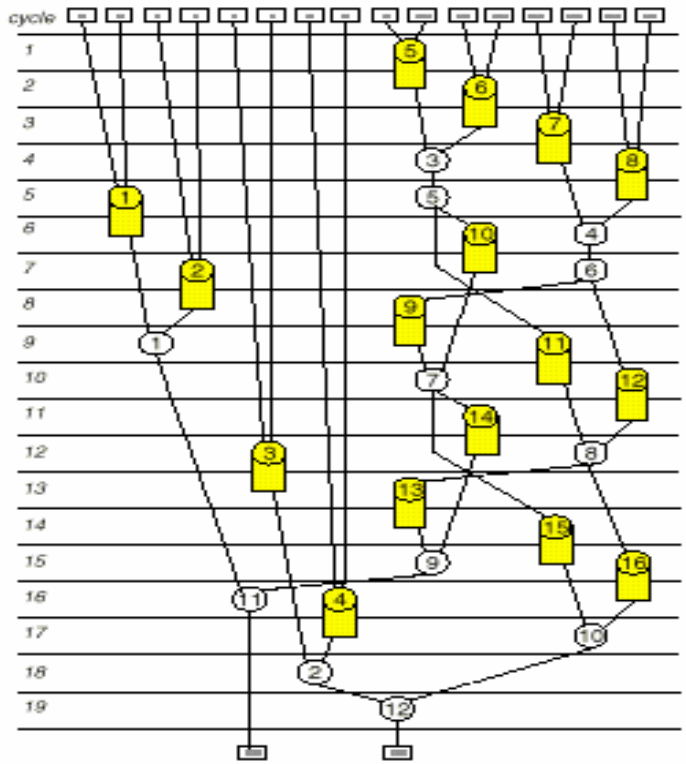
Effect of Register Sharing on FU

- For a small increase in the number of registers, it is possible to significantly reduce or eliminate spurious operations
- For Design 1, synthesized from Assignment 1, the power consumption was 30.71mW
- For Design 2, synthesized from Assignment 2, the power consumption was 18.96mW @1.2-❍m standard-cell library

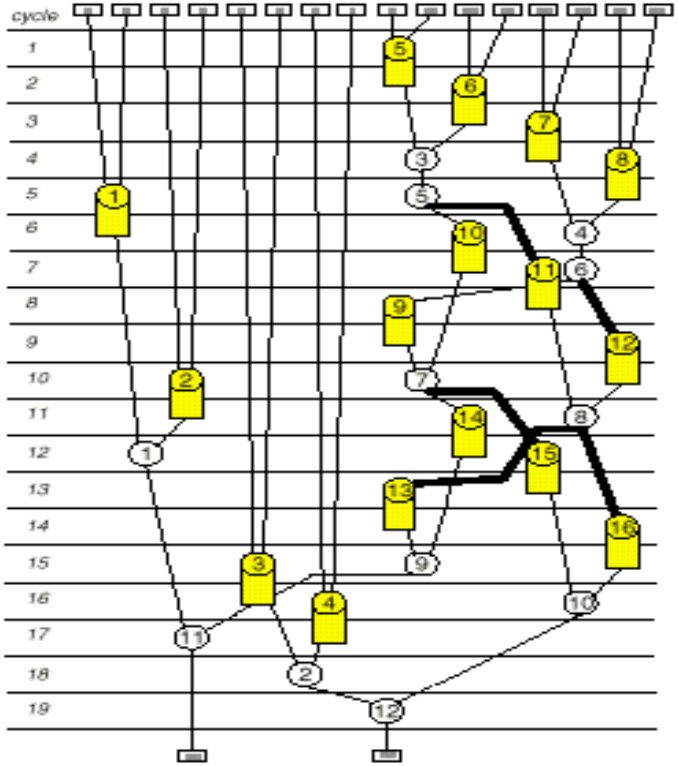
Operand sharing

- To schedule and bind operations to functional units in such a way that the activity of the input operands is reduced.
- Operations sharing the same operand are bound to the same functional unit and scheduled in such a way that the function unit can reuse that operand.
- we will call operand reutilization (OPR) the fact that an operand is reused by two operations consecutively executed in the same functional unit.

Two scheduling and bindings



(b)



(c)

Power Saving by OPR

Thus, the estimated power consumption of one iteration in schedule (b) is:

$$P_b\ (\beta) = 12\ P_{add2} + 16\ P_{mul2} = P_{mul2}\ (16 + 12\ \beta)\ ,$$

and the estimated power consumption of one iteration in schedule (c) is:

$$P_c\ (\alpha_{mul}, \beta) = 12\ P_{add2} + 12\ P_{mul2} + 4\ P_{mul1} = P_{mul2}\ (12 + 4\ \alpha_{mul} + 12\ \beta)\ .$$

The estimated power-consumption reduction is:

$$R(\alpha_{mul}, \beta) = \frac{1 - \alpha_{mul}}{4 + 3\ \beta}\ ;$$

achieving an 8.5% reduction.

Results obtained by applying the operand-sharing technique.

Benchmark	+/-	⊗, ⊕	Power red.
5th-order Elliptic filter [DDN85]	26/8	1 ⊗ (2 cycles), 2 ⊕ (1 cycle)	12%
4th-order Daubechies filter [PTVT92]	12/12	1 ⊗ (2 cycles), 1 ⊕ (1 cycle)	21%
SHARF [VJL87]	11/12	1 ⊗ p. (2 cycles), 2 ⊕ (1 cycle)	10%
1-D 8-input Lee DCT [RY96]	29/13	2 ⊗ (2 cycles), 2 ⊕ (1 cycle)	6%
1-D 8-input Chen DCT [RY96]	26/16	2 ⊗ (2 cycles), 2 ⊕ (1 cycle)	19%
4 × 4 matrix multiplier	4/8	2 ⊗ (2 cycles), 1 ⊕ (1 cycle)	26%

Loop unrolling

- The technique of loop unrolling replicates the body of a loop some number of times (unrolling factor u) and then iterates by step u instead of step 1. This transformation reduces the loop overhead, increases the instruction parallelism and improves register, data cache or TLB locality.

```
for i = 2 to N – 1
    A(i) – A(i) + A(i – 1) A(i + 1)
```

(a)

```
for i = 2 to N – 2 step 2
    A(i) – A(i) + A(i – 1) A(i + 1)
    A(i + 1) = A(i + 1) + A(i) A(i + 2)
```

(b)

Loop Unrolling Effects

- **Loop overhead is cut** in half because two iterations are performed in each iteration.
- If array elements are assigned to registers, **register locality is improved** because A(i) and A(i +1) are used twice in the loop body.
- **Instruction parallelism is increased** because the second assignment can be performed while the results of the first are being stored and the loop variables are being updated.

Loop Unrolling (IIR filter example)

loop unrolling : localize the data to reduce the activity of the inputs of the functional units or two output samples are computed in parallel based on two input samples.

$$Y_{n-1} = X_{n-1} + A \times Y_{n-2}$$
$$Y_n = X_n + A \times Y_{n-1} = X_n + A \times (X_{n-1} + A \times Y_{n-2})$$

Neither the capacitance switched nor the voltage is altered. However, loop unrolling enables several other transformations (distributivity, constant propagation, and pipelining). After distributivity and constant propagation,

$$Y_{n-1} = X_{n-1} + A \times Y_{n-2}$$
$$Y_n = X_n + A \times Y_{n-1} + A^2 \times Y_{n-2}$$

The transformation yields critical path of 3, thus voltage can be dropped.

Loop Unrolling for Low Power

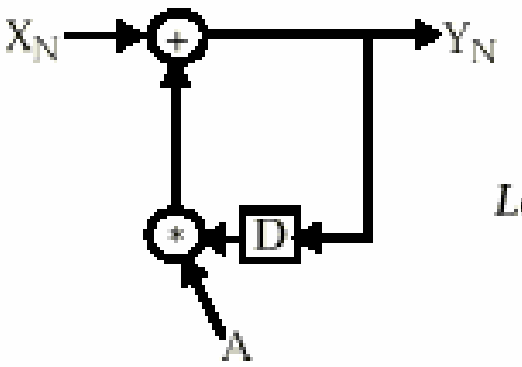


Figure 35a

$C_{eff} = 1$
Voltage = 5
Throughput = 1
Power = 25

Loop Unrolling

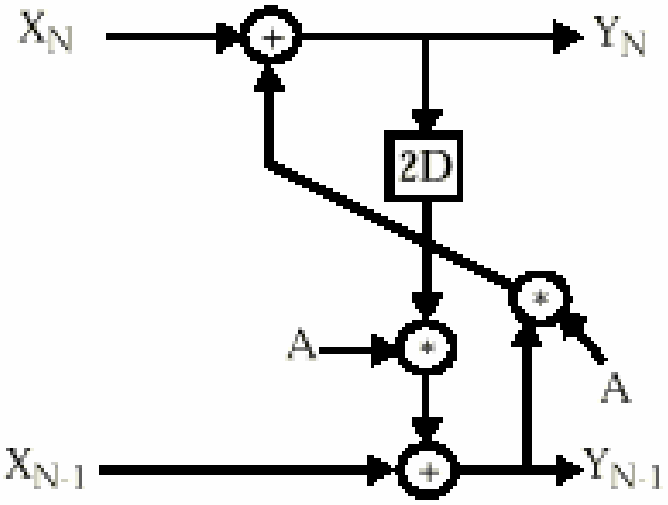


Figure 35b

$C_{eff} = 1$
Voltage = 5
Throughput = 1
Power = 25

Loop Unrolling for Low Power

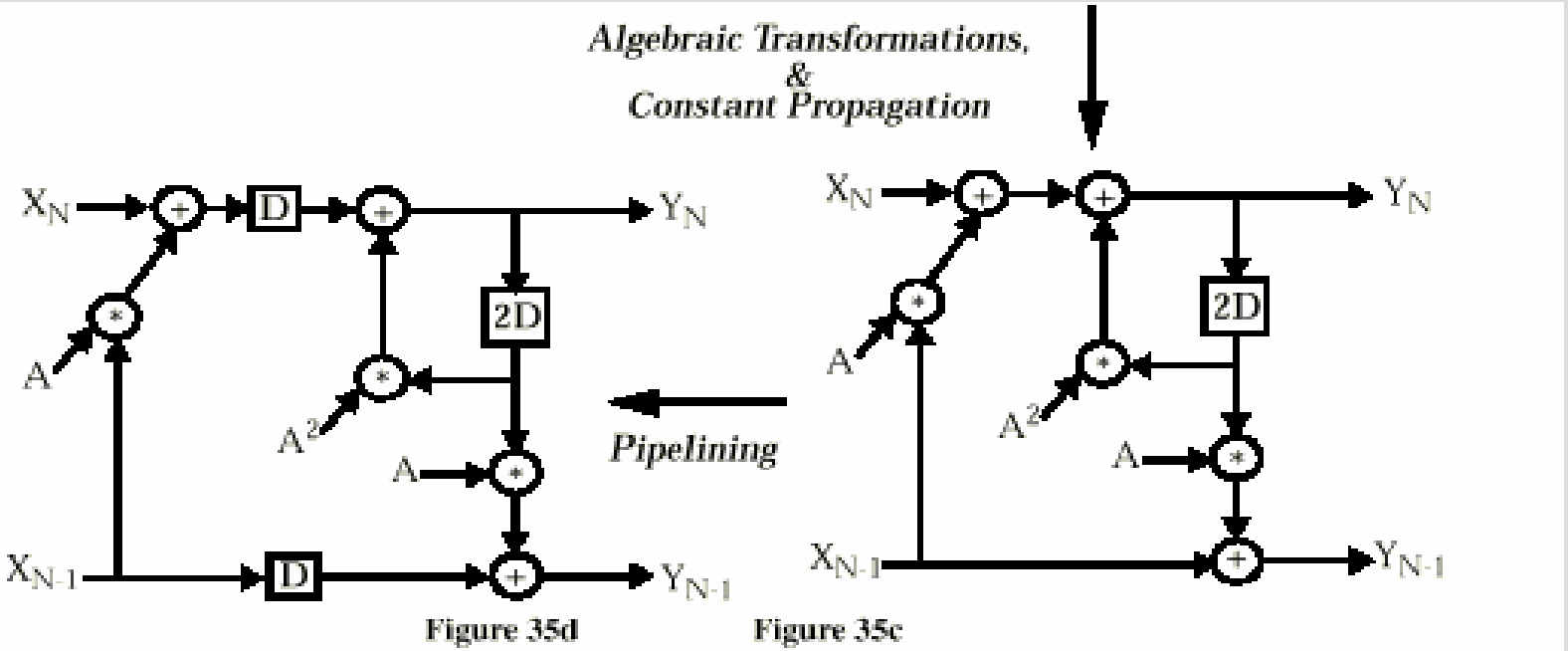


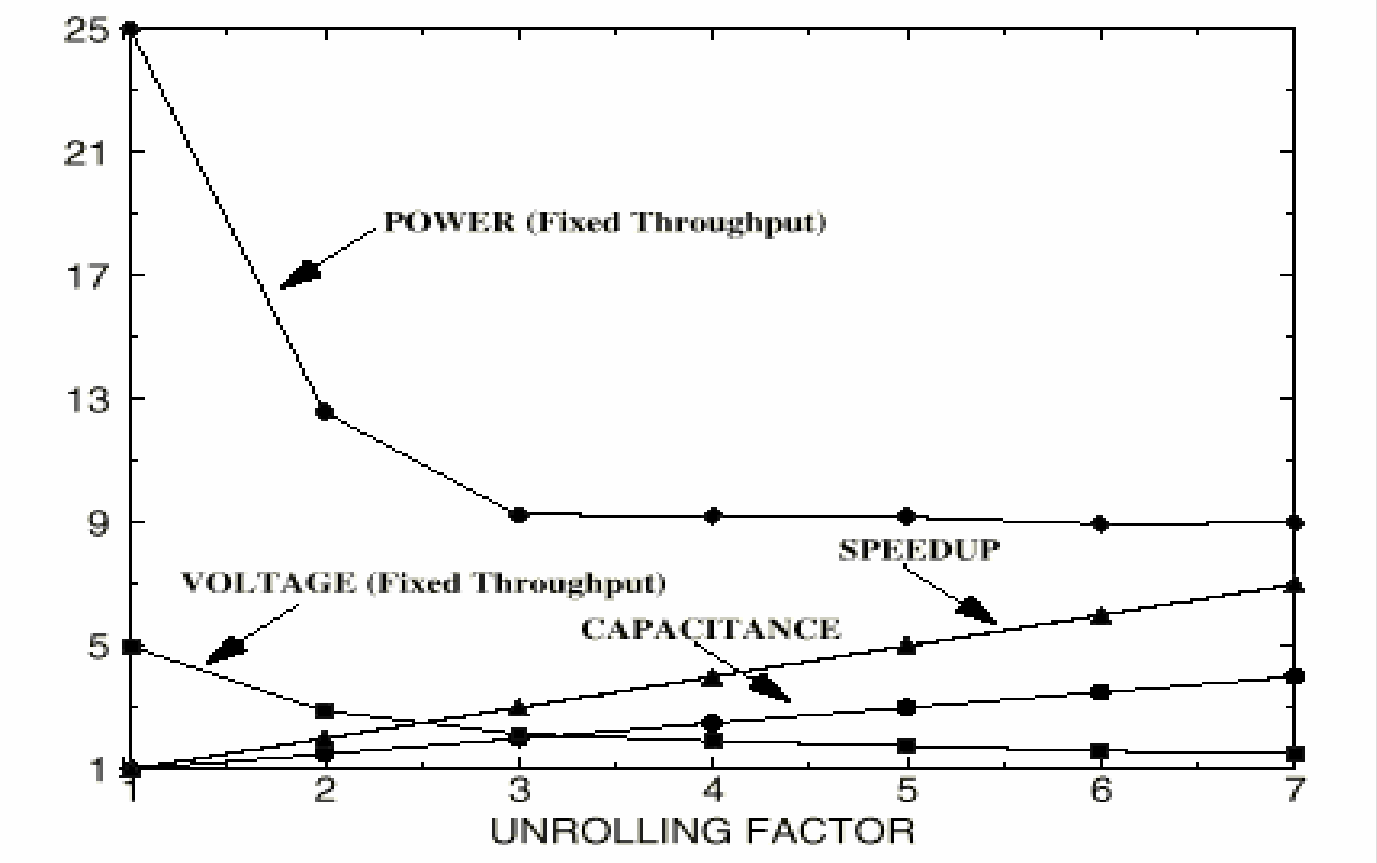
Figure 35d

C_{eff} = 1.5
Voltage = 2.9
Throughput = 1
Power = 12.5 (x2 reduction)

Figure 35c

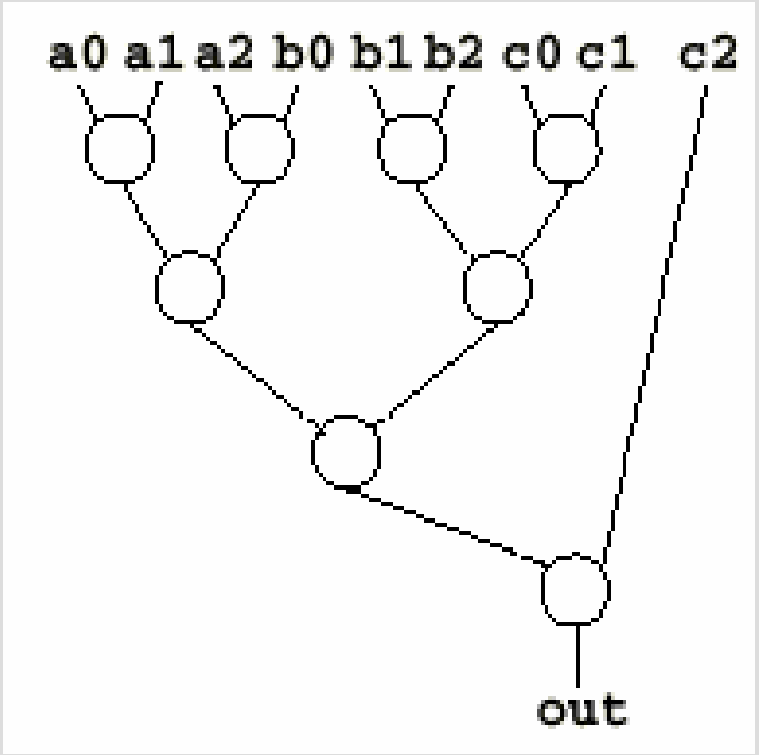
C_{eff} = 1.5
Voltage = 3.7
Throughput = 1
Power = 20 (20% reduction)

Loop Unrolling for Low Power

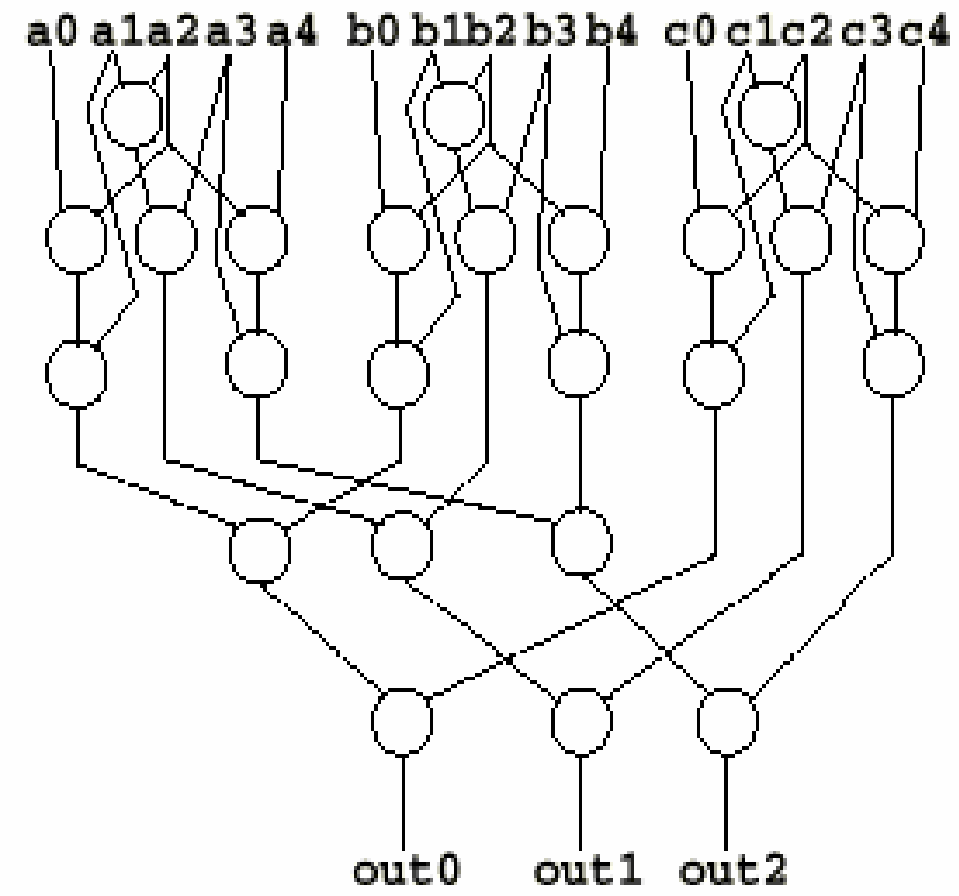


Loop Unrolling for OPR

```
for i = 0 to M
  for j = 0 to N
    out =
      (A[i-1][j-1]+      /* a0 */
       A[i-1][j]+        /* a1 */
       A[i-1][j+1]+      /* a2 */
       A[i][j-1]+        /* b0 */
       A[i][j]+          /* b1 */
       A[i][j+1]+        /* b2 */
       A[i+1][j-1]+      /* c0 */
       A[i+1][j]+        /* c1 */
       A[i+1][j+1])/9    /* c2 */
```



DFG after Loop Unrolling



The estimated power-consumption reduction is now:

$$R(\alpha_{add}) = \frac{3}{8}(1 - \alpha_{add})$$

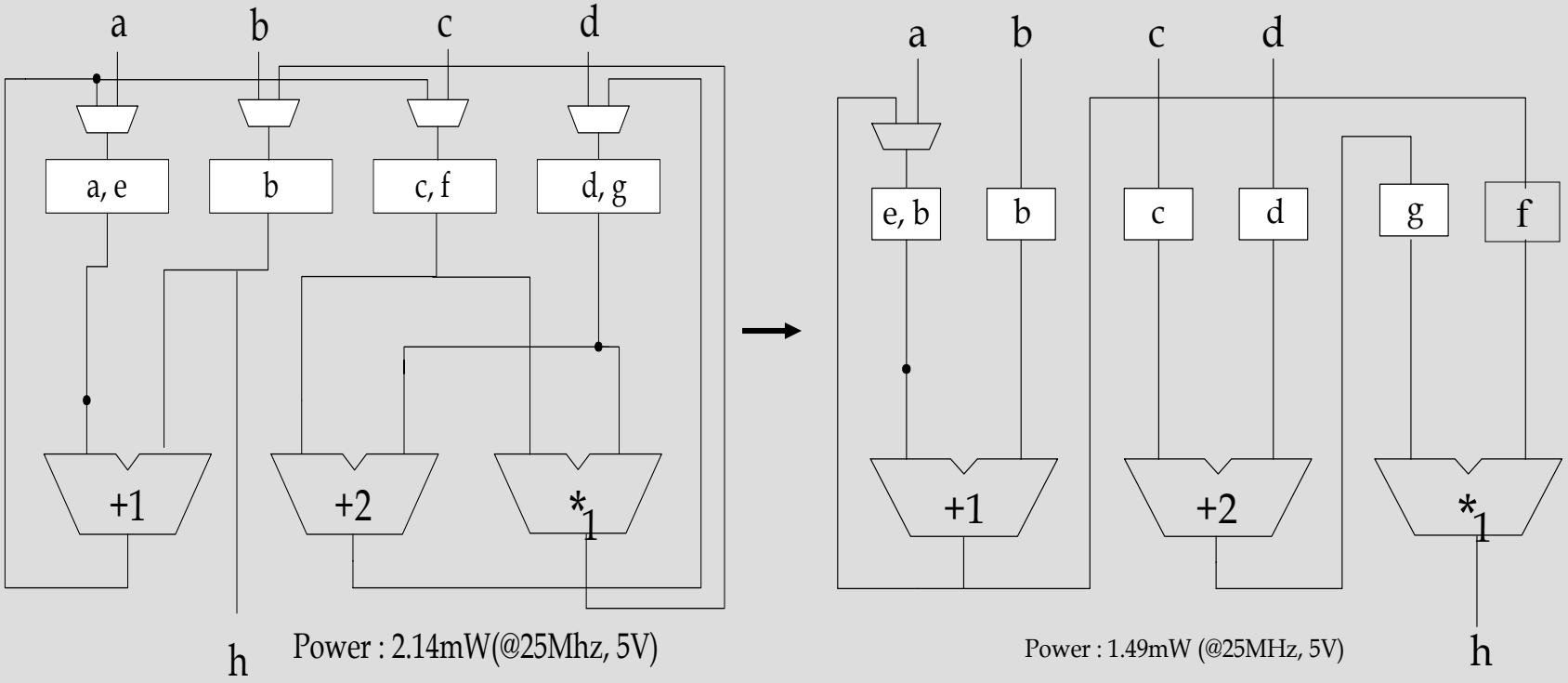
obtaining a reduction of 9.4%.

Operand Retaining

- An idle functional unit may have input operand changes because of the variation of the selection signals of multiplexers.
- The operand-retaining technique attempts to minimize the useless power consumption of the idle functional units.

	Benchmark	+/*	⊖, ⊗	Idle time	Power reduc.
(1)	*AR filter [Kun84]*	12/16	1 p. ⊗ (2 c), 1 ⊖ (1 c)	27%	12%
(2)	*4th-order Daubechies filter [PTVF92]*	12/16	1 ⊗ (2 c), 1 ⊕ (1 c)	37%	9%
(3)	*1-D 8-input Lee DCT [RY96]*	28/13	4 ⊗ (2 c), 3 ⊕ (1 c)	40%	38%
(4)	*4 × 4 matrix multiplication*	4/8	2 ⊗ (2 c), 1 ⊕ (1 c)	33%	15%
(5)	*unrolled low-pass image filter (3.13)*	24/0	4 ⊕ (1 c)	25%	20%
(6)	*LMS adaptive filter [TJL87]*	8/9	2 ⊗ (2 c), 1 ⊕ (1 c)	45%	34%
(7)	*pixel interpolation [BS95]*	5/0	3 ⊕ (1 c)	44%	34%
(8)	*5th-order Elliptic filter [DDN85]*	26/8	1 ⊗ (2 c), 1 ⊕ (1 c)	33%	21%

Spurious 연산을 최소화



Minimize the useless power consumption of idle units

- (a) with a proper register binding that minimizes the activity of the functional units
- (b) by wisely defining the control signals of the multiplexors during the idle cycles in such a way that the changes at the inputs of the functional units are minimized (this may result in defining some of the don't care values of the control signals) [RJ94]
- (c) latching the operands of those units that will be often idle.

(C) latching the operands

- It consists of the insertion of latches at the inputs of the functional units to store the operands only when the unit requires them. Thus, in those cycles in which the unit is idle no consumption in produced.
- The control unit has to be redesigned accordingly, in such a way that input latches become transparent during those cycles in which the corresponding functional unit must execute an operation.
- Similar to putting the functional units to sleep when they are not needed through gated-clocking strategies [CSB94,BSdM94, AMD94].

LMS filter Example

- The power consumption generated by the idle units (useless consumption) is:

$$P_{usclcss}\left(\alpha_{add};\alpha_{mul},\beta\right)=8\,P_{add1}+7\,P_{mul1}=8\,\alpha_{add}\,\beta+7\,\alpha_{mul}$$

- the power consumption because of the useful calculations (useful consumption) is:

$$P_{useful}\left(\alpha_{add},\alpha_{mul},\beta\right)=8\,P_{add2}+9\,P_{mul2}=8\,\beta+9$$

The estimated reduction in power consumption is:

$$R\left(\alpha_{add},\alpha_{mul},\beta\right)=\frac{P_{usclcss}}{P_{useless}+P_{useful}}$$

reduction of 34%.

레지스터 할당을 위한 가중치와 캐패시티

입력(input) : V, 레지스터 공유를 위한 네트워크
출력(output) : 분리된 V개의 경로

$C_{ij}=1, B_{ij}=L$

$C_{ij}=1, B_{ij}=W$

$W = -(M - Wa*N)$: 에지 가중치

Wa : 두 노드간의 스위칭 확률

N : 스위칭 확률을 정수화하는 값

M : $Wa * N$의 최대값과 크거나 같은 정수

L : W의 최대값과 크거나 같은 정수

V : 레지스터의 갯수)

최소 비용 흐름 알고리즘(Minimum Cost Flows)의 목적함수와 제한조건

Minimize $Z = \sum B_{ij} X_{ij}$

$$\sum_i X_{ij} - \sum_k X_{jk} = -V \quad \text{if } j = s$$

$$\sum_i X_{ij} - \sum_k X_{jk} = 0 \quad \text{if } j \neq s$$

$$\sum_i X_{ij} - \sum_k X_{jk} = V \quad \text{if } j = t$$

$$0 \leq X_{ij} \leq C_{ij}$$

s : 출발지 (source)　　t : 목적지 (target)

i, j, k : 노드 (node)　　C_{ij} : 제한양 (capacity)

B_{ij} : 비용 (cost)　　V : 흐름의 양 (flow)

최소 비용 흐름 알고리즘

단계 1 : flow = 0;

단계 2 : 네트워크 상에서 존재하는 흐름에 의해 결정되는 변형된 비용 B_{ij}^{*}를 다음과 같이 정의한다.

$$B_{ij}^{*} = B_{ij} \quad \text{if} \quad 0 < X_{ij} < C_{ij}$$

$$B_{ij}^{*} = \infty \quad \text{if} \quad 0 \; X_{ij} = C_{ij} \; (\text{flow saturation})$$

$$B_{ij}^{*} = -B_{ji} \quad \text{if} \quad X_{ji} > 0$$

단계 3 : 단계 2에서 변형된 비용으로 S에서 T까지의 최단 경로 알고리즘을 사용하여 ε개의 flow를 그경로를 통하여 보낸다.

ε = min (ε1, ε2)

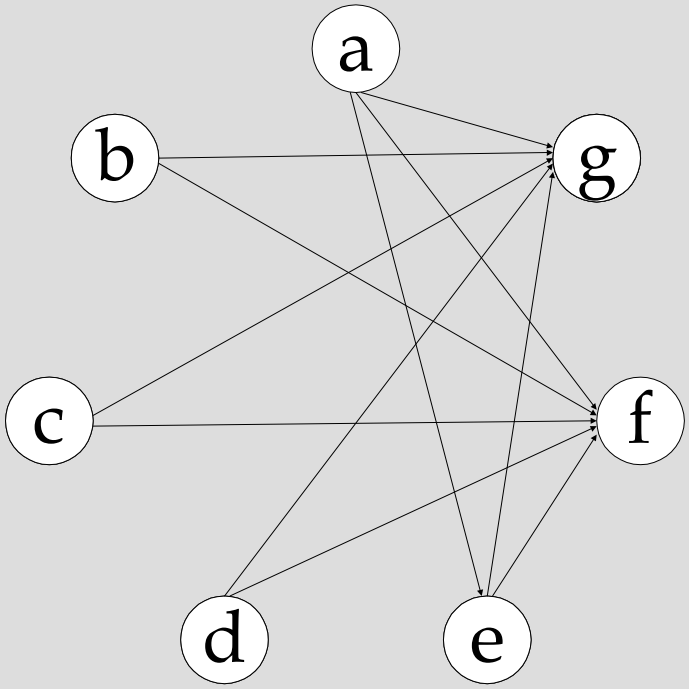
ε1 = 모든 정방향 에지의 min{Cij - Xij}

ε2 = 모든 역방향 에지의 min{Xij}

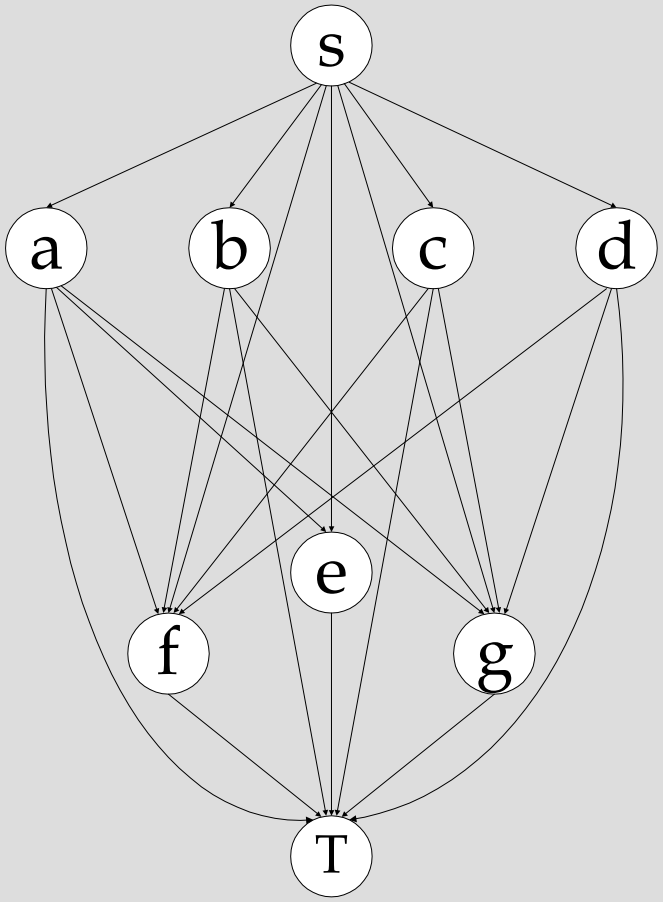
단계 4: 현재흐름의 양을 ε만큼 증가시키고 단계 2로 돌아간다.

단계 5: 현재흐름의 양이 V일때까지 위 단계를 반복한다.

레지스터 호환 그래프에서 네트워크 형성



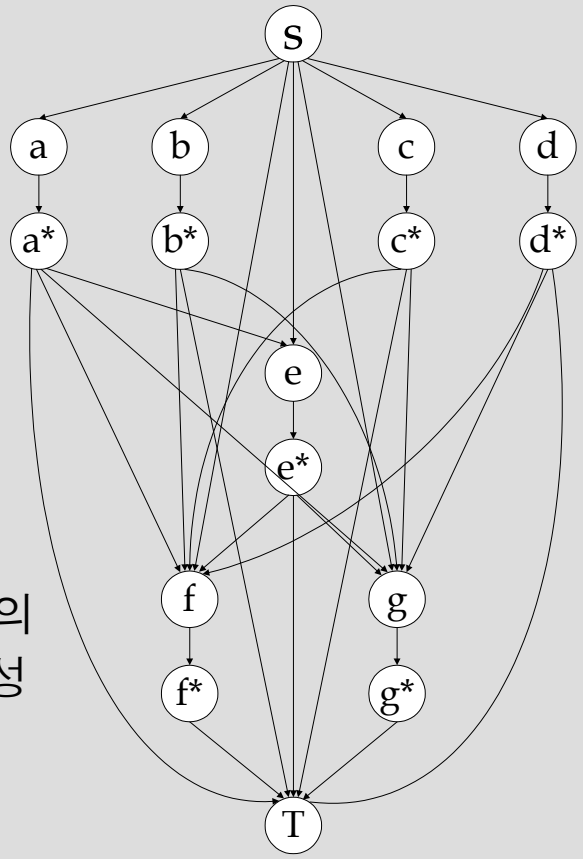
호환가능 그래프



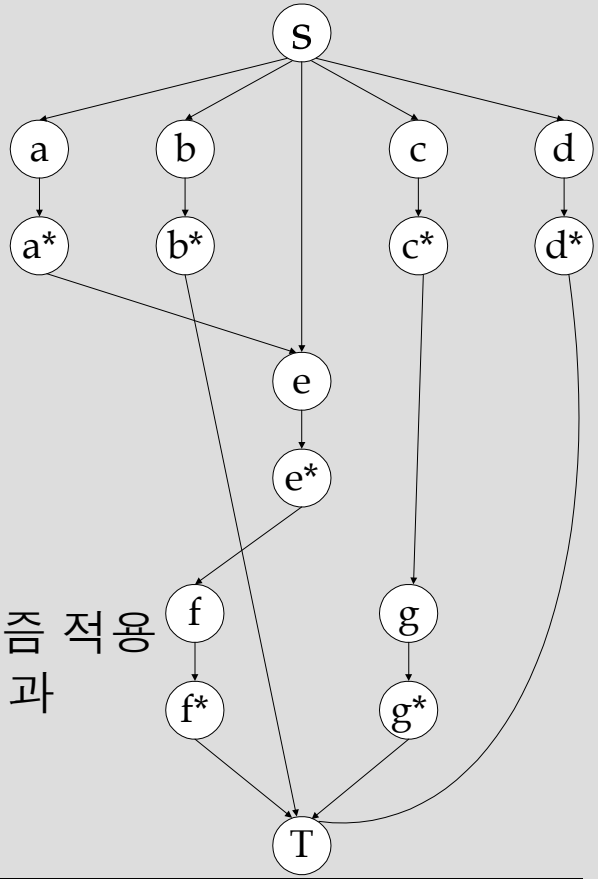
노드 분리 전의 네트워크 형성

노드분리와 알고리즘 적용

노드 분리 후의
네트워크 형성



알고리즘 적용
결과



적용 결과

PATH 1 : S-a-e-f-T REG1 : a, e, f

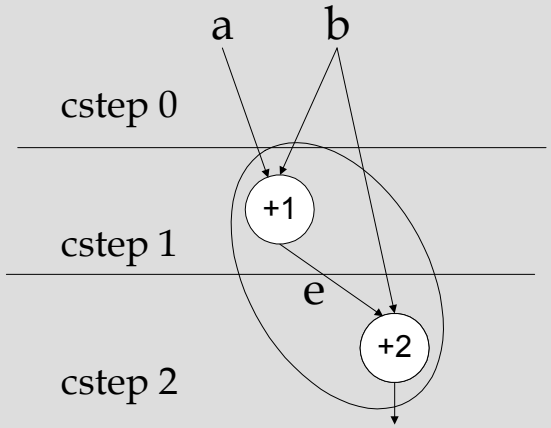
PATH 2 : S-b-T REG2 : b

PATH 3 : S-c-g-T REG3 : c, g

PATH 4 : S -d-T REG4 : d

저전력을 위한 리소스 할당 방법

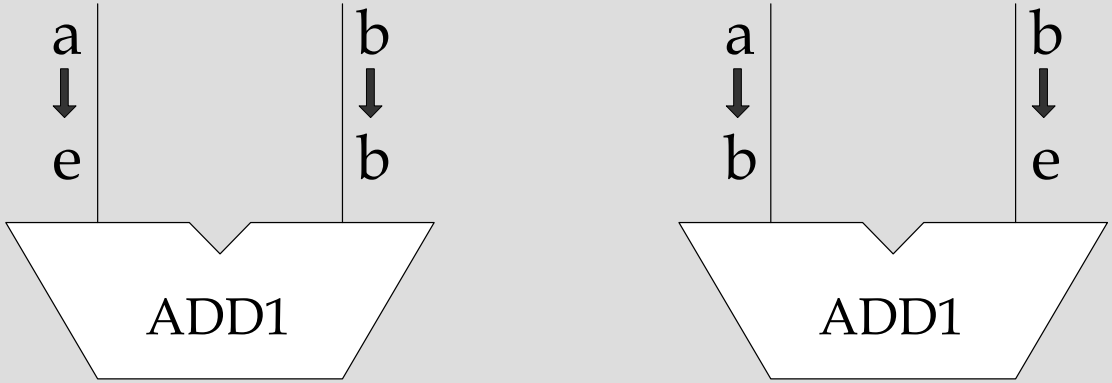
두 연산자를 공유시
발생하는 일련의 입력



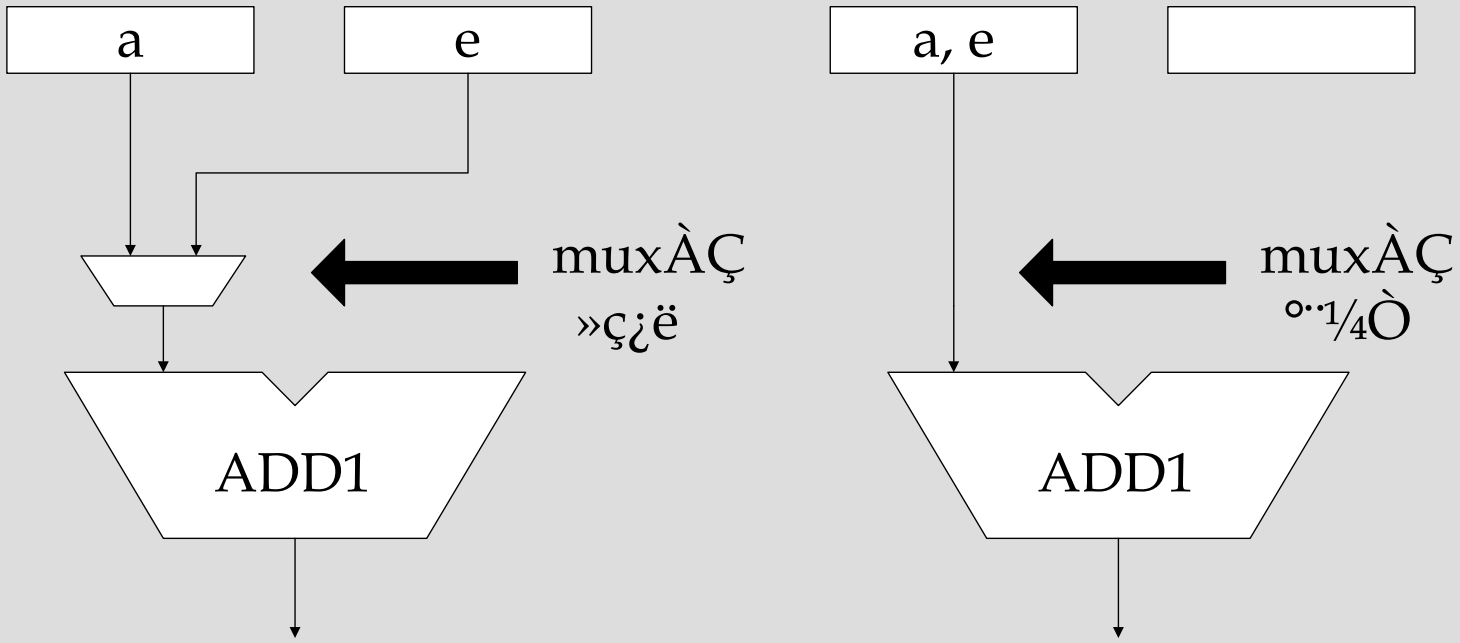
(a,e) +(b, b),

(a,b)+(b+e)

중 작은 경우
선택



변수의 저장에 따른 멀티플렉서의 증가와 감소



리소스 할당을 위한 가중치와 캐패시티

입력(input) : V, Network for resource sharing
출력(output) : 분리된 V개의 경로

$C_{ij}=1, B_{ij}=L$

$C_{ij}=1, B_{ij}=W$

W : -[M - ($Wai*N$ + $Wmux*K$)] (edge weight)

$Wmux$: 연결 구조 가중치

K : 정규화하기 위한 상수

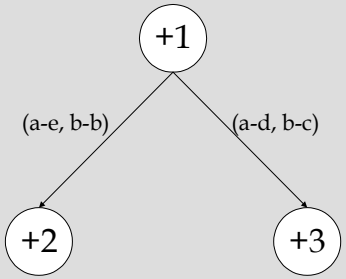
$Wmux$ 0: 변수 i, j가 같은 레지스터에 할당되고 모듈의 동일 입력단으로 할당될시

: 1: 변수 i, j가 다른 레지스터에 할당되고 모듈의 동일 입력단으로 할당될시

V: 리소스의 수

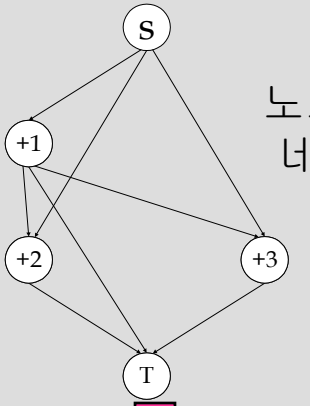
리소스 할당을 위한
최소 흐름 비용 알고리즘 적용 과정

호환 가능
그라프

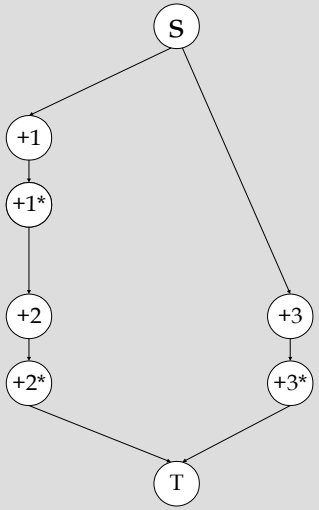


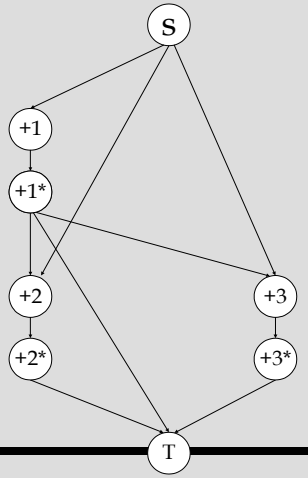
PATH 1 :
S-1-2-T
adder 1 : +1 , +2

PATH 2 :
S-3-T
adder 2 : +3



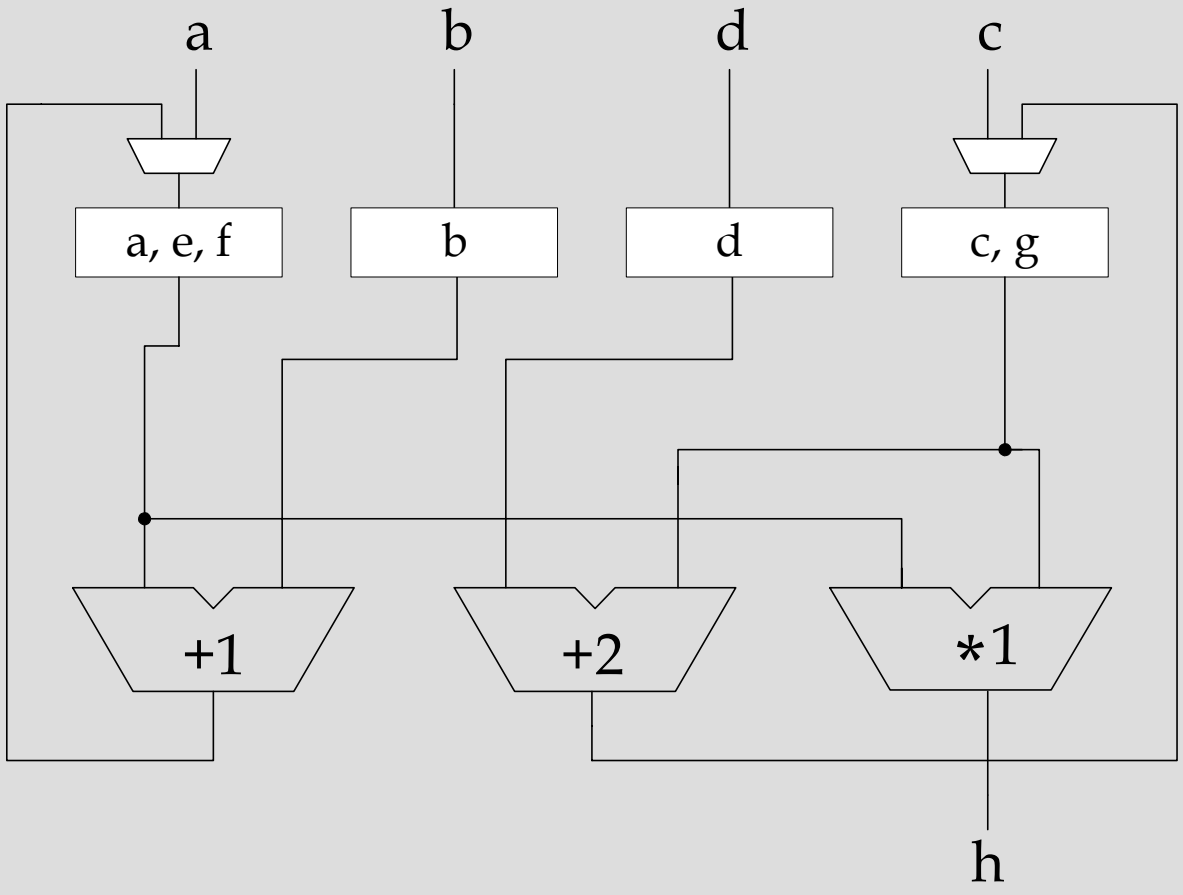
노드 분리 전의
네트워크형성



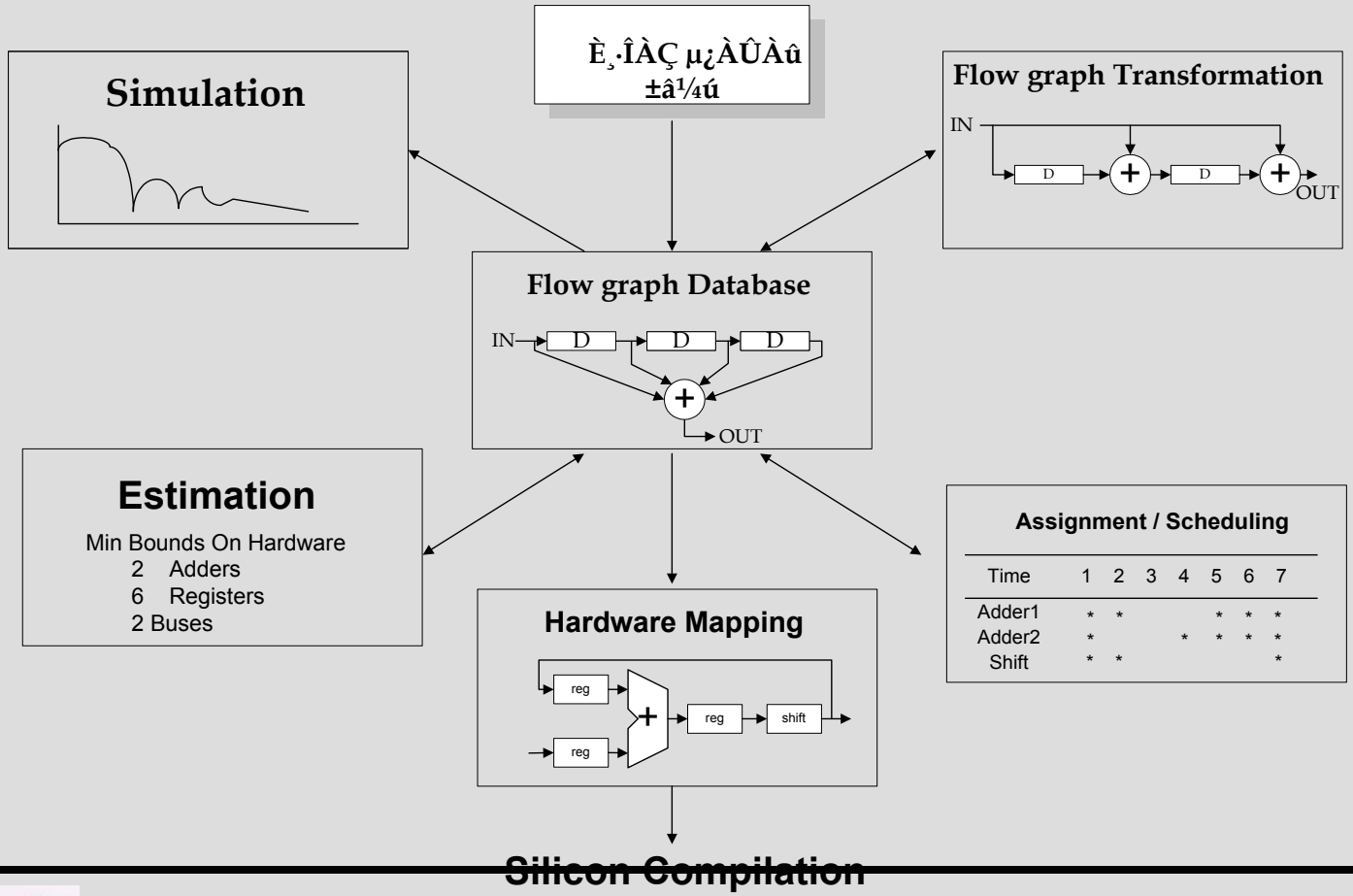


노드 분리 후의
네트워크 형성

레지스터와 리소스 할당 후의 최종 데이터 경로



실험 과정


È¸·ÎÀÇ µ¿ÀÛÀû ±â¼ú
Simulation
Flow graph Transformation
IN
D
D
OUT
Flow graph Database
IN
D
D
D
OUT
Estimation
Min Bounds On Hardware
2 Adders
6 Registers
2 Buses
Assignment / Scheduling
Time 1 2 3 4 5 6 7
Adder1 * * * * *
Adder2 * * * * * *
Shift * * *
Hardware Mapping
reg
reg
reg
shift
Silicon Compilation

스위칭율 계산 (Hamming Distance ratio , W_{sa})

- CDFG 기능적 시뮬레이션
- 두 변수의 exclusive-OR
- bit-width로 정규화
- 논리 수준이나 레이아웃 수준 보다 빠른 측정

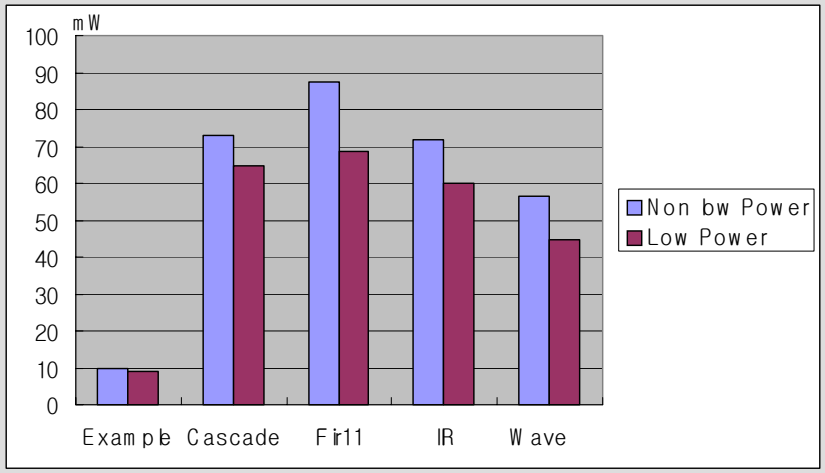
벤치마크 회로의 특성

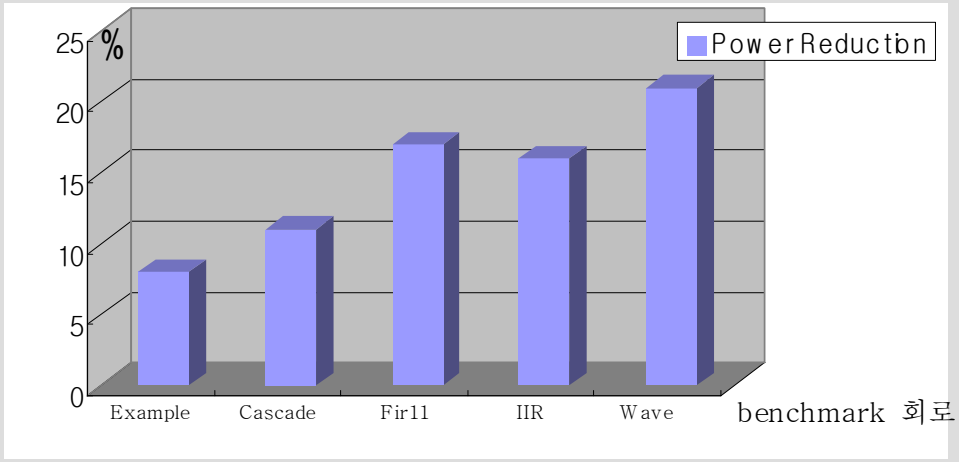
benchmark	mult(*)	add(+)	sub(-)	critical path
cascade	7	7	.	6
fir 11	11	10	.	11
iir5	10	10	.	8
wave	7	8	3	8

Resource allocation

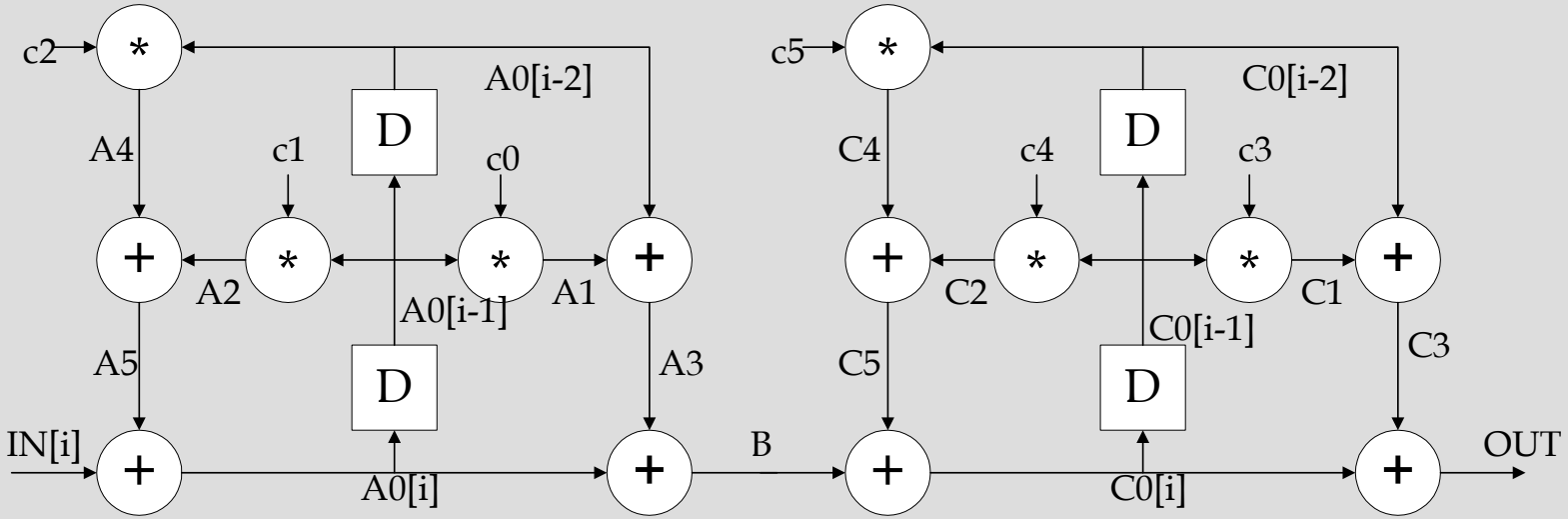
+(2), *(2), reg(7)
+(2), *(2), reg(7)
+(2), *(2), reg(7)
+(2) *(2),sub(1), reg(7)

실험 결과

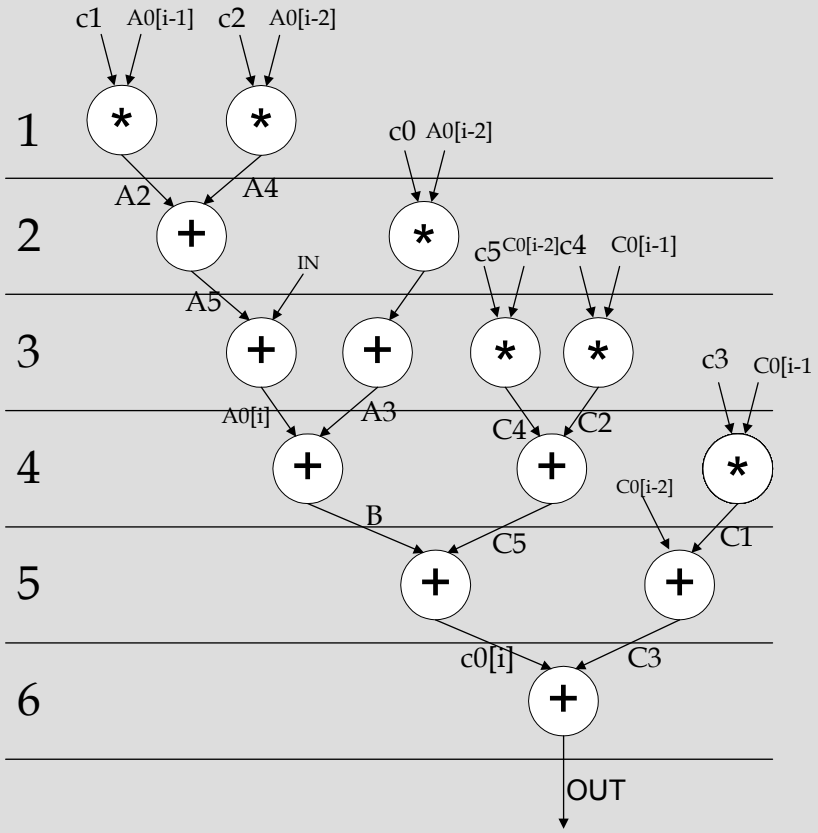

mW
100
90
80
70
60
50
40
30
20
10
0
Example
Cascade
Fir11
IR
Wave
Non bw Power
Low Power


%
25
20
15
10
5
0
Power Reduction
Example
Cascade
Fir11
IIR
Wave
benchmark 회로

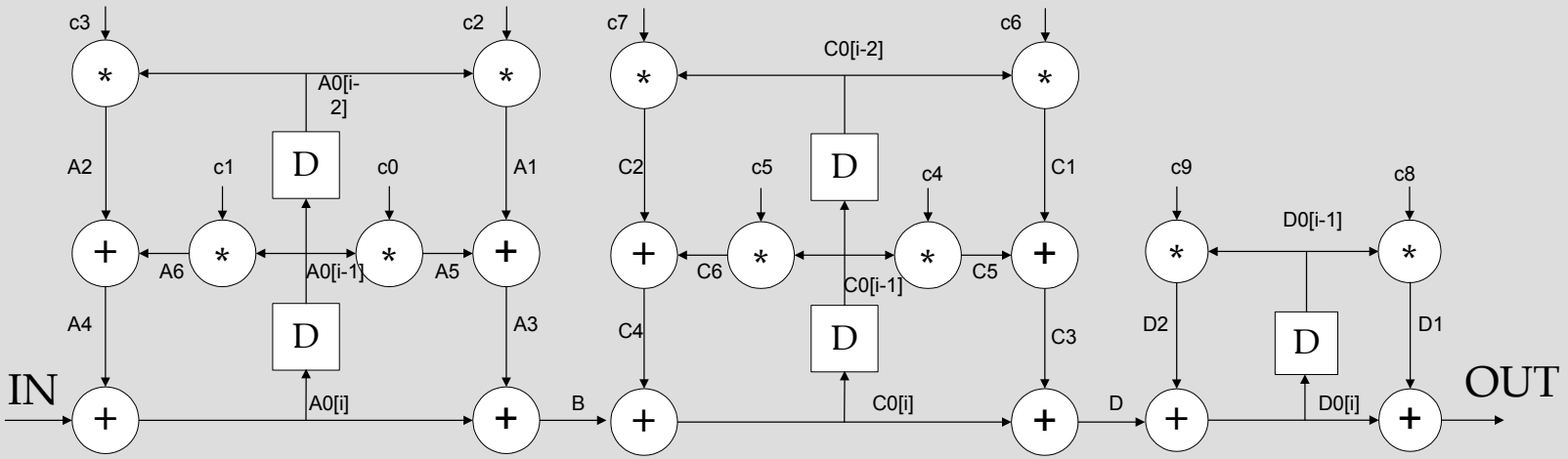
Cascade Filter



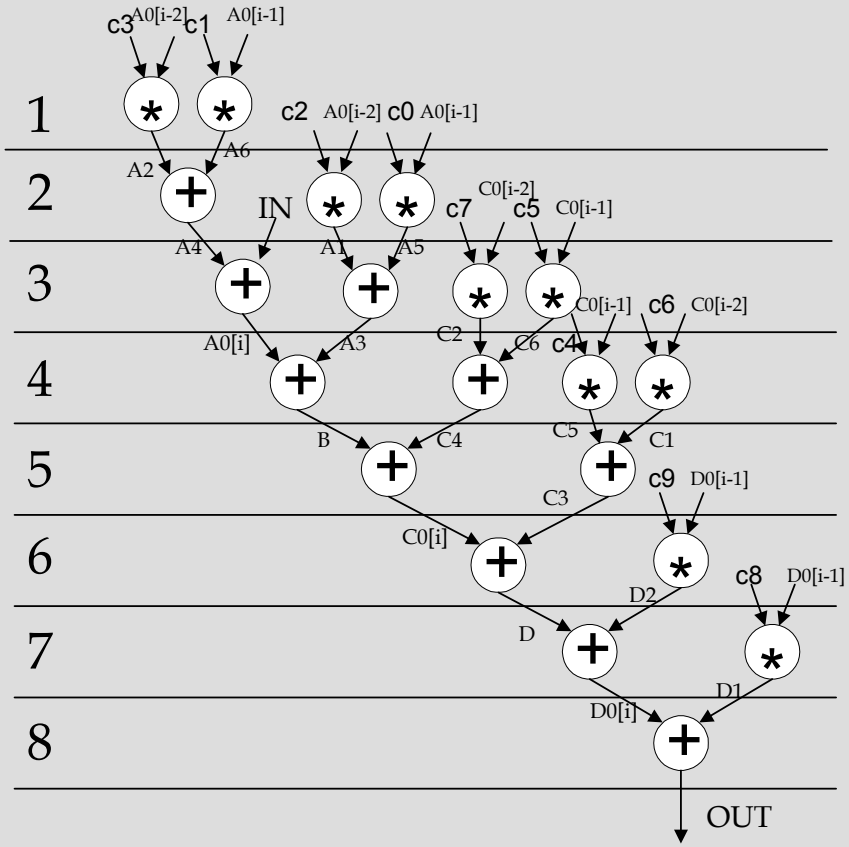
Cascade Filter Scheduling



Infinite Impulse Response Filter



IIR Filter Scheduling



결 론

- 스위칭 동작 최소화를 위해 해밍거리(Hamming distance) 의 목적 함수
- 저전력 구현을 위한 스케쥴링, 리소스 할당과정
- 평균 15%의 전력 감소
- 제한된 시간내의 최적의 결과 알고리즘 적용
  (polynomial time and optimal solution algorithm)
- 고성능 저전력 TOP-DOWN 상위수준 설계에
  적용 (DSP, Microcontroller, ASIC, etc)

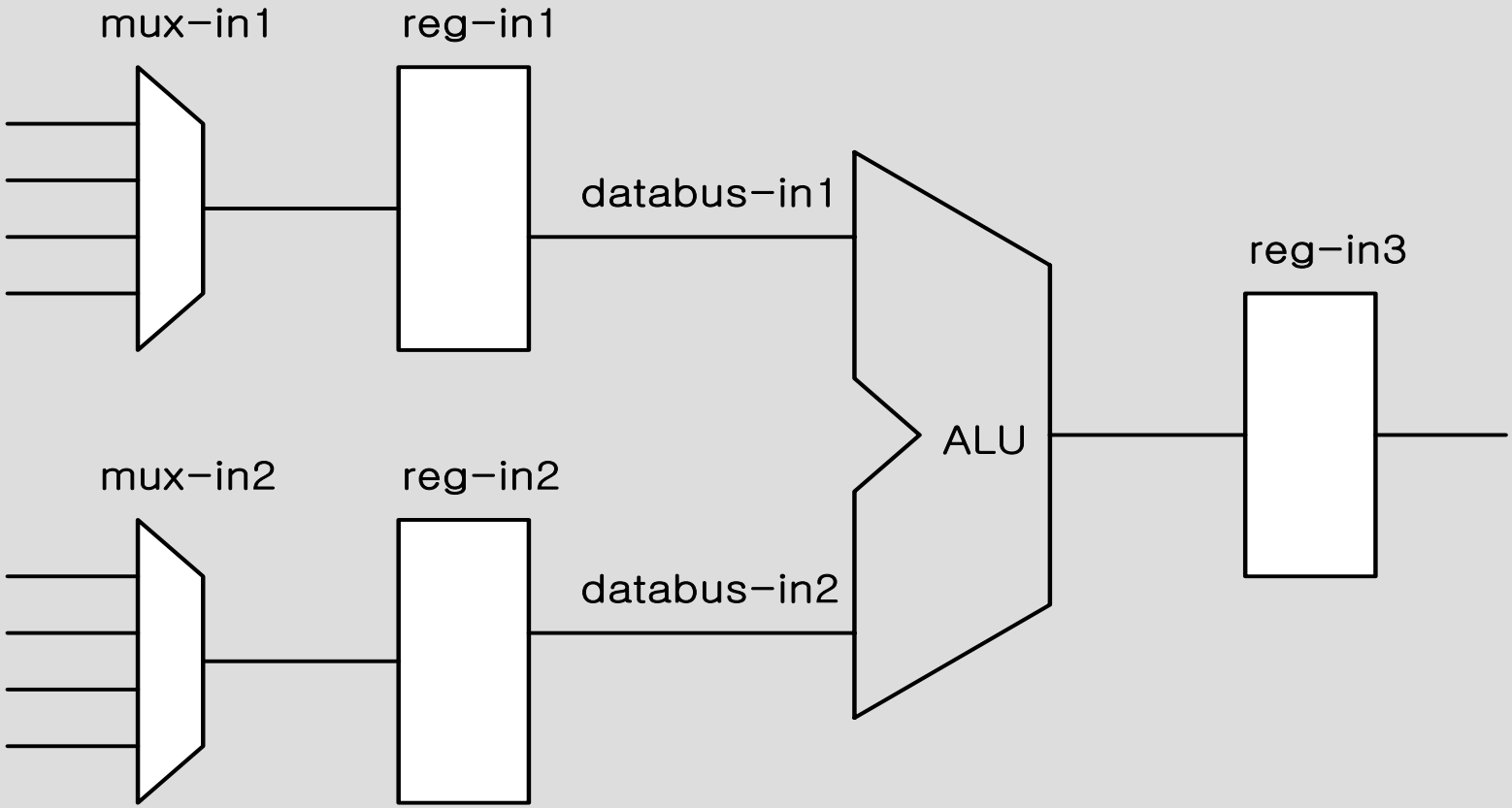
Reducing of bus transitions with bit-swapping for DSP

- Programmable DSP에서 전력 소비를 감소시키기 위한 새로운 bit-swapping 알고리즘을 제안한다.
- DSP같은 파이프라인 프로그래머블 프로세서에서 전력소비의 주성분을 형성하는 것은 외부/내부 버스이므로 모두에서 어드레스 전력 감소를 해야한다.
- ALU의 입력버스에서의 전력소비 감소 알고리즘을 제안

Motivation

- Bus switchings/transition은 bus-based system에서 power dissipation의 주성분을 형성한다. off-chip 구동에서의 power consumption은 전체 chip power의 70%
- CPU와 메모리에서 레지스터와 ALU같은 Datapath 사이의 상호연결을 하는 클럭라인과 내부버스를 포함한다. ALU 입력으로 제공되는 bus에서 signal transition을 감소시키는 bus에서의 전력 손실을 감소시킬 뿐만 아니라 ALU자체의 전력 소비를 감소시킨다

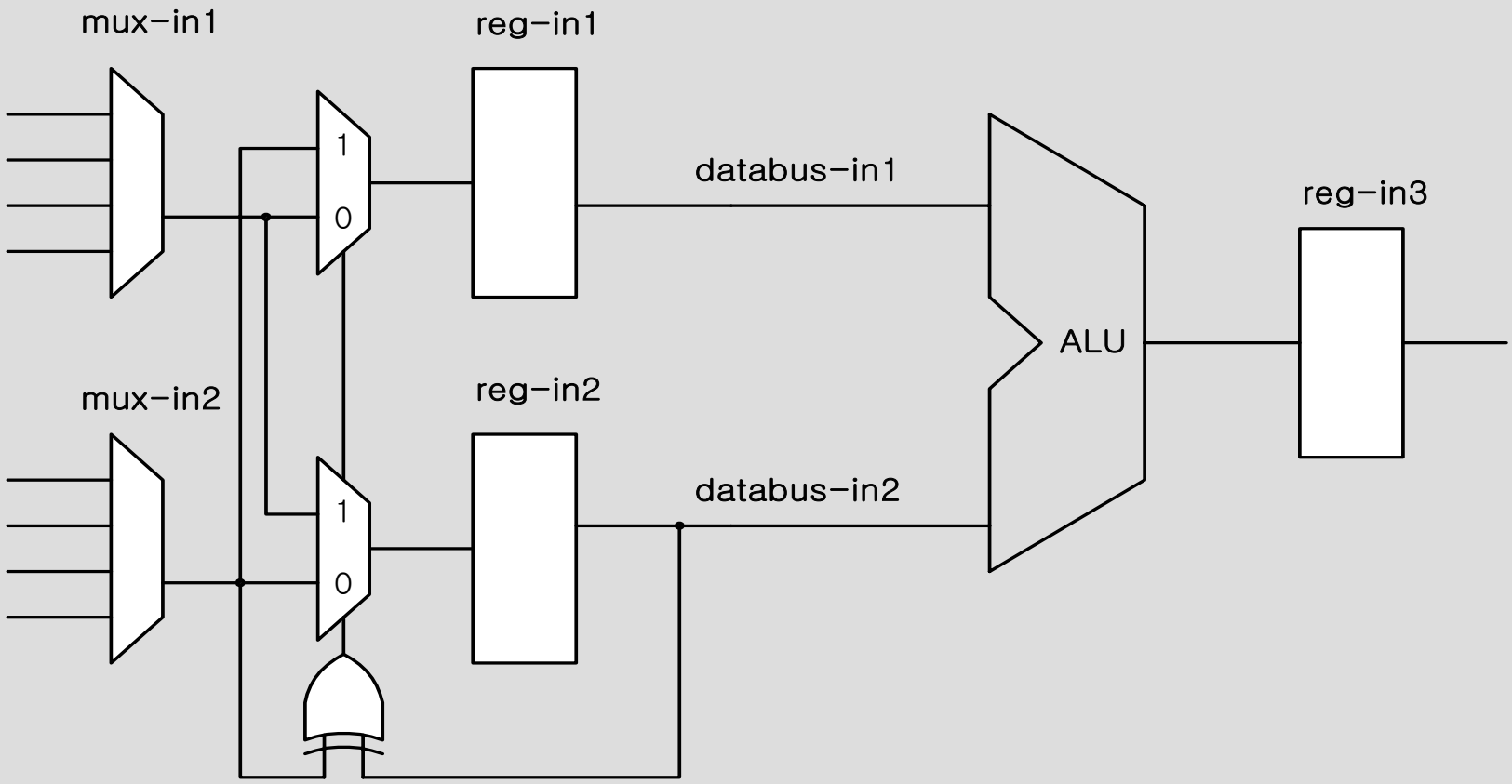
Reducing Power Dissipation in the Register-Datapath Buses



Bitwise commutativity property

- Y1 = 01000010 + 11001101 = A1 + B1
- Y2 = 10111001 + 01101010 = A2 + B2
- C1 = Y1(A2) EXOR Y2(B2)
- bit-wise swapping a bit inY2(A2) and a bit in Y2(B2) when a bit in C1 is “1”

Bit-wise Swapping Logic



실험 결과

- Alg1: 29%
- register입력 bus에서의 signal transition 감소가 ALU에서의 power 감소
- 추가된 회로로 인한 전체 회로에 미치는 다른 문제점 등을고려
- 응용분야의 적용부분 및 적용가능성 여부 검증.
- simulation에서의 신뢰도를 높이기 위해 많은 testvector로 test.

참고문헌

[1] D. Gajski and N. Dutt, High-level Synthesis : Introduction to Chip and System Design. *Kluwer Academic Publishers*, 1992.

[2] G. D. Micheli, Synthesis and Optimization of Digital Circuits. New York : *McGraw Hill. Inc*, 1994.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-Power CMOS digital design", *IEEE J. of Solid-State Circuits*, pp. 473-484, 1992.

[4] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformation," *IEEE Tr. on CAD/ICAS*, pp. 12-31, Jan. 1995.

[5] E. Musool and J. Cortadella, "Scheduling and resource binding for low power", *Int'l Symp on Synstem Syntheiss*, pp. 104-109, Apr. 1995.

[6] Y. Fang and A. Albicki, "Joint scheduling and allocation for low power," *in Proc. of Int'l Symp. on Circuits & Systems*, pp. 556-559, May. 1996.

[7] J. Monteiro and Pranav Ashar, "Scheduling techniques to enable power management", *33rd Design Automation Conference*, 1996.

[8] R. S. Martin, J. P. Knight, "Optimizing Power in ASIC Behavioral Synthesis", *IEEE Design & Test of Computers*, pp. 58-70, 1995.

[9] R. Mehra, J. Rabaey, "Exploting Regularity for Low Power Design", *IEEE Custom Integrated Circuits Conference*, pp.177-182. 1996.

[10] A. Chandrakasan, T. Sheng, and R. W. Brodersen, "Low Power CMOS Digital Design", *Journal of Solid State Circuits*, pp. 473-484, 1992.

[11] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," *in Proc. of Int'l Symp. on Low Power Design*, pp. 197-202, Apr. 1994.

[12] A. Raghunathan and N. K. Jha, "An iterative improvement algorithm for low power data path synthesis," *in Proc. of Int'l Conf. on Computer-Aided Design*, pp. 597-602, Nov. 1995.

[13] R. Mehra, J. Rabaey, "Low power architectural synthesis and the impact of exploiting locality," *Journal of VLSI Signal Processing*, 1996.

참고문헌

[14] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Tr. on VLSI Systems*, pp. 42-55, Mar. 1996.

[15] A. Abnous and J. M. Rabaey, "Ultra low power domain specific multimedia processors," *in Proc. of IEEE VLSI Signal Processing Workshop*, Oct. 1996.

[16] M. C. Mcfarland, A. C. Parker, R. Camposano, "The high level synthesis of digital systems," *Proceedings of the IEEE*. Vol 78. No 2 , February, 1990.

[17] A. Chandrakasan, S. Sheng, R. Brodersen, "Low power CMOS digital design,", *IEEE Solid State Circuit*, April, 1992.

[18] A. Chandrakasan, R. Brodersen, "Low power digital CMOS design, *Kluwer Academic Publishers*, 1995.

[19] M. Alidina, J. Moteiro, S. Devadas, A. Ghosh, M. Papaefthymiou, "Precomputation based sequential logic optimization for low power," *IEEE International Conference on Computer Aided Design*, 1994.

[20] J. Monterio, S. Devadas and A. Ghosh, "Retiming sequential circuits for low power," *In Proceeding of the IEEE International Conference on Computer Aided Design*, November, 1993.

[21] F. J. Kurdahi, A. C. Parker, REAL: A Program for Register Allocation,: *in Proc. of the 24th Design Automation Conference*, ACM/IEEE, June. pp. 210-215, 1987.

[22] A. Wolfe. A case study in low-power system level design. In Proc.of the IEEE International Conference on Computer Design, Oct., 1995.

[23] T.D. Burd and R.W. Brothersen. Energy ecient CMOS micropro-cessor design. In Proc. 28th Annual Hawaii International Conf. On System Sciences, January 1995.

[24] A. Dasgupta and R. Karri. Simultaneous scheduling and binding for power minimization during microarchitectural synthesis. In Int. Symposium on Low Power Design, pages 69-74, April 1995.

[25] R.S. Martin. Optimizing power consumption, area and delay in behavioral synthesis. PhD thesis, Department of Electronics, Faculty of Enginering, Carleton University, March 1995.

[26] A. Matsuzawa. Low-power portable design. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, March 1996. Invited lecture.

[27] J.D. Meindl. Low-power microelectronics: retrospect and prospect. Proceedings of the IEEE 83(4):619-635, April 1995.