PENGOLAHAN PARALEL

Ernastuti

LATAR BELAKANG

- Banyak aplikasi2 membutuhkan kemampuan komputasi yang jauh lebih besar dari kemampuan komputer prosesor tunggal

- Ada 2 cara yang dapat dicapai untuk memenuhi kebutuhan ini :

1) *mengembangkan komputer prosesor tunggal menjadi lebih cepat*
2) *melakukan komputasi paralel.*

PENGOLAHAN PARALEL

Minat penelitian dalam Pengolahan paralel **diantaranya adalah sebagai berikut :**

1) arsitektur paralel

2) algoritma paralel

3) bahasa pemograman paralel

4) analisis kinerja paralel.

4 langkah penyelesaian masalah komputasi secara paralel:

Pertama , mengerti dasar komputasi didalam bidang aplikasi tertentu.

Kedua , mendisain suatu algoritma paralel atau me-paralelkan algoritma sekuensial yang sudah ada.

Ketiga , memetakan algoritma paralel kedalam arsitektur komputer paralel yang sesuai,

Keempat , melibatkan penulisan program paralel dengan memanfaatkan suatu pendekatan pemrograman paralel yang aplikatif.

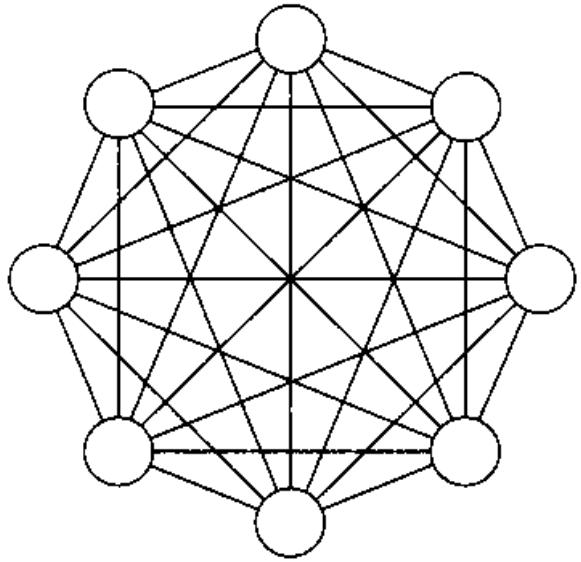
- Pokok persoalan utama arsitektur paralel adalah terletak pada disain

  jaringan interkoneksi prosesor

- Idealnya didalam jaringan,

  setiap prosesor didisain terhubung dengan semua prosesor lainnya.

- Pada *graph*,
  jaringan interkoneksi ideal
  digambarkan sebagai
  complete graph :
  (fully connected)



Untuk p prosesor pada jaringan interkoneksi *complete graph* , jumlah *edge* penghubungnya adalah p x (p-1) *edge*.

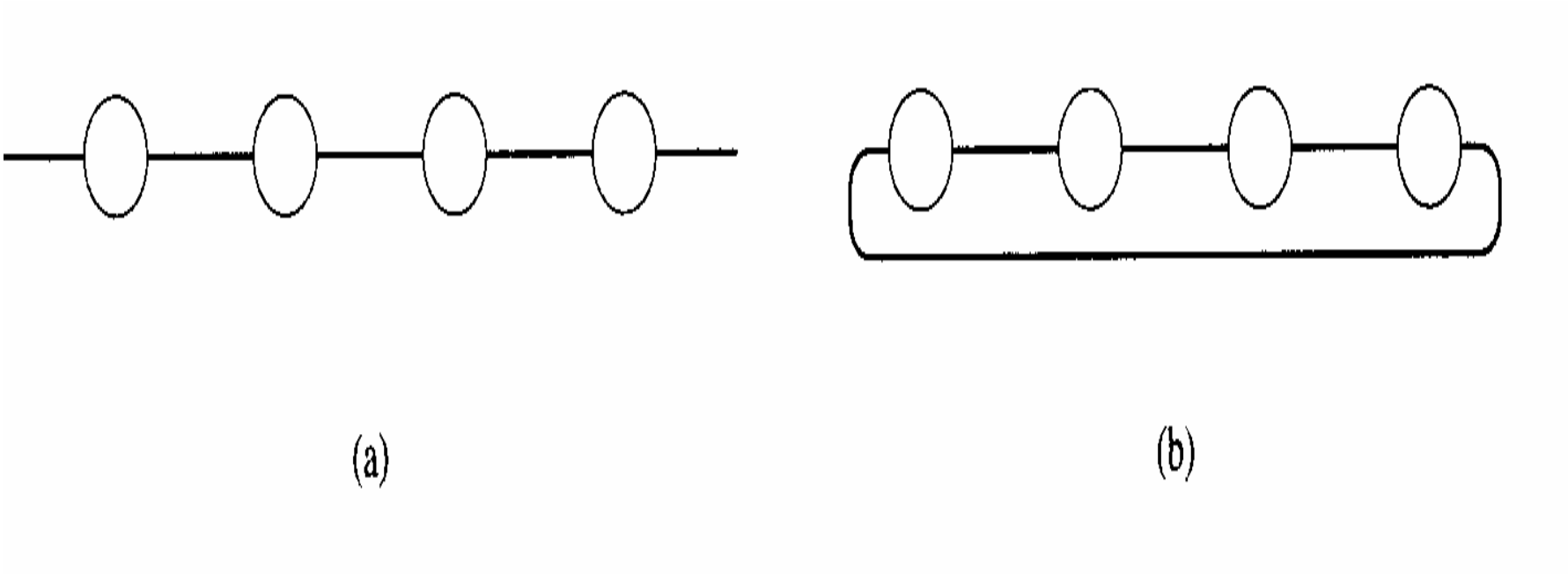
- Jaringan interkoneksi seperti ini jelas sangat mahal. (*semakin besar jumlah edge dikatakan semakin mahal*)

Topologi Model jaringan interkoneksi yang *lebih murah* dari *complete graph* yang ada saat ini antara lain adalah

- *linear & ring, shuffle exchange, hypercube,*
- *star, de bruijn, binary tree, delta,*
- *butterfly, mesh, omega* dan *pyramid*

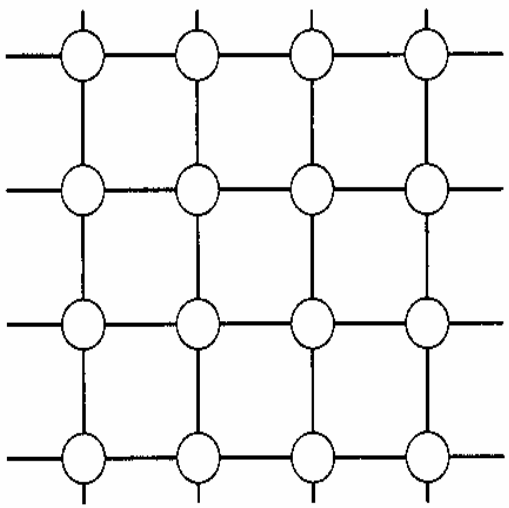
LINEAR + RING

Untuk mereduksi interconnect cost,
dicoba membuat jaringan yang lebih jarang (sparse) :

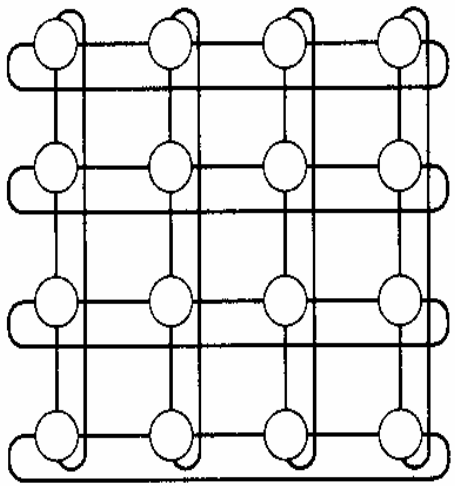


MESH + TORUS: 2D, 3D

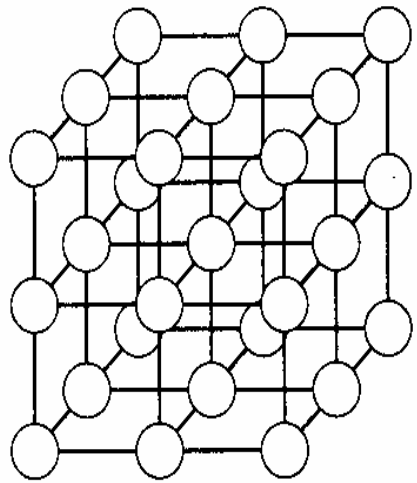
Kemudian diperluas ke suatu jaringan multidimensional :



(a)

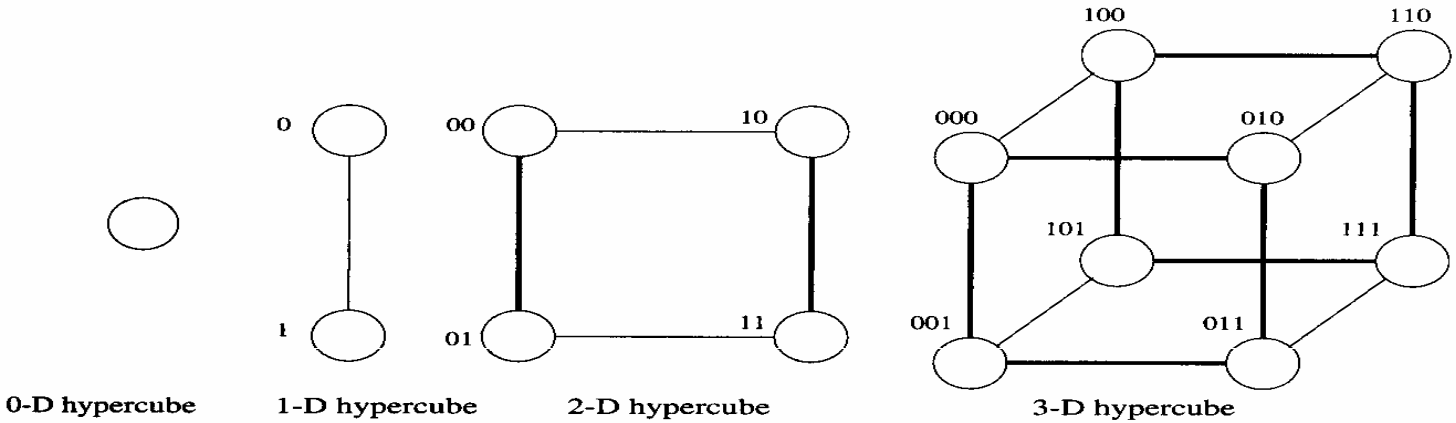


(b)

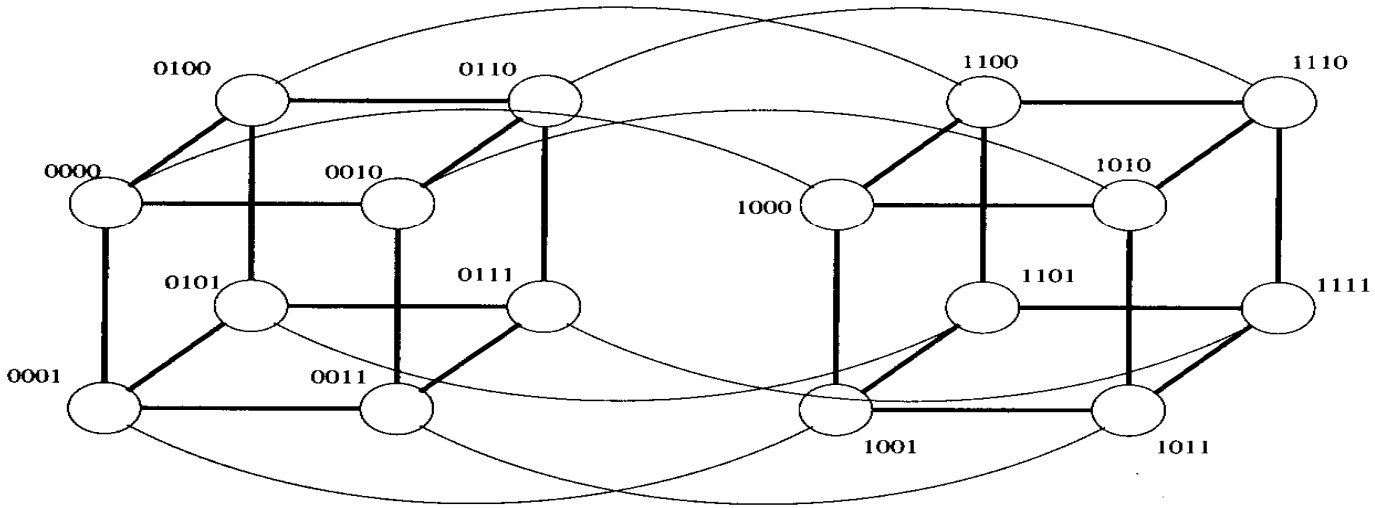


(c)

HYPERCUBE (n-CUBE)



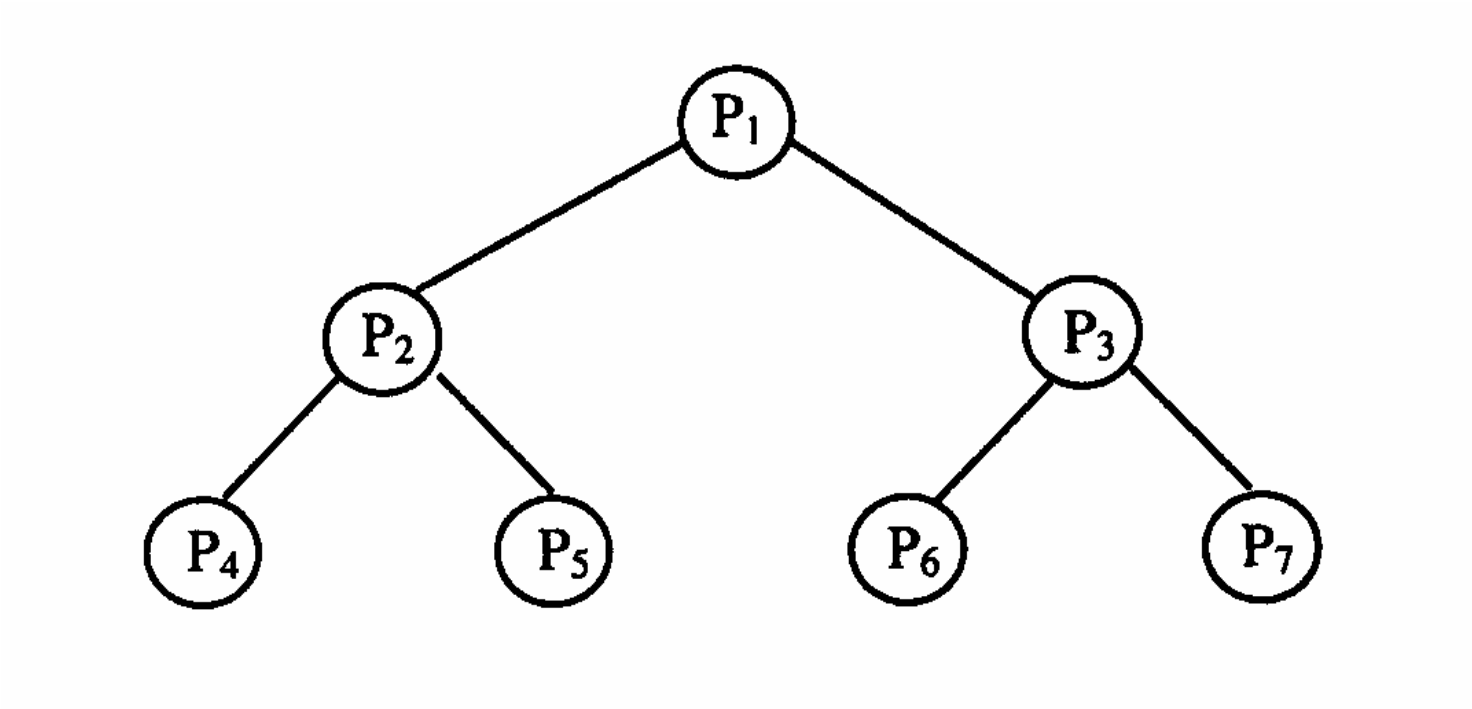
0-D hypercube 1-D hypercube 2-D hypercube 3-D hypercube



4-D hypercube

TREE

Pada jaringan TREE hanya ada satu jalur untuk setiap 2 simpul. Semakin tinggi Tree, semakin beresiko akan terjadi komunikasi bottleneck pada level-level yang tinggi dalam tree.

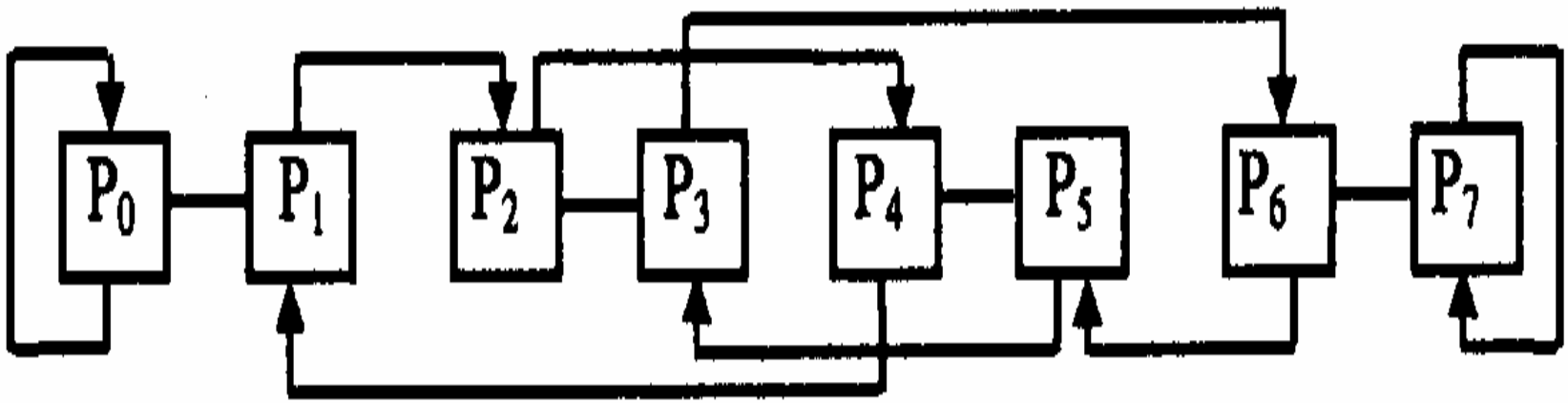


SHUFFLE EXCHANGE

Perfect shuffle menghubungkan processor Pi and Pj dengan cara komunikasi satu arah sbb :

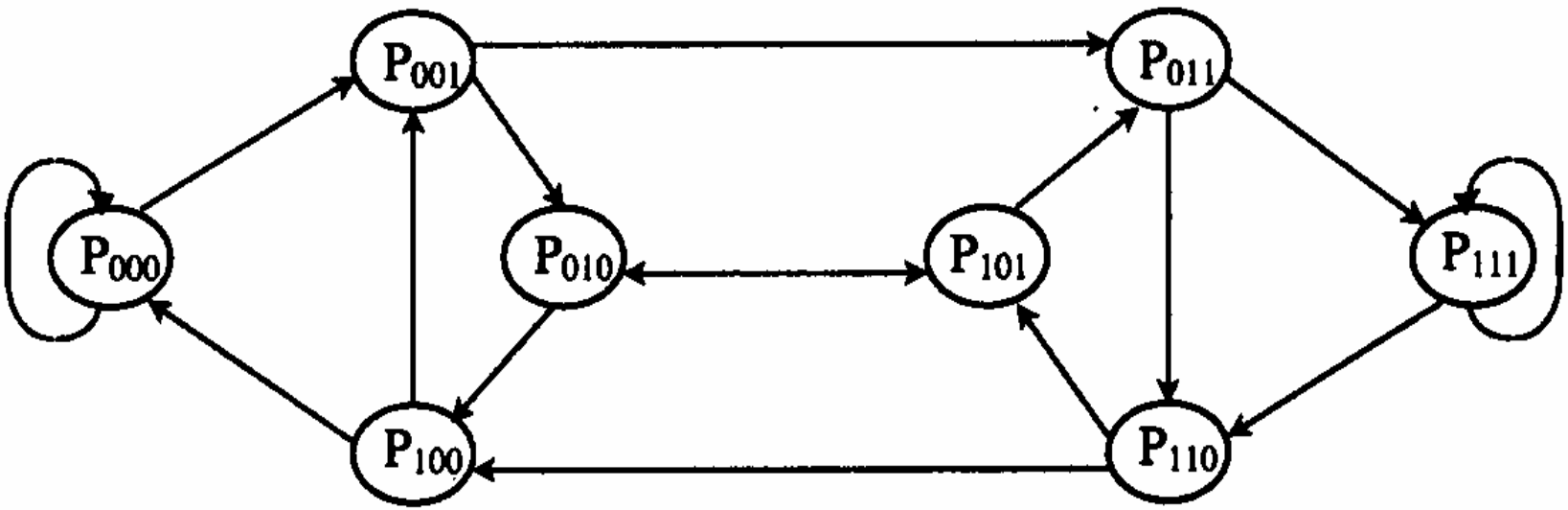
j = 2*i , $0 \leq i \leq N/2 - 1$ atau

j = 2*i + 1 – N , lainnya.



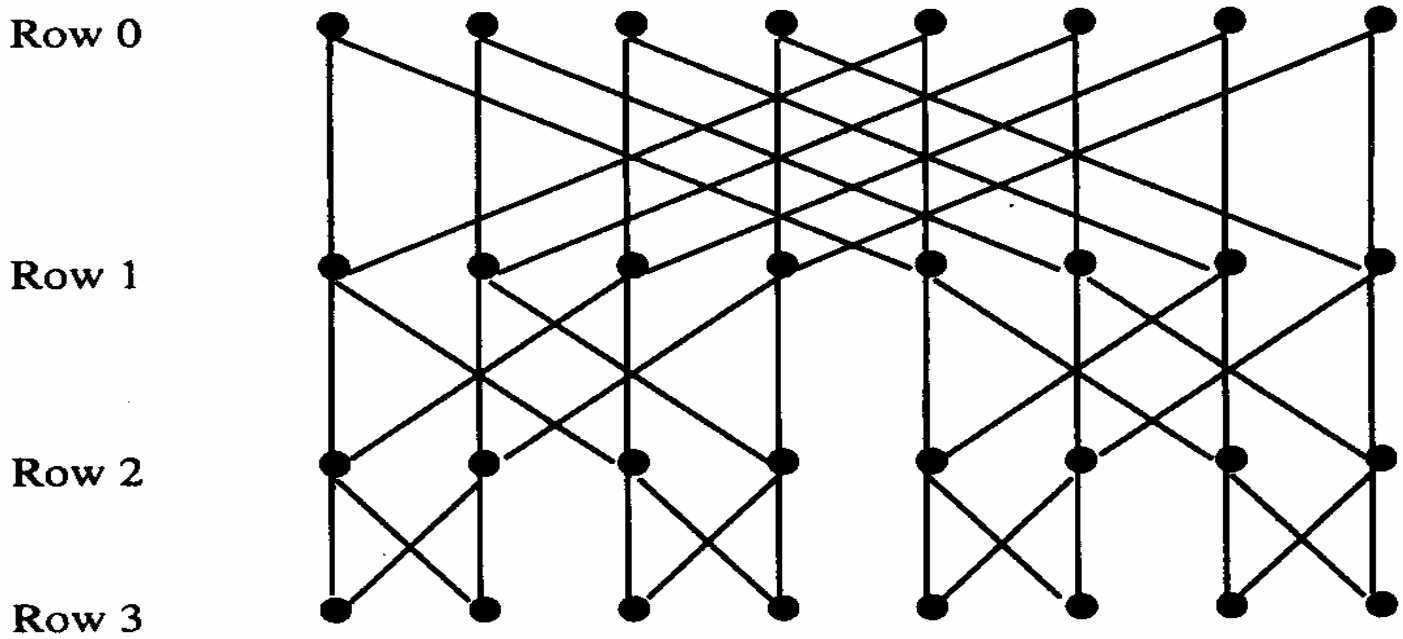
DE BRUIJN

A network consisting of $N = d^k$ processors, each labeled with a k-digit word $(a_{k-1}\ a_{k-2} \ldots a_1\ a_0)$ where a_j is a digit (radix d), i.e. a_j is one of (0, 1, ... , d-1)
The processors directly reachable from $(a_{k-1}\ a_{k-2} \ldots a_1\ a_0)$ are $(a_{k-2} \ldots a_1\ a_0\ q)$ and $(q\ a_{k-1}\ a_{k-2} \ldots a_1)$ where q is another digit (radix d).
Berikut adalah jaringan de Bruijn untuk d=2 dan k=3



BUTTERFLY

A Butterfly network is made of $(n + 1)*2^n$ processors organized into $n+1$ rows, each containing 2^n processors.



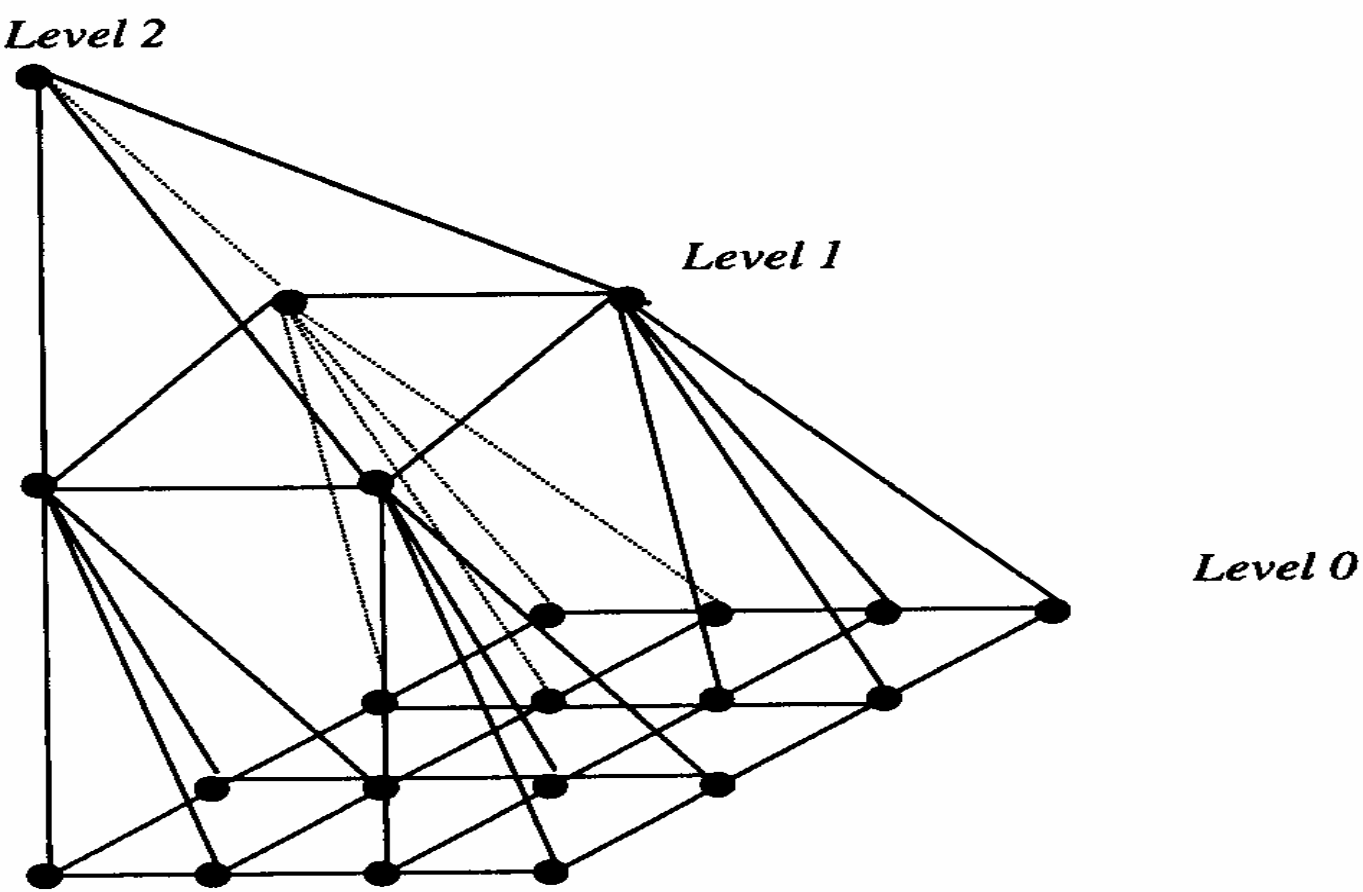
Rows are labeled 0…n. Each processor has 4 connections to other processors (except processors in top and bottom row).

Processor P(r, j), i.e. processor number j in row r is connected to P(r-1, j) and P(r-1, m) where m is obtained by inverting the r^{th} significant bit in the binary representation of j.

PYRAMID

A pyramid consists of $(4^{d+1} - 1)/3$ processors organized in d+1 levels so as:

- **Levels are numbered from d down to 0**
- **There is 1 processor at level d**
- **Every level below d has four times the number of processors than the level immediately above it.**



Untuk membandingkan model jaringan interkoneksi diperlukan beberapa kriteria pengukuran.

Kriteria

yang digunakan industri berkaitan dengan

komunikasi dan kompleksitas

pada jaringan interkonekasi

adalah sebagai berikut

Standard criteria used by industry:

- **Network diameter** = Max. number of hops necessary to link up two most distant processors
- **Network bisection width** = Minimum number of links to be severed for a network to be into two halves (give or take one processor)
- **Maximum-Degree of PEs** = maximum number of links to/from one PE
- **Minimum-Degree of PEs** = minimum number of links to/from one PE

- 1) *Diameter* of *the Network* = jarak maksimum jalur terpendek diantara semua prosesor didalam jaringan
- 2) *Degree of processor* = jumlah maksimum *edge* penghubung yang keluar/masuk dari/ke prosesor
- 3) *Bisection width of the network* = Jumlah edge minimum yang diputus dari jaringan sedemikian sehingga network terbagi dua sama besar

Selain itu ada kriteria lain :

Suatu model jaringan interkoneksi dikatakan lebih baik dari yang lain bila

- lebih efisien (*efficient)* ,
- lebih tepat/cocok (*convenient)*,
- lebih mudah diimplementasi (*regularity*),
- lebih mudah diperluas (*expandable/modularity*)
- dan/atau tidak berpotensi *bottleneck*.

Kenyataannya tak ada jaringan interkoneksi yang memenuhi semua kriteria ini.

COMPARISON OF INTERCONNECTION NETWORKS

Intuitively, one network topology is more desirable than another if it is :

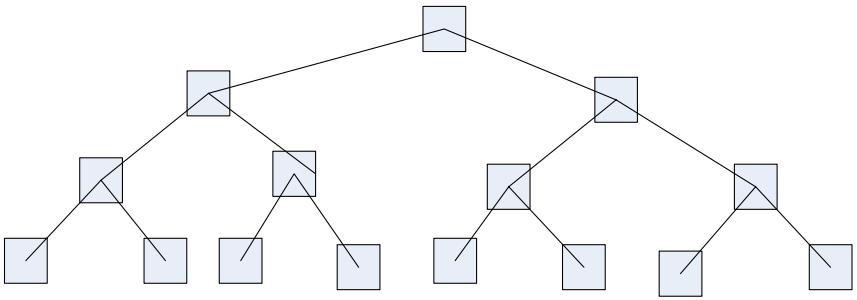
- More efficient
- More convenient
- More regular (i.e. easy to implement)
- More expandable (i.e. highly modular)
- Unlikely to experience bottlenecks

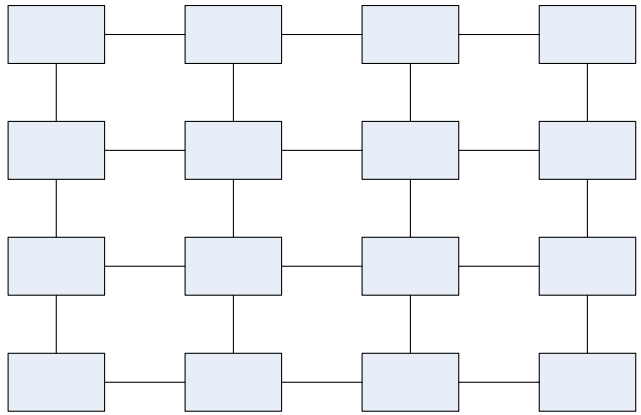
- Clearly no one interconnection network maximizes all these criteria.
  Some tradeoffs are needed.

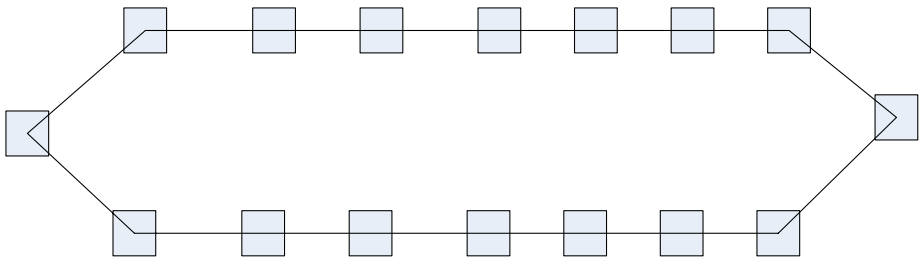
- Dengan menjaga diameter jaringan tetap kecil maka akan memberikan *lower bound* pada kompleksitas algoritma yang diimplementasikan pada jaringan.

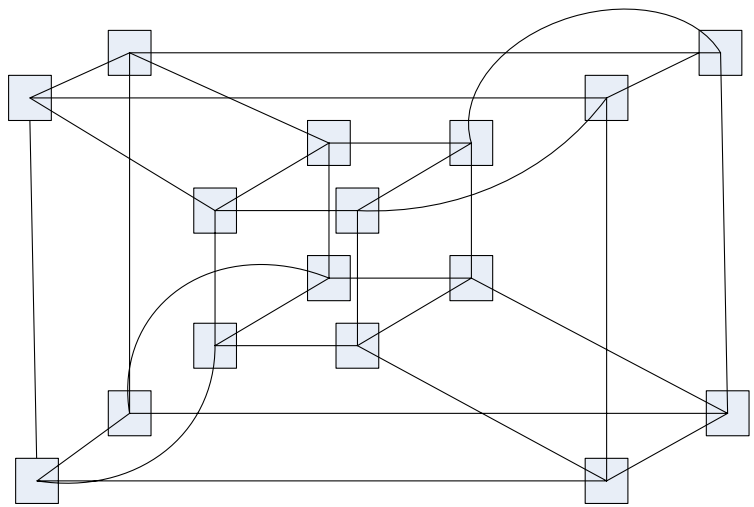
- Akibatnya , untuk menjaga diameter tetap kecil berarti diperlukan sejumlah *edge* penghubung lebih besar pada setiap prosesor.

Contoh topologi jaringan interkoneksi dalam arsitektur paralel yang umum saat ini :









Perbandingan jumlah edge, diameter, max-degree dan min-degree untuk 5 model topologi jaringan 16 node.

Topologi Jaringan	Jumlah node	Jumlah edge	Diameter	Max-degree	Min-degree
Complete Graph	16	240	1	15	15
Tree	16	14	6	3	1
Mesh	16	24	6	4	2
Ring	16	16	8	2	2
Hypercube	16	32	4	4	4

- Saat ini, komputer-komputer yang mendukukng komputasi paralel telah tersedia secara komersil dengan berbagai macam topologi.

- Topologi interkoneksi ***Hypercube*** merupakan topologi yang paling dominan/menonjol pada kelas komputer paralel ini.

- *Ametek, Floating Point System, Intel ScienticComputers, NCUBE* dan *Thingking Machines* adalah beberapa *vendor* dari komputer *Hypercube*.

TRADE-OFF

Model jaringan dapat diukur, salah satunya, dari ***trade-off*** :

Network cost = derajat * diameter

DIAMETER - DERAJAT

- Model jaringan dengan
  derajat simpul yang kecil,
  mempunyai ***diameter yang besar***.

- Kebalikannya,
  model jaringan yang mempunyai ***diameter kecil***
  biasanya memiliki ***derajat simpul yang besar***.

DIAMETER - DERAJAT

Tabel 1.1: Jumlah simpul, busur, diameter, derajat maksimum dan network cost pada 6 model jaringan.

Hypercube mempunyai karakteristik yang layak

TopologiJaringan	*Jumlah simpul*	*Jumlah busur*	*Diameter*	*Derajat maksimun*	*Network cost*
Fully connected	64	2016	1	63	63
Binary Tree	64	63	12	3	36
4×4 *Mesh*	64	112	14	4	56
Linear Array	64	63	63	2	126
Ring	64	64	32	2	64
Hypercube	64	192	6	6	36
Star	64	63	2	63	126

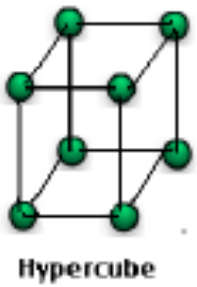
NETWORK COST

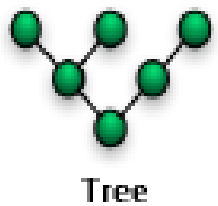
TopologiJaringan	*Jumlah simpul*	*Diameter*	*Derajat*	*Network Cost*
Fully connected	2^n	1	$2^n - 1$	$\approx 2^n$
Binary Tree	2^n	$(\frac{2^n}{2}) - 1$	3	$\approx 2^{n-1}$
2D Mesh	2^n	$2\sqrt{2^n} - 2$	4	$\approx 2^{\frac{n}{2}}$
Ring	2^n	$\lfloor \frac{2^n}{2} \rfloor$	2	$\approx 2^n$
Star	2^n	2	$2^n - 1$	$\approx 2^n$
Hypercube	2^n	n	n	n^2

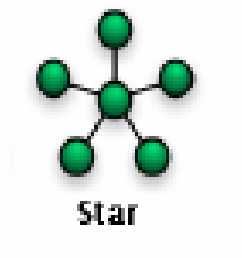
Hypercube mempunyai karakteristik yang layak

LATAR BELAKANG PENELITIAN

Topologi jaringan interkoneksi multiprosesor yang populer saat ini







Linear array

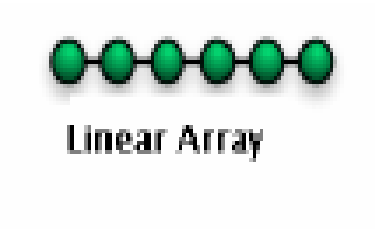
Ring

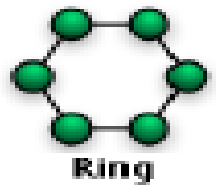
2D mesh

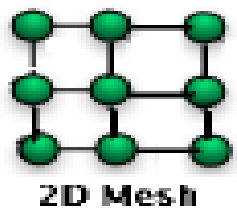
Hypercube

Tree

Star







Hypercube

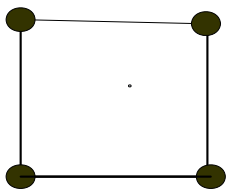
paling banyak menarik perhatian dan diteliti secara intensif

HYPERCUBE
dimensi 1,2,3 dan 4

Dimensi 1

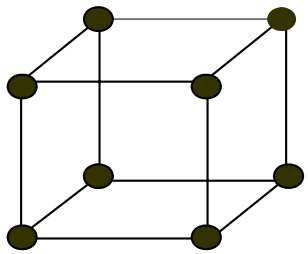
2

Dimensi 2



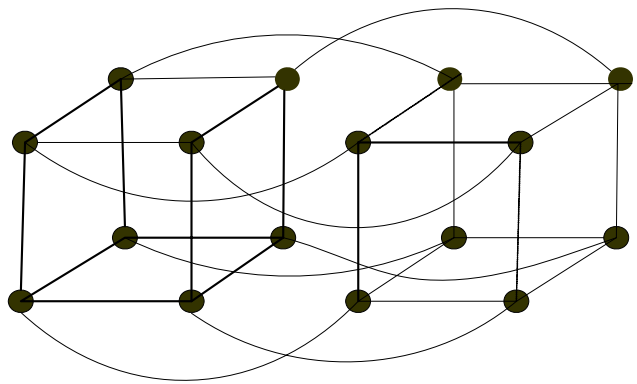
4

Dimensi 3



8

Dimensi 4



16

MODEL-MODEL KOMPUTASI yang mendasari komputer paralel

Pengklasifikasian oleh Flynn, dikenal sebagai Taksonomi Flynn, membedakan komputer paralel ke dalam empat kelas berdasarkan konsep aliran data (data stream) dan aliran instruksi (instruction stream), sebagai : SISD, SIMD, MISD, MIMD.

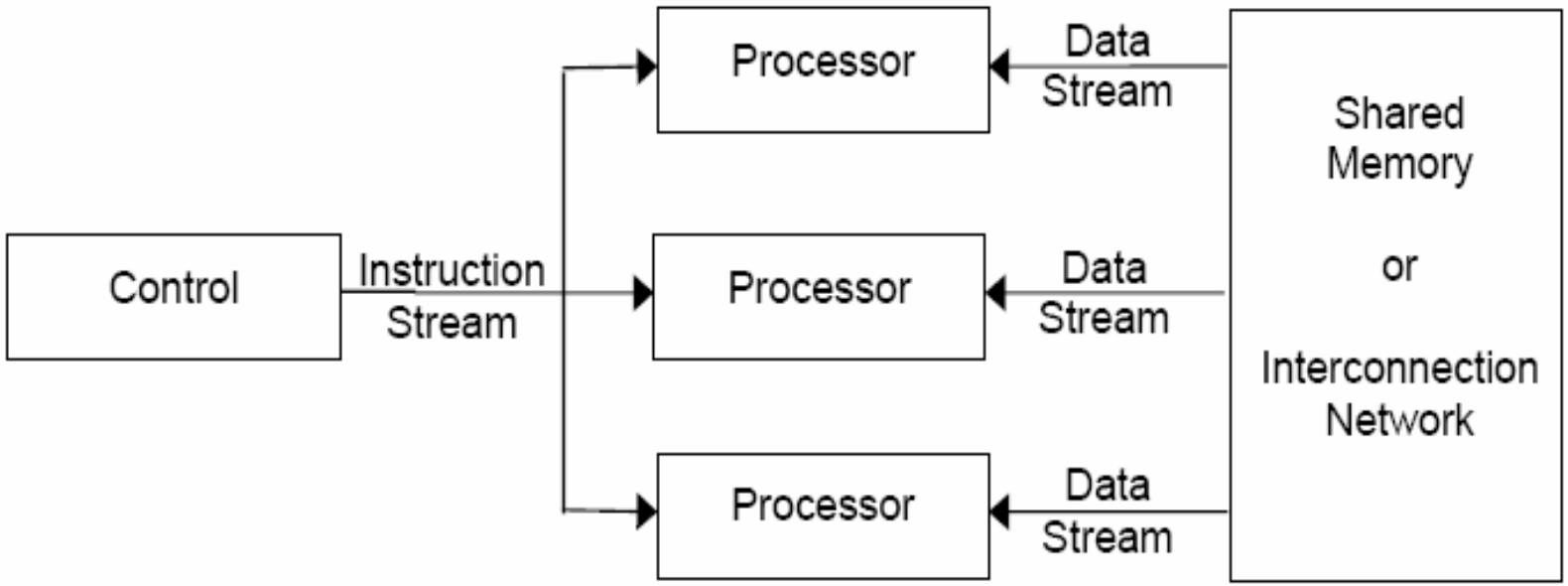
SISD (Single Instruction stream, Single Data stream)

Komputer tunggal yang mempunyai satu unit kontrol, satu unit prosesor dan satu unit memori.

[Akl 1989]

SIMD (Single Instruction stream, Multiple Data stream)

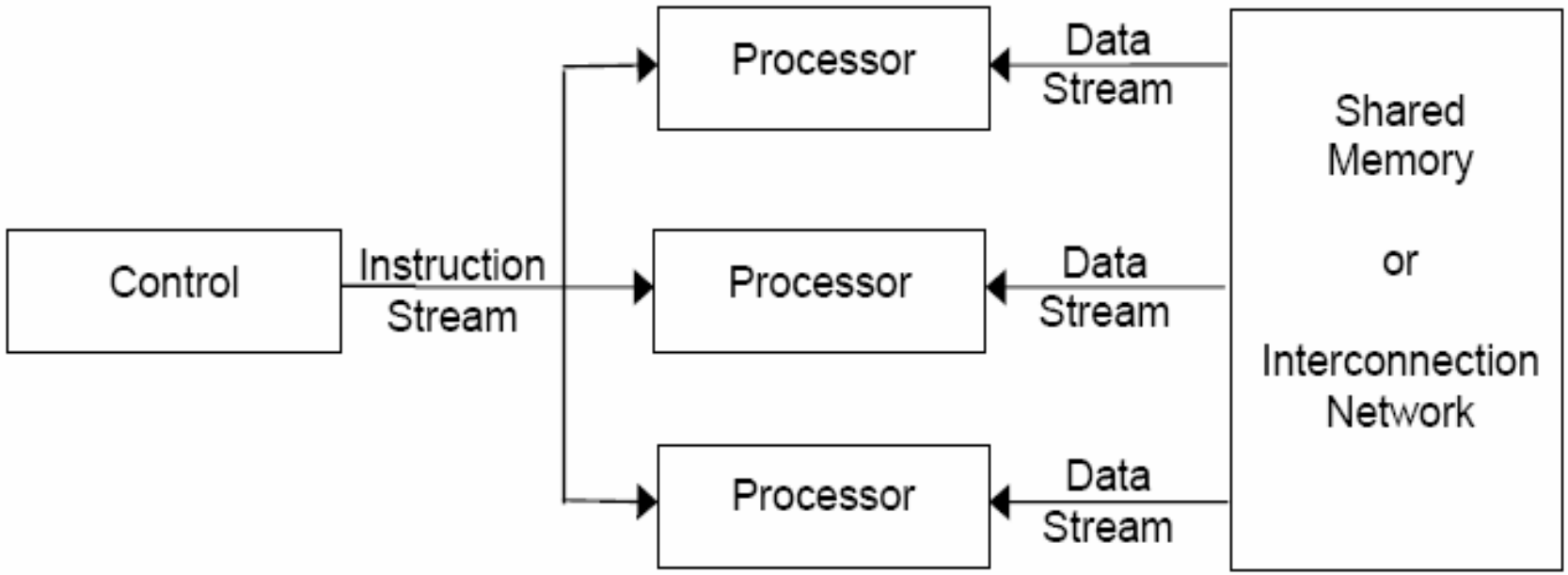
Komputer yang mempunyai beberapa unit prosesor di bawah satu supervisi satu unit common control. Setiap prosesor menerima instruksi yang sama dari unit kontrol, tetapi beroperasi pada data yang berbeda.



[Akl 1989]

MISD (Multiple Instruction stream, Single Data stream)

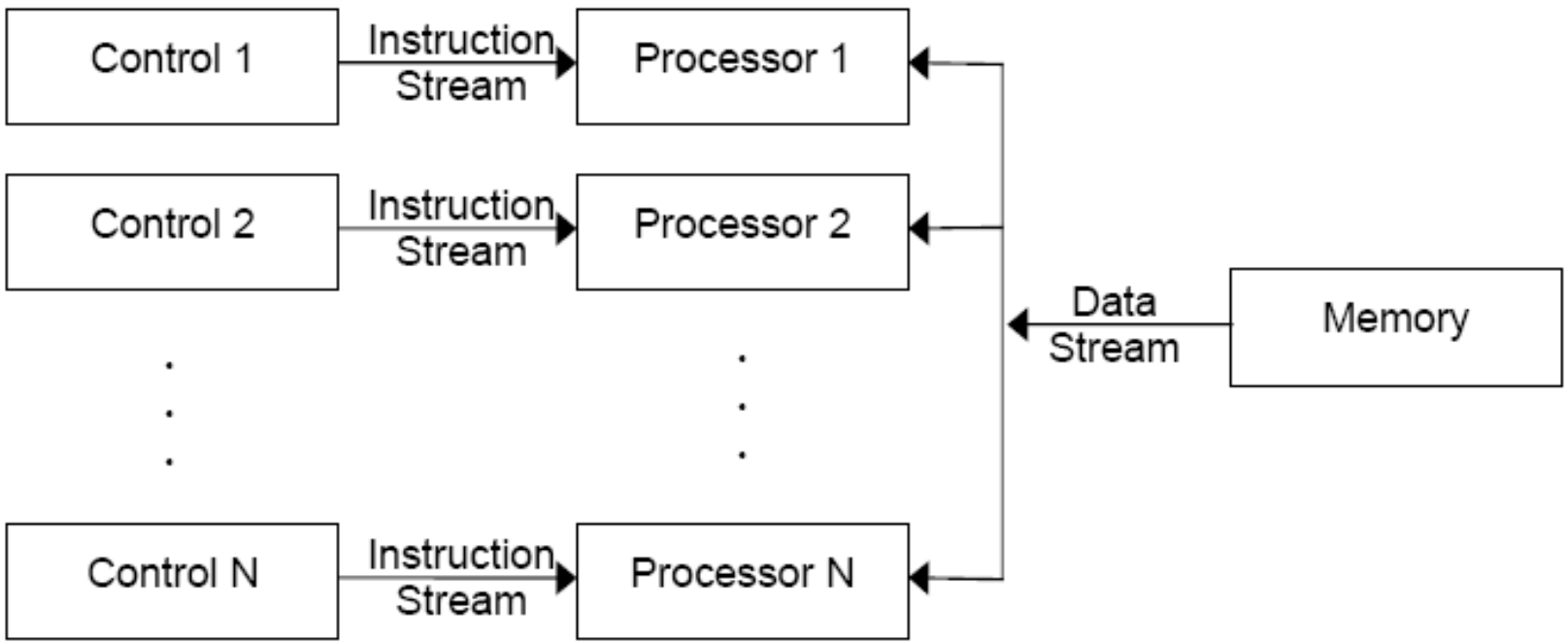
Sampai saat ini struktur ini masih merupakan struktur teoritis dan belum ada komputer dengan model ini.



[Akl 1989]

MISD (Multiple Instruction stream, Single Data stream)

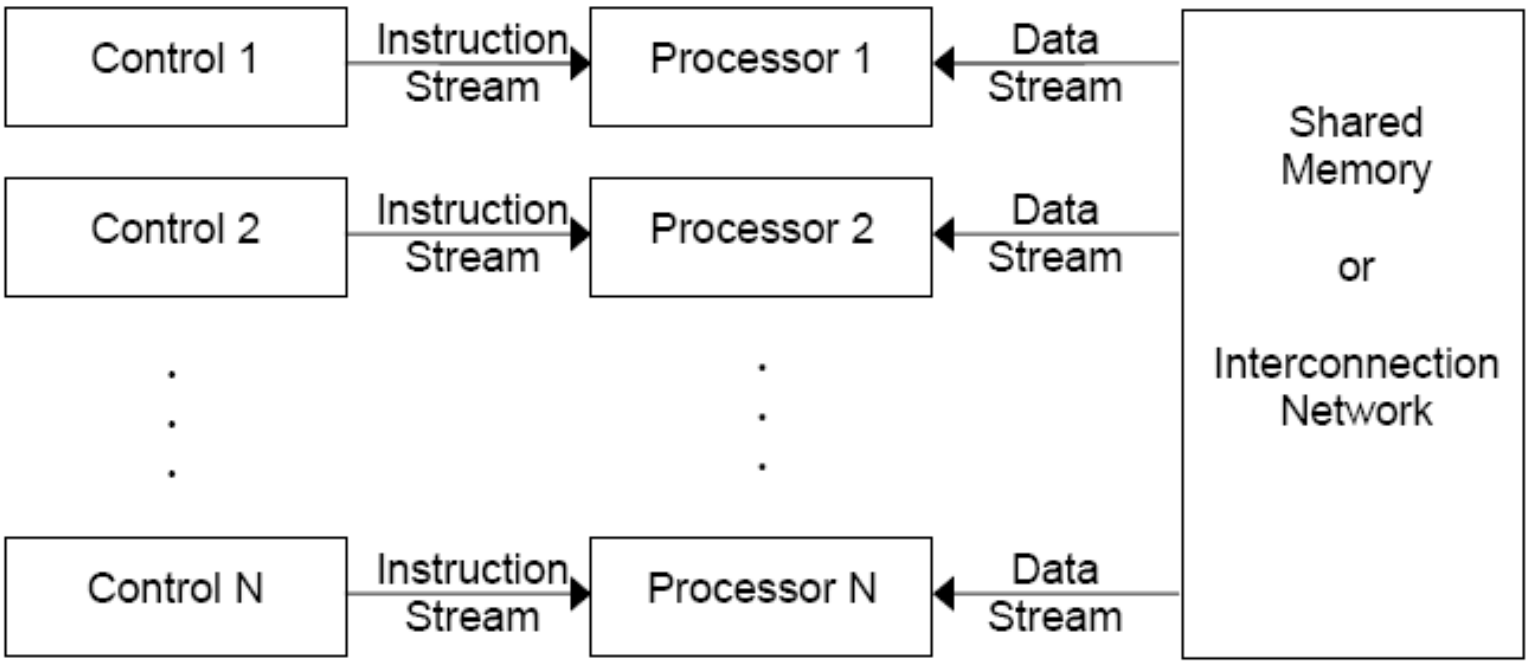
Sampai saat ini struktur ini masih merupakan struktur teoritis dan belum ada komputer dengan model ini.



[Akl 1989]

MIMD (Multiple Instruction stream, Multiple Data stream)

Organisasi komputer yang memiliki kemampuan untuk memproses beberapa program dalam waktu yang sama. Pada umumnya multiprosesor dan multikomputer termasuk dalam kategori ini.



[Akl 1989]

ALGORITMA SEQUENTIAL PENJUMLAHAN

Penjumlahan (SISD)

Begin

sum ← a_0

for i ← 1 to n-1 do

sum ← sum + a_i

endfor

end

ALGORITMA SEQUENTIAL PENJUMLAHAN

Misal Input : $a_i = \{1,2,3,4,5,6,7,8\}$

Sum = $a_0 = 1$

i = 1 sum = sum + a_1 = 1 + 2 = 3

i = 2 sum = sum + a_2 = 3 + 3 = 6

i = 1 sum = sum + a_3 = 6 + 4 = 10

i = 1 sum = sum + a_4 = 10 + 5 = 15

i = 1 sum = sum + a_5 = 15 + 6 = 21

i = 1 sum = sum + a_6 = 21 + 7 = 28

i = 1 sum = sum + a_7 = 28 + 8 = 36

Shared Memory & Interconnection Network

- In most interesting problems that we wish to solve on a SIMD computer, it is desirable for the processors to be able to communicate among themselves during the computation in order to exchange data or intermediate results.

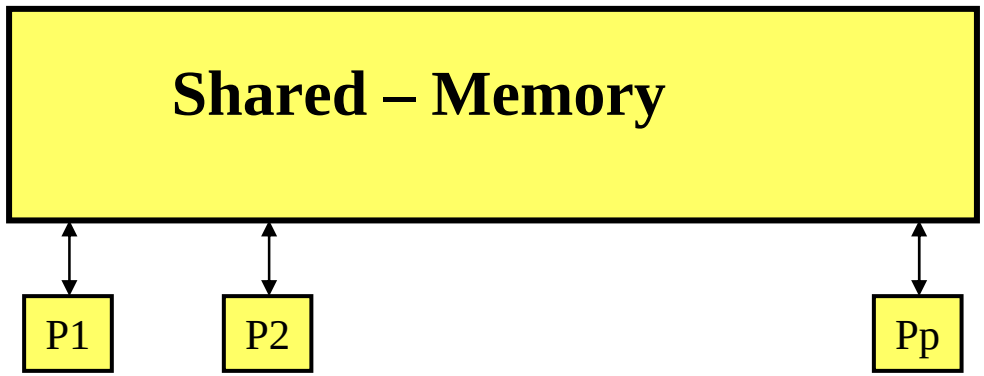
This can be achieved in two ways,

SIMD computers

- where communication is through *a shared memory*, and
- Where it is done via *an interconnection network*.

Parallel Randon Access Machine

- Extend the traditional RAM (Random Access Memory) machine



- Interconnection network between global memory and processors
- Multiple processors

Parallel Randon Access Machine

Characteristics

- Processors Pi (i ($0 \leq i \leq p-1$)
  - each with a local memory
  - i is a unique identity for processor P_i
- A global shared memory
  - it can be accessed by all processors

Parallel Randon Access Machine

Types of operations:

- Synchronous
  - Processors work in locked step
    - ☞at each step, a processor is active or idle
    - ☞suited for SIMD and MIMD architectures

- Asynchronous
  - processors have local clocks
  - needs to synchronize the processors
    - ☞ suited for MIMD architecture

Parallel Randon Access Machine

- Example of synchronous operation

Algorithm : Processor i (i=0 … 3)

Input : A, B
i processor id
Output : (1) C
Begin
If (B==0) C = A
Else C = A/B
End

Parallel Randon Access Machine

Initial

Processor 0	Processor 1	Processor 2	Processor 3
A : 5 B : 0 C : 0	A : 4 B : 2 C : 0	A : 2 B : 1 C : 0	A : 7 B : 0 C : 0

Step 1

Processor 0	Processor 1	Processor 2	Processor 3
A : 5 B : 0 C : 5	A : 4 B : 2 C : 0	A : 2 B : 1 C : 0	A : 7 B : 0 C : 7
(active B = 0)	**(idle B ≠ 0)**	**(idle B ≠ 0)**	**(active B = 0)**

Parallel Randon Access Machine

Step 2

Processor 0	Processor 1	Processor 2	Processor 3
A : 5 B : 0 C : 5	A : 4 B : 2 C : 2	A : 2 B : 1 C : 2	A : 7 B : 0 C : 7
(idle B = 0)	**(active B ≠ 0)**	**(active B ≠ 0)**	**(idle B = 0)**

Parallel Randon Access Machine

Read / Write conflicts

EREW : Exclusive - Read, Exclusive -Write

- no concurrent (read or write) operation on a variable

- CREW : Concurrent – Read, Exclusive – Write
  - concurrent reads allowed on same variable
  - exclusive write only

Parallel Randon Access Machine

- ERCW : Exclusive Read – Concurrent Write
- CRCW : Concurrent – Read, Concurrent – Write

Parallel Randon Access Machine

Concurrent write on a variable X

- Common CRCW : only if all processors write the same value on X
- SUM CRCW : write the sum all variables on X
- Random CRCW : choose one processor at random and write its value on X
- Priority CRCW : processor with hign priority writes on X

Parallel Randon Access Machine

Example:

Concurrent write on X by processors

P1 (50 → X) , P2 (60 → X), P3 (70 → X)

- Common CRCW ou ERCW : Failure

- SUM CRCW : X is the sum (180) of the written values
- Random CRCW : final value of

$$X \in \{ 50, 60, 70 \}$$

ALGORITMA PARALEL PENJUMLAHAN

PSEUDOCODE

SUM(EREW PRAM)

Initial condition : List of $n \geq 1$ elements stored in A[0 … (n - 1)]

Final condition : Sum of elements stored in A[0]

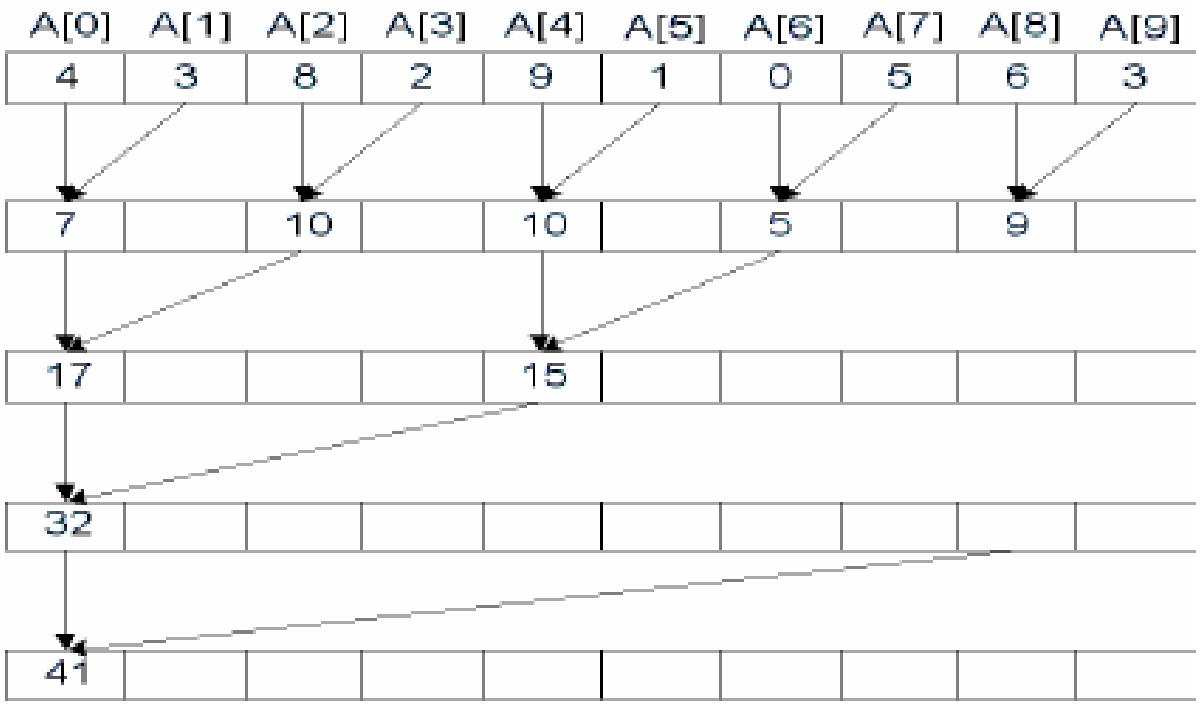
Global variables : n, A[0 … (n -1)], j

Begin

```
for all Pi where 0 ≤ i ≤ ⌊n/2⌋ –1 do
  for j ← 0 to ⌈log n⌉ - 1 do
      if i modulo 2^j = 0 and 2i + 2^j < n then
        A[2i] ← A[2i] + A[2i + 2^j]
      endif
  endfor
endfor
end
```

Gambar 3. Algoritma PRAM EREW untuk menjumlah n elemen dengan $\lfloor n/2 \rfloor$ prosesor

HASIL KOMPUTASI ALGORITMA PARALEL PENJUMLAHAN



Gambar 4. Meniumlahkan 10 nilai

ALGORITMA SEQUENTIAL PREFIX SUM

Input : A : array of integer
Output: pref_sum: array of integer

```
pref_sum[0] ← 0
begin
  for i ← 1 to n
     pref_sum[i] ← pref_sum[i-1] + A[i]
  endfor
end
```

ALGORITMA SEQUENTIAL PREFIX SUM

A[1,…,9] = {1,2,3,4,5,6,7,8,9}

pref_sum[1] = 0 + 1 = 1
pref_sum[2] = 1 + 2 = 3
pref_sum[3] = 3 + 3 = 6
pref_sum[4] = 6 + 4 = 10
pref_sum[5] = 10 + 5 = 15
pref_sum[6] = 15 + 6 = 21
pref_sum[7] = 21 + 7 = 28
pref_sum[8] = 28 + 8 = 36
pref_sum[9] = 36 + 9 = 45

ALGORITMA PARALEL PREFIX SUM

PSEUDOCODE

PREFIX.SUMS (CREW PRAM):

Initial condition : List of $n \geq 1$ elements stored in A[0 … (n -1)]

Final condition : Each element a[i] contains A[0] $\oplus$ … $\oplus$ A[i]

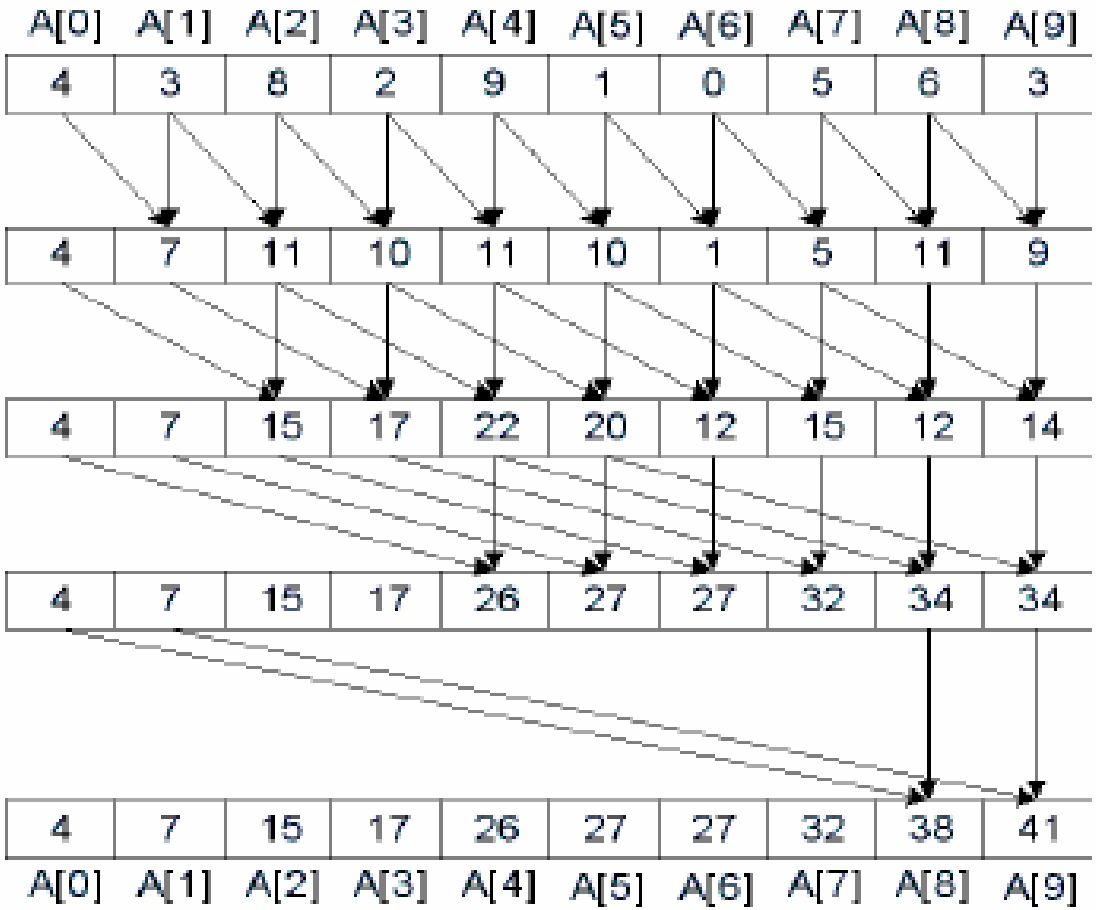
Global variables : n, A[0 … (n-1)], j

```
begin
        spawn (P1, P2, …, Pn - 1)
        for all Pi where 1 ≤ i ≤ n – 1 do
          for j ← 0 to ⌈log n⌉ – 1 do
                if i – 2^j ≥ 0 then
                        A[i] ← A[i] + A[i – 2^j]
                endif
          endfor
        endfor
end
```

Gambar 6. Algoritma PRAM untuk menemukan prefix sum dari n elemen dengan n-1 prosesor

ALGORITMA PARALEL PREFIX SUM

GAMBARAN PSEUDOCODE



Gambar 7. Algoritma Prefix sum dari 10 nilai

ALGORITMA PARALEL PREFIX SUM

```
Algorithm
Input
        int A[n] ; /* An array of n elements in global memory */
Output
        int  A;
Variables locales
        int k, a, b, c ;

for all I (i=0 …, n-1) do in parallel begin
     for k =  0 to ⌊ log (n – 1) ⌋  do
                    if ( i ≥ 2^k )  then  begin
                                global read ( a, A [ i  - 2^k] )
                                global read (b, A[ i ])
                                c = a + b
                                global write (c, A[i])
                    end
end
```

The following table is an illustration of computation of above algorithm which computing the prefix sums of the values of A = { 1,2,3,4,5,6 }

	P_0	P_1	P_2	P_3	P_4	P_5
k \ i	0	1	2	3	4	5
k=0	$0 < 2^0$	$1 = 2^0$ a=A[0]=1 b=A[1]=2 c=3	$2 > 2^0$ a=A[1]=2 b=A[2]=3 c=5	$3 > 2^0$ a=A[2]=3 b=A[3]=4 c=7	$4 > 2^0$ a=A[3]=4 b=A[4]=5 c=9	$5 > 2^0$ a=A[4]=5 b=A[5]=6 c=11
	A[0]=1	A[1]=3	A[2]=5	A[3]=7	A[4]=9	A[5]=11
k=1	$0 < 2^1$	$1 < 2^1$	$2 = 2^1$ a=A[0]=1 b=A[2]=5 c=3	$3 > 2^1$ a=A[1]=3 b=A[3]=7 c=10	$4 > 2^1$ a=A[2]=5 b=A[4]=9 c=14	$5 > 2^1$ a=A[3]=7 b=A[5]=11 c=18
	A[0]=1	A[1]=3	A[2]=6	A[3]=10	A[4]=14	A[5]=18
k=2	$0 < 2^2$	$1 < 2^2$	$2 < 2^2$	$3 < 2^2$	$4 = 2^2$ a=A[0]=1 b=A[4]=14 c=15	$5 > 2^2$ a=A[1]=3 b=A[5]=18 c=18
	A[0]=1	A[1]=3	A[2]=6	A[3]=10	A[4]=15	A[5]=21