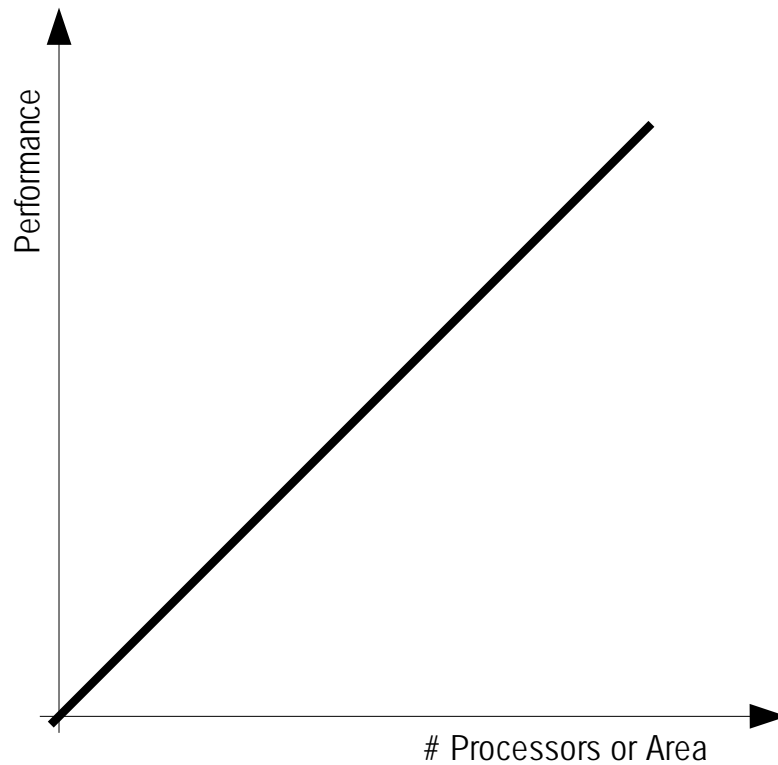




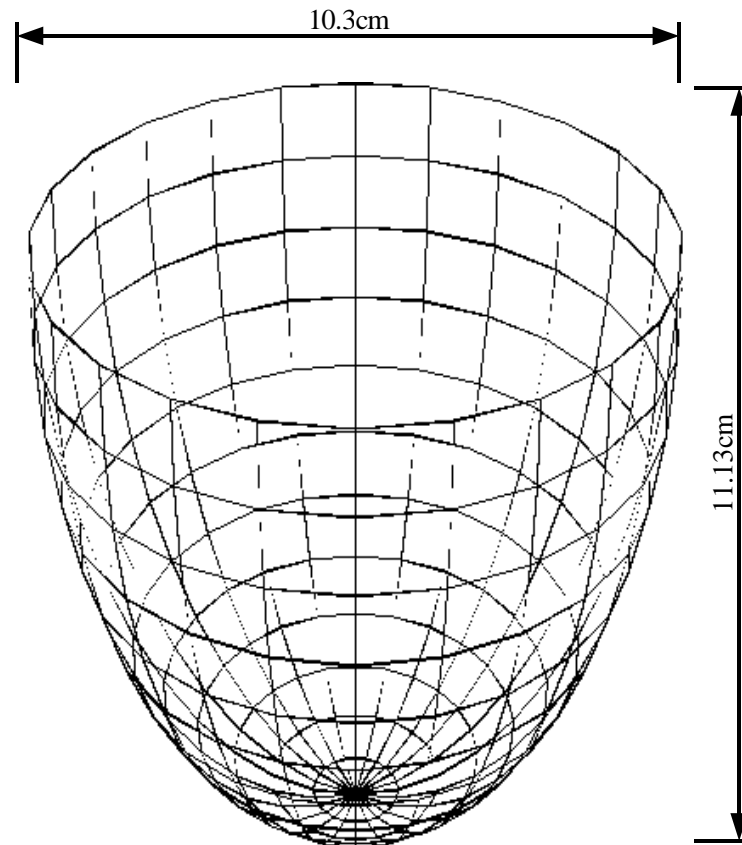
Is there a parallel Universe?

Will Robbins

Potential gains from parallel processing

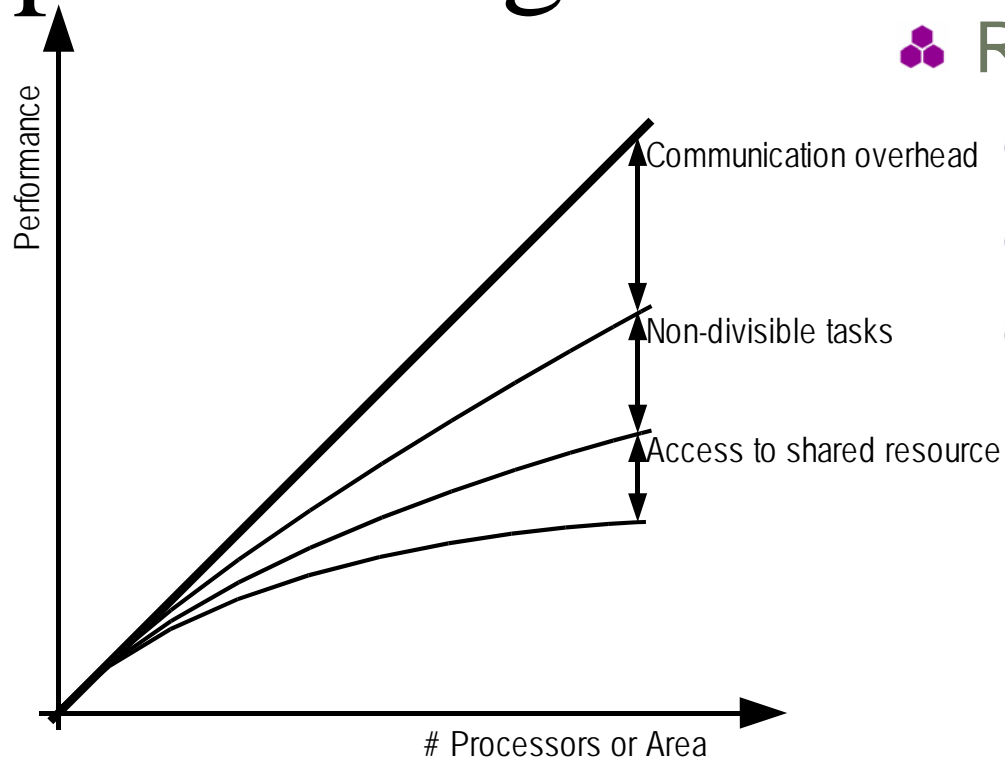


Plan of hole



- ❖ Reducing gains of parallel effort
 - ❖ Non-divisible tasks
 - ❖ Communication overhead
 - ❖ Shared resource

Reduced gains from parallel processing



Reducing gains of parallel effort

- Non-divisible tasks
- Communication overhead
- Shared resource

Flies in the soup

- ❖ Non-divisible tasks
- ❖ Communication overhead
- ❖ Shared resource

Flies in the soup – digging a hole

- ❖ Non-divisible tasks - its too small a problem to divide
- ❖ Communication overhead - discuss who has done what
- ❖ Shared resource - looking at the plan

Flies in the soup – software project

- ❖ Non-divisible tasks – very divisible
- ❖ Communication overhead – specification, meetings
- ❖ Shared resource – no problem

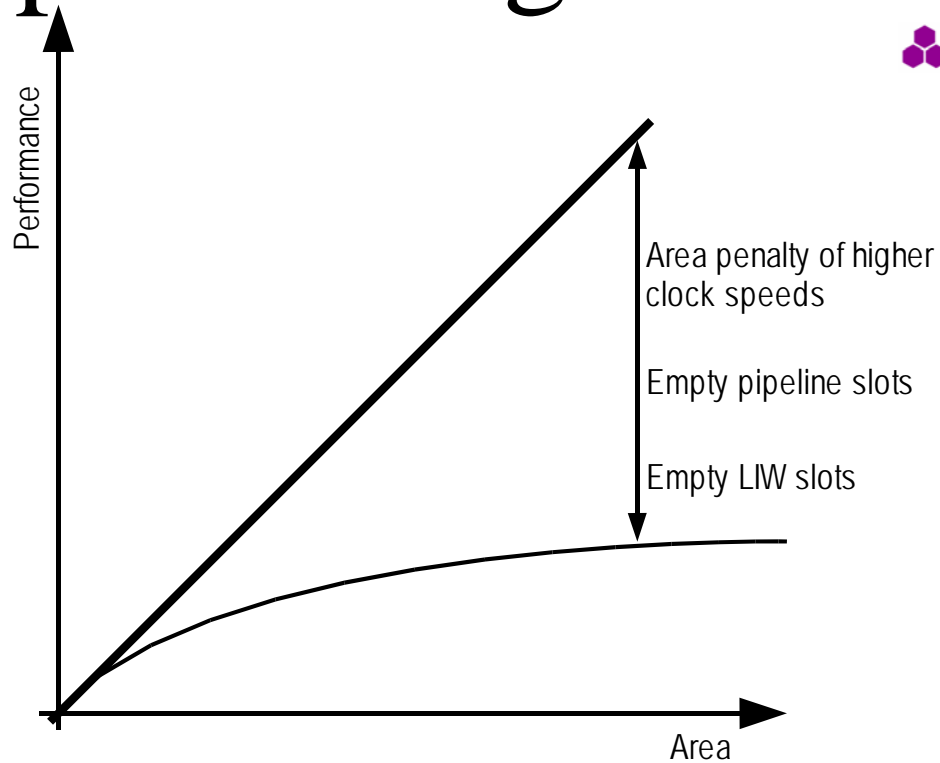
Flies in the soup – file system

- ❖ Non-divisible tasks – good – very divisible
- ❖ Communication overhead – good – separate tasks
- ❖ Shared resource – bad – the file system is shared

There is NO Parallel Universe

- ❖ Gains are application dependent
- ❖ Parallel processing is not universally applicable

Reduced gains from serial processing

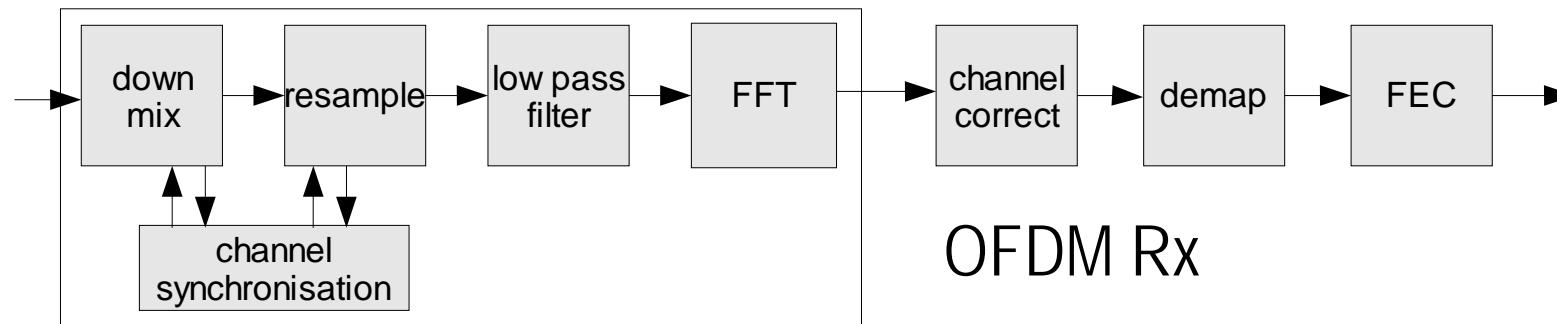


❖ Reducing gains of serial tricks

- ❖ Higher clock speeds
- ❖ Deeper pipelines
- ❖ Wider LIW

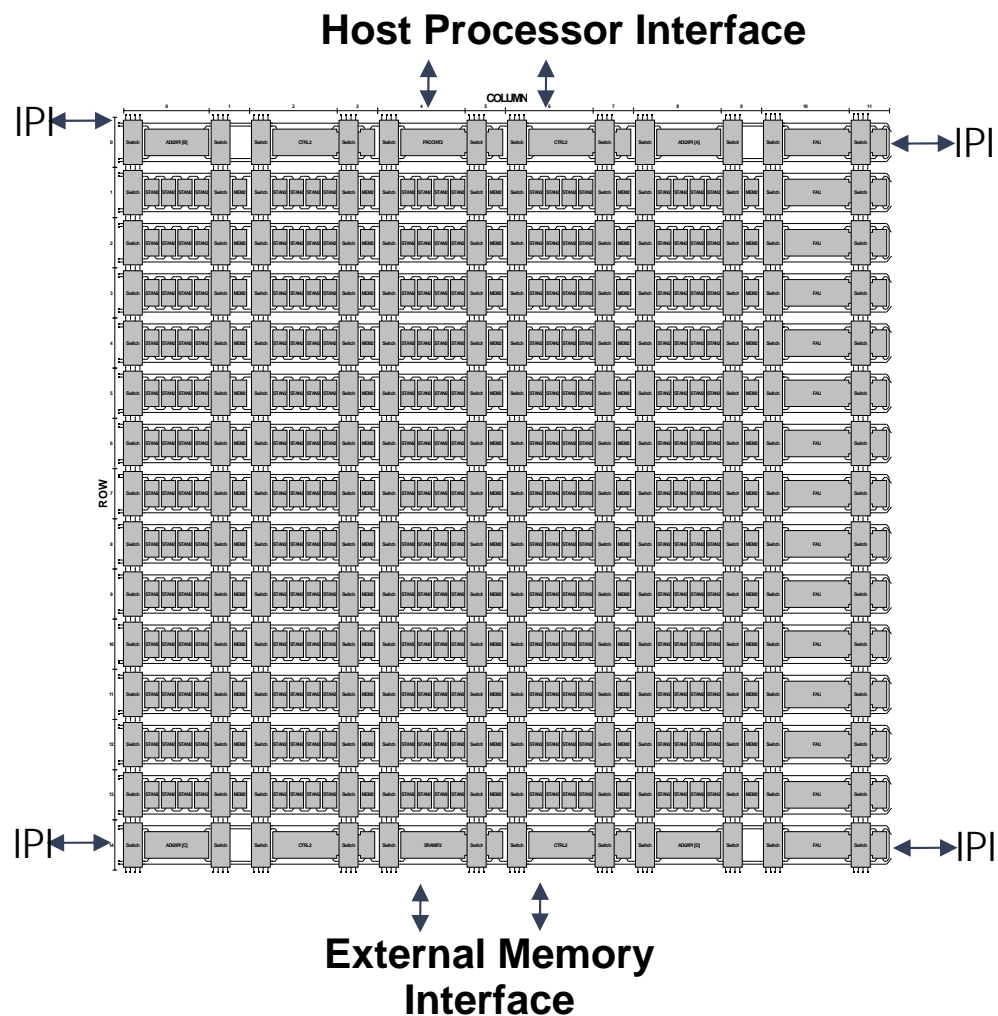
DSP/flow based problems

- ❁ Non-divisible tasks - good - very divisible
- ❁ Communication overhead - an issue of massively parallel
- ❁ Shared resource - good - data passes down the stream



- ❁ Parallelism obvious
- ❁ Data flows means little shared memory

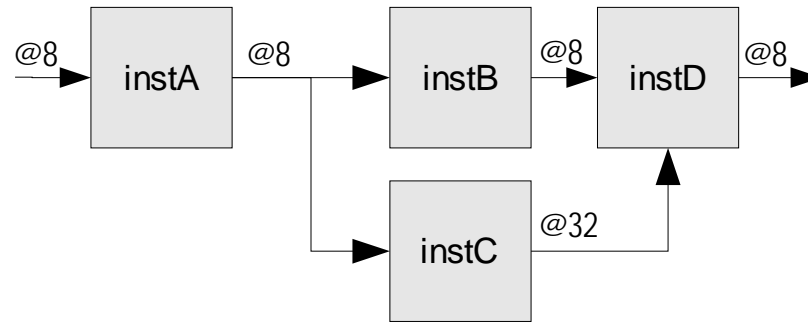
picoArray PC102



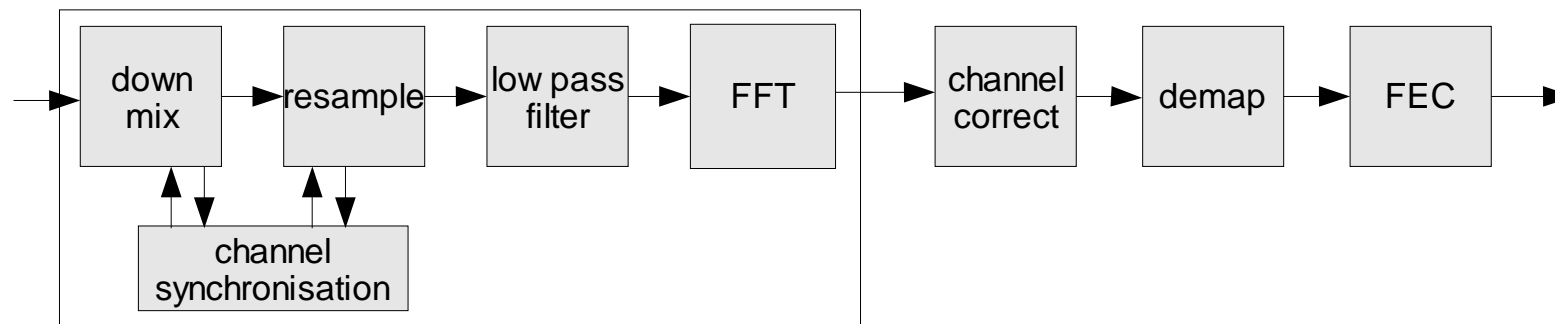
- Instantiation of the picoArray
- Peak performance
 - 197 GIPS
 - 38.6 GMACs
 - 3.3Tbps communications bandwidth
- 308 processors
- 14 co-processor accelerators for FEC, correlators

Programming model

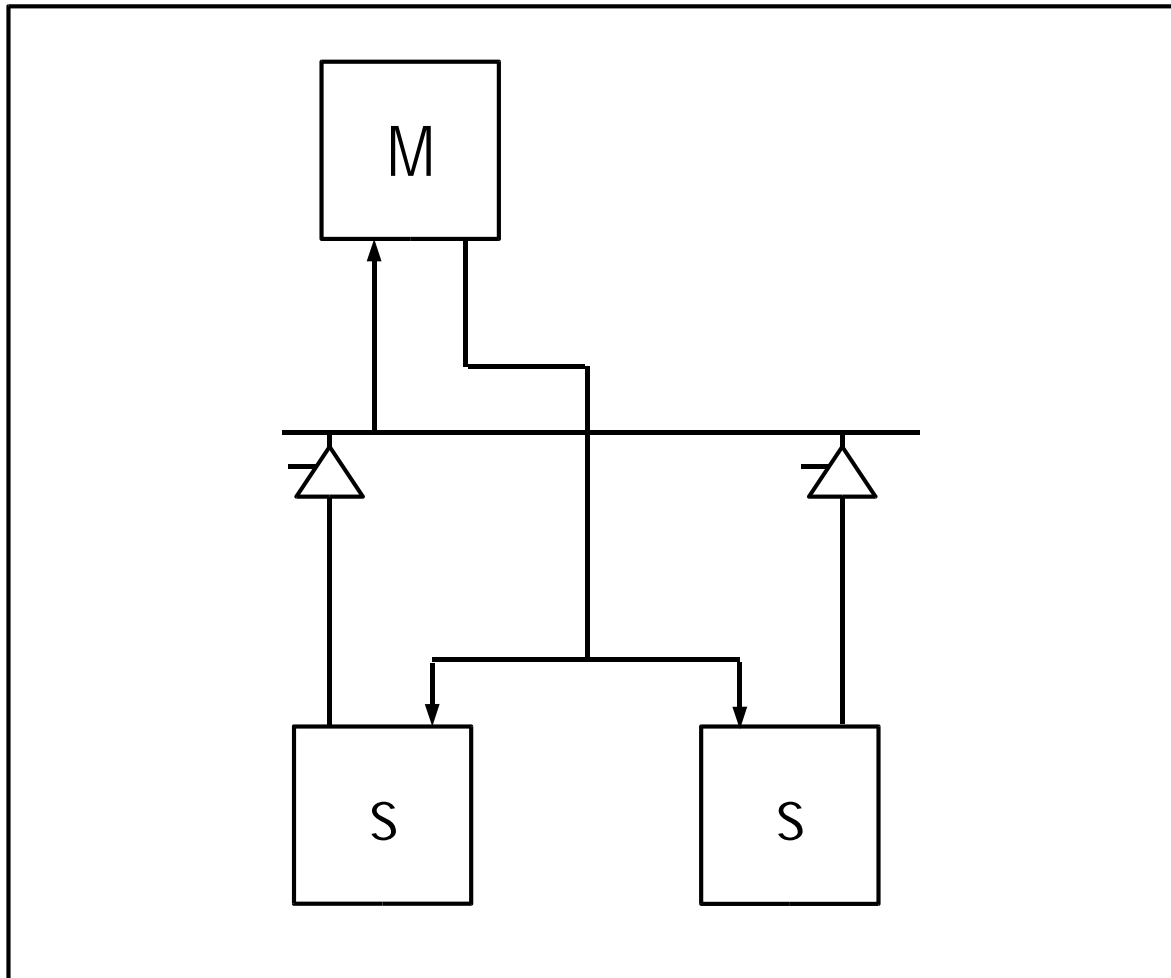
- Define fixed interconnect and specified bandwidth between processes
- Define processes in C or Assembler



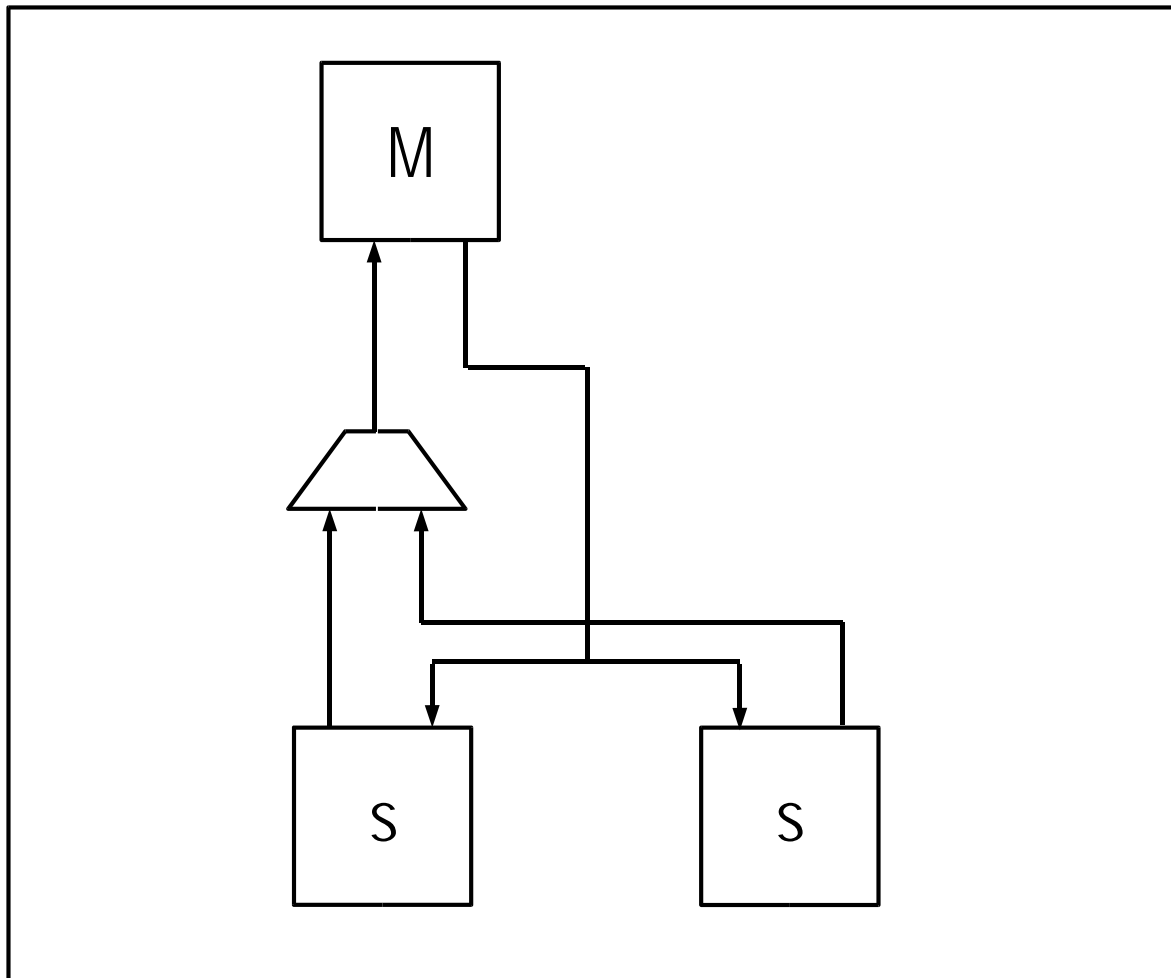
- Maps very naturally to DSP systems



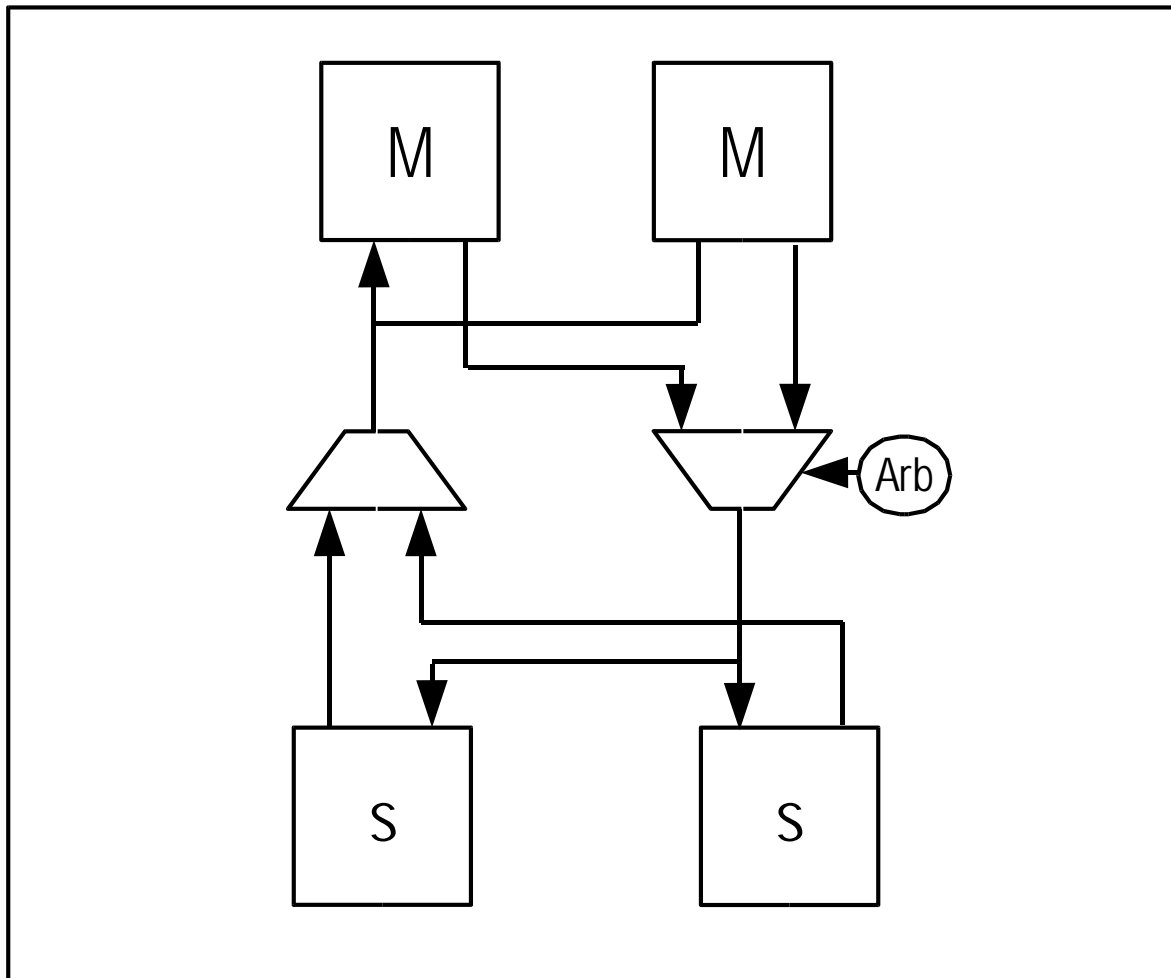
Bus structure - Tristate



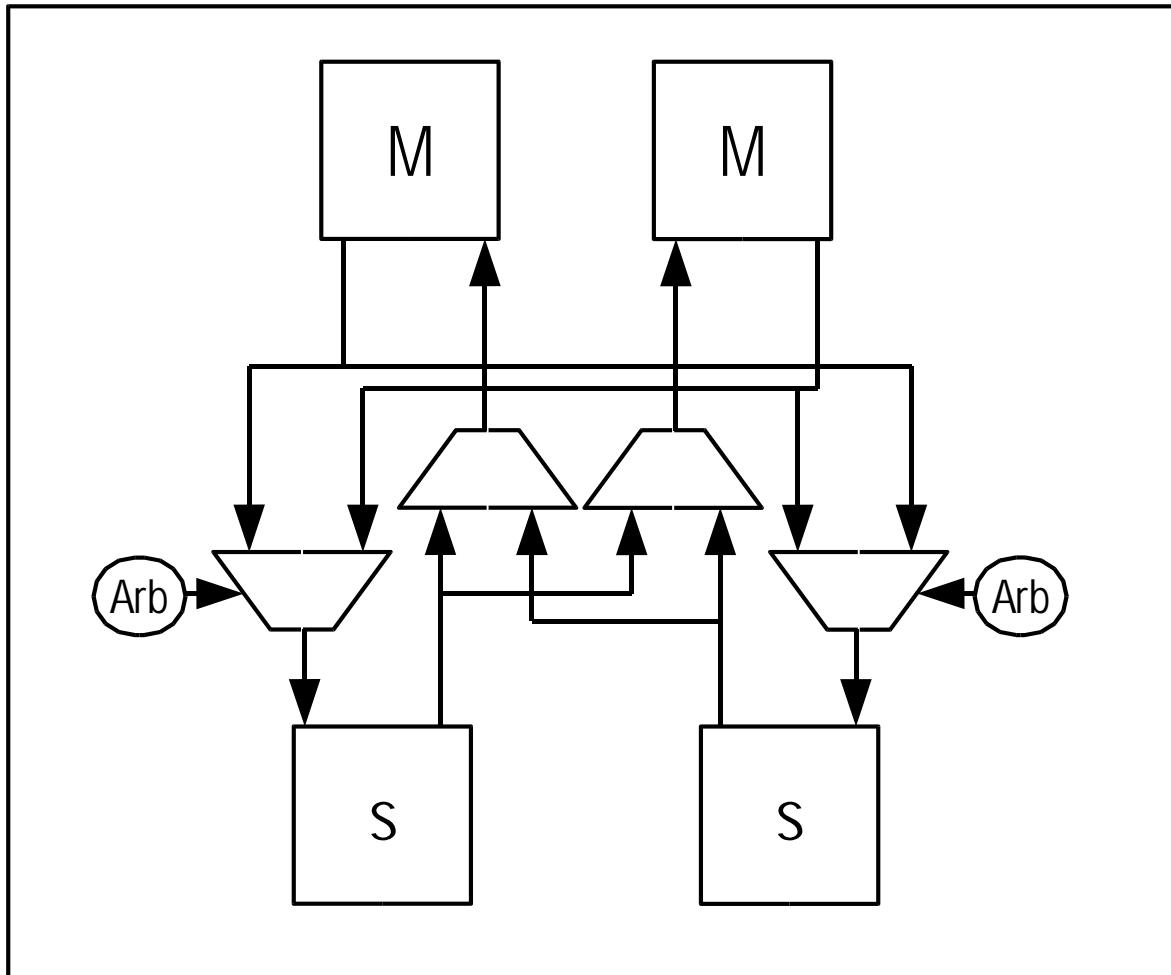
Bus structures – multiplexed bus



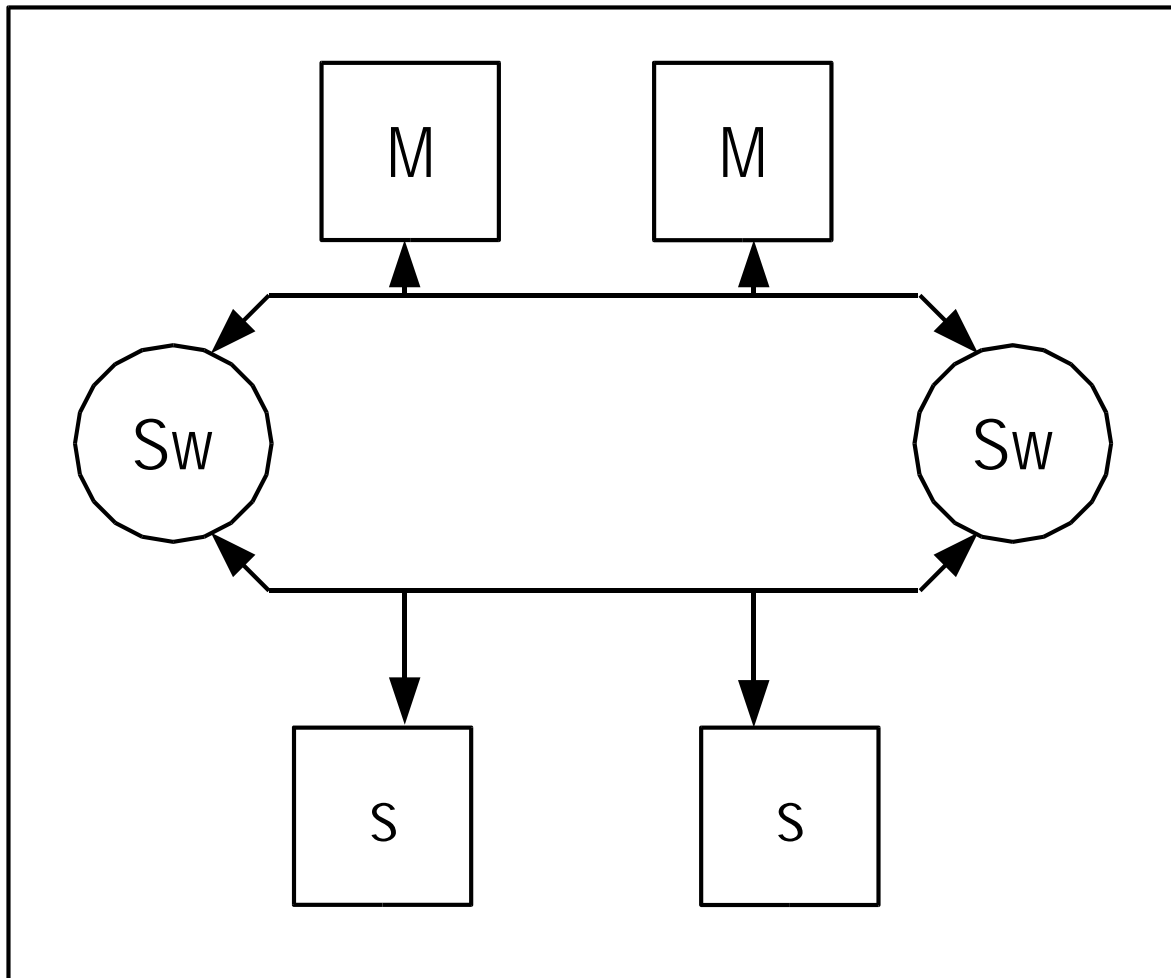
Bus structures – multi-master, single bus



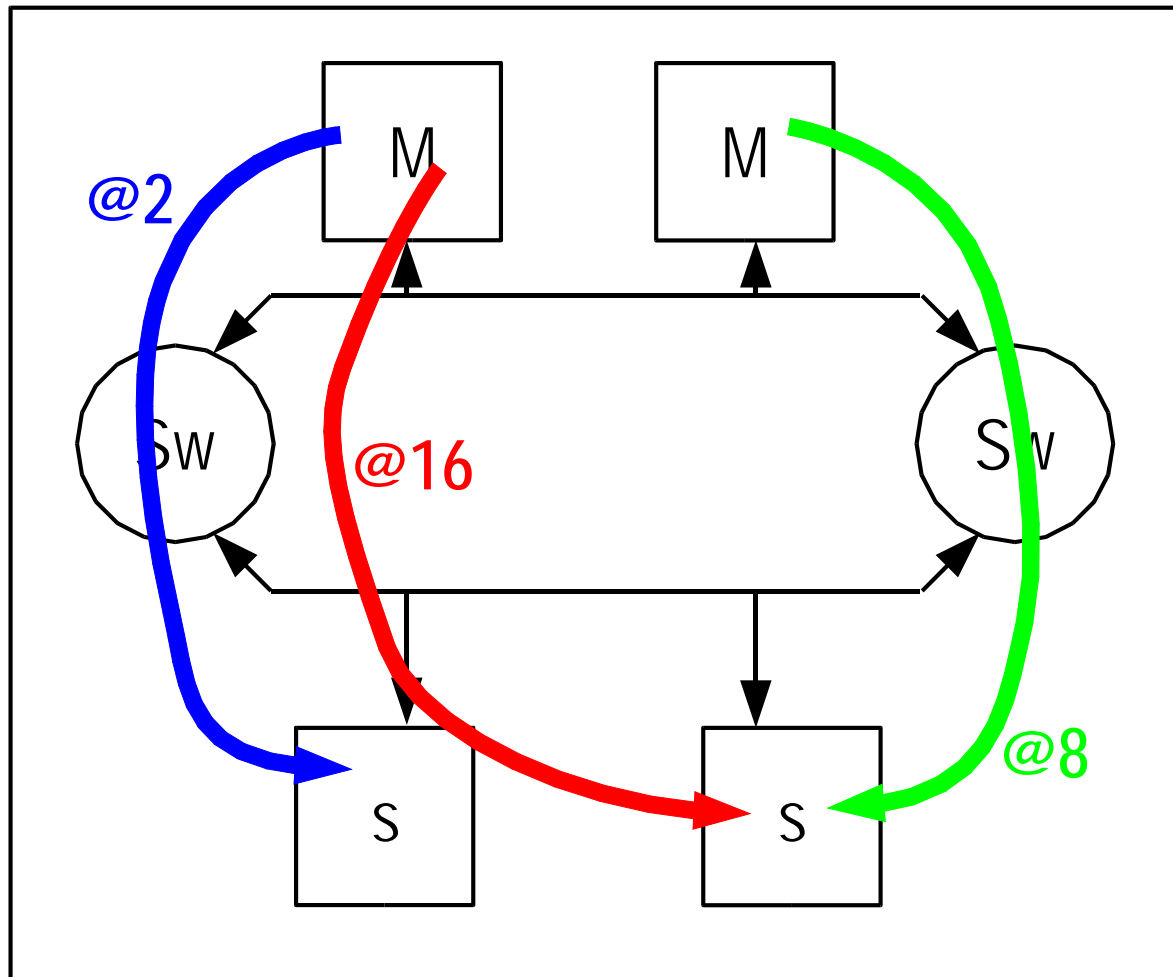
Bus structures – multi-master, multi-bus



picoBus – statically defined TDM bus

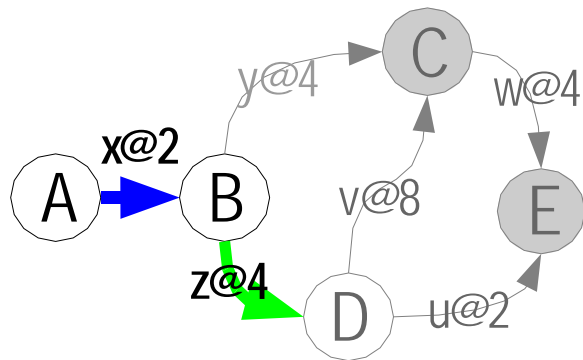


picoBus – statically defined TDM bus

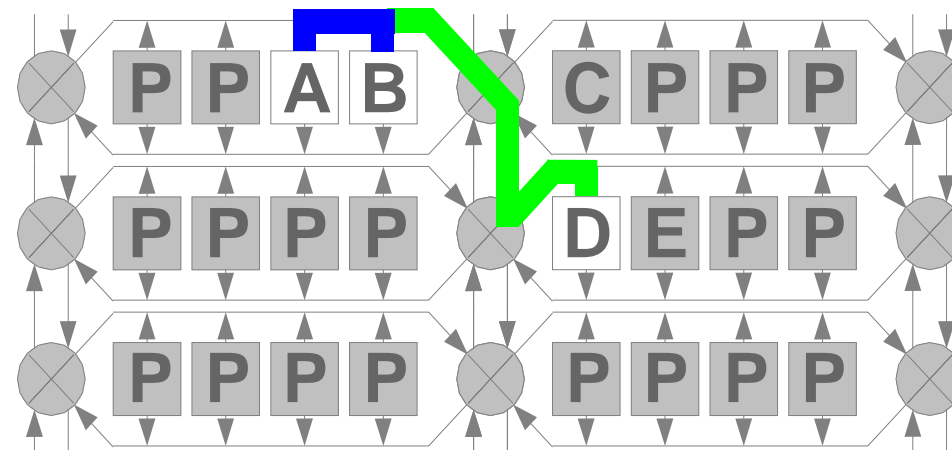


picoBus – statically defined TDM bus

Programmers model



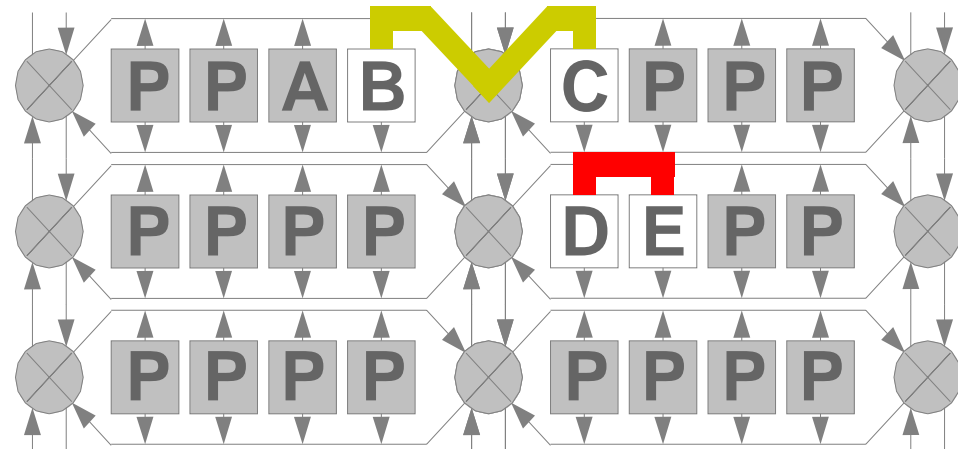
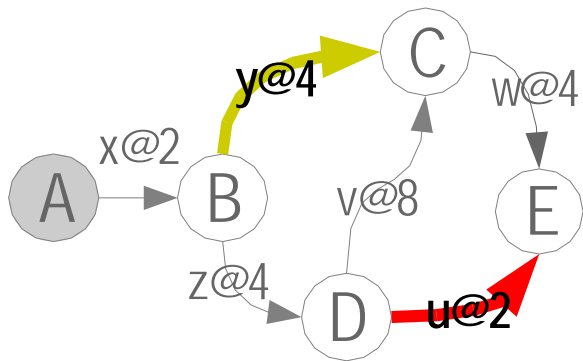
portion of picoArray



	0	1	2	3	4	5	6	7
U		Red		Red		Red		Red
V								Cyan
W			Purple				Purple	
X	Blue		Blue		Blue		Blue	
Y		Yellow				Yellow		
Z	Green				Green			

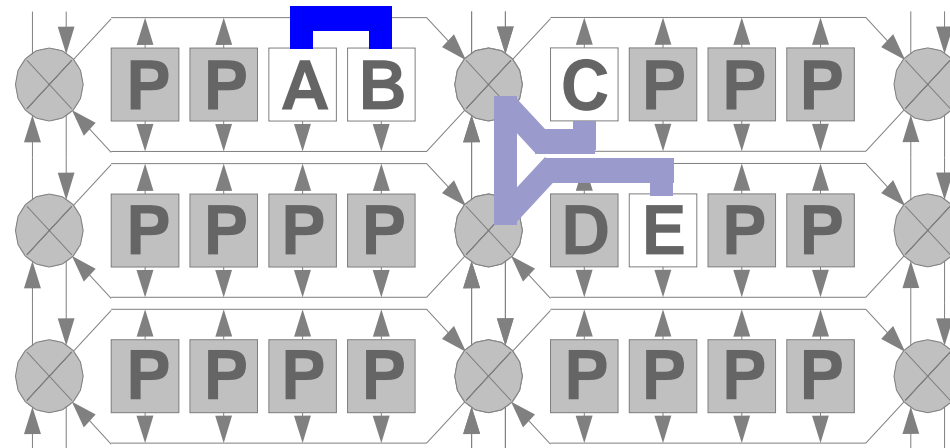
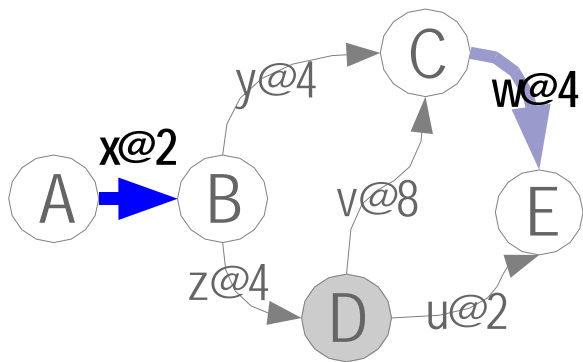
Signal activity

picoBus – statically defined TDM bus



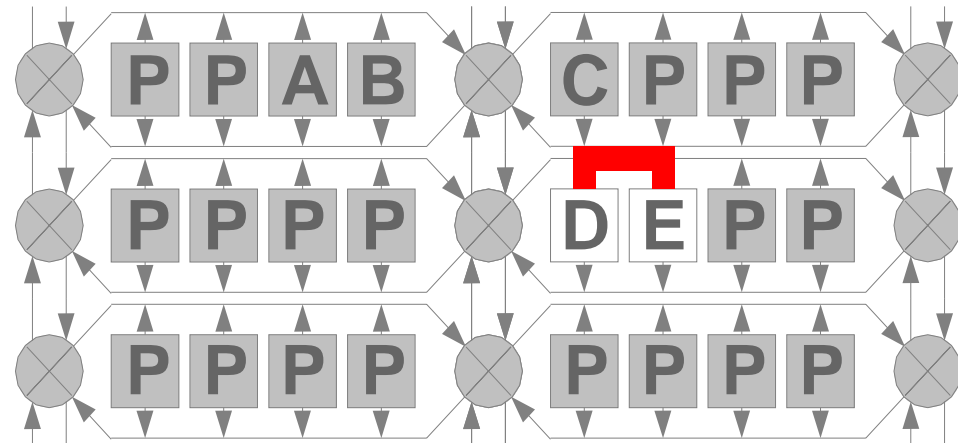
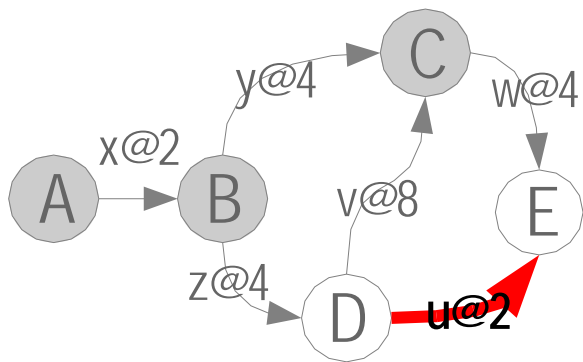
	0	1	2	3	4	5	6	7
U		Red		Red		Red		Red
V		Grey						Cyan
W		Grey	Purple				Purple	
X	Blue	Grey	Blue		Blue		Blue	
Y		Yellow				Yellow		
Z	Green	Grey			Green			

picoBus – statically defined TDM bus



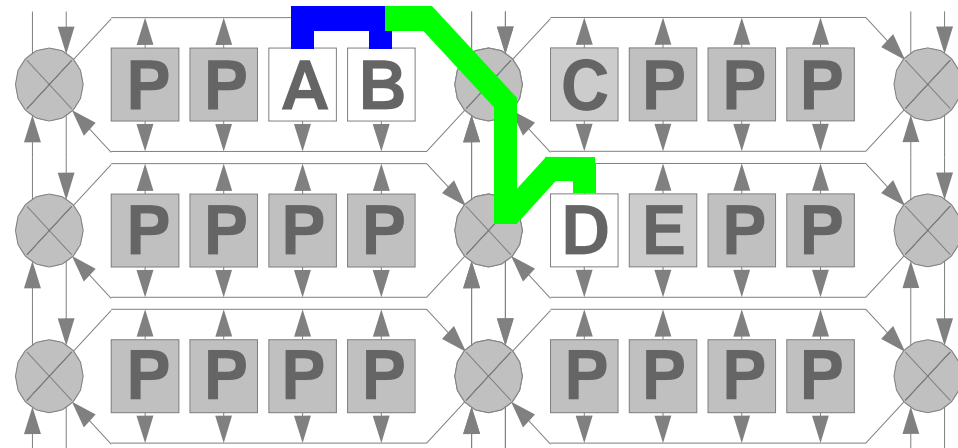
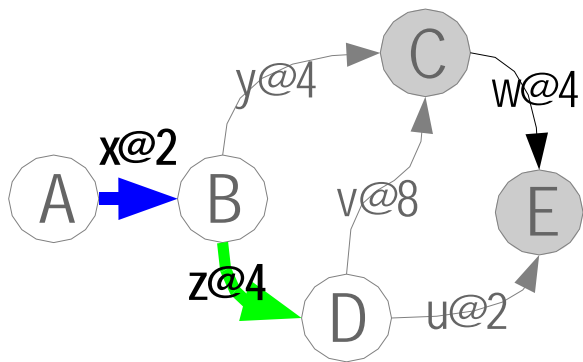
	0	1	2	3	4	5	6	7
U		Red	Grey	Red		Red		Red
V			Grey					Cyan
W			Purple				Purple	
X	Blue		Blue		Blue		Blue	
Y		Yellow	Grey			Yellow		
Z	Green		Grey		Green			

picoBus – statically defined TDM bus



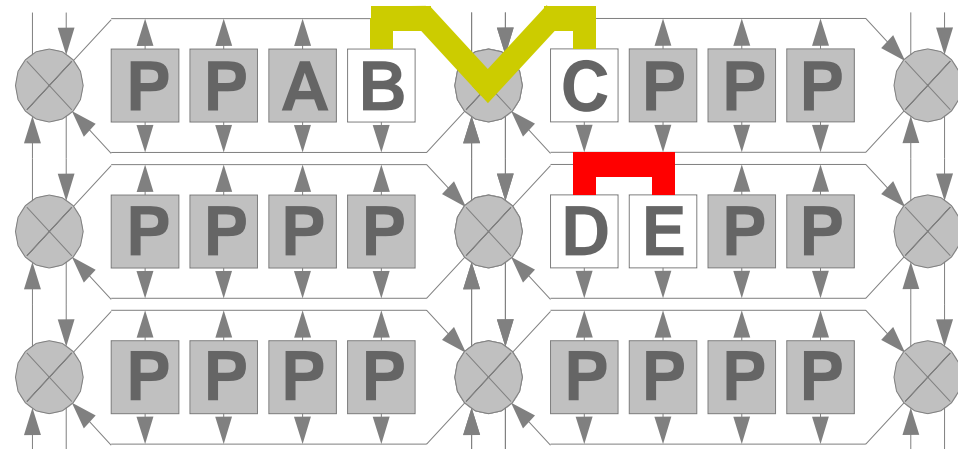
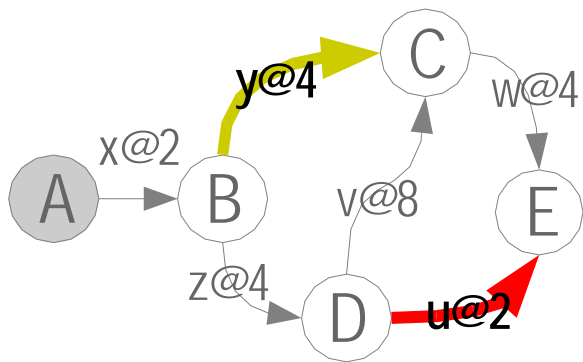
	0	1	2	3	4	5	6	7
U								
V								
W								
X								
Y								
Z								

picoBus – statically defined TDM bus



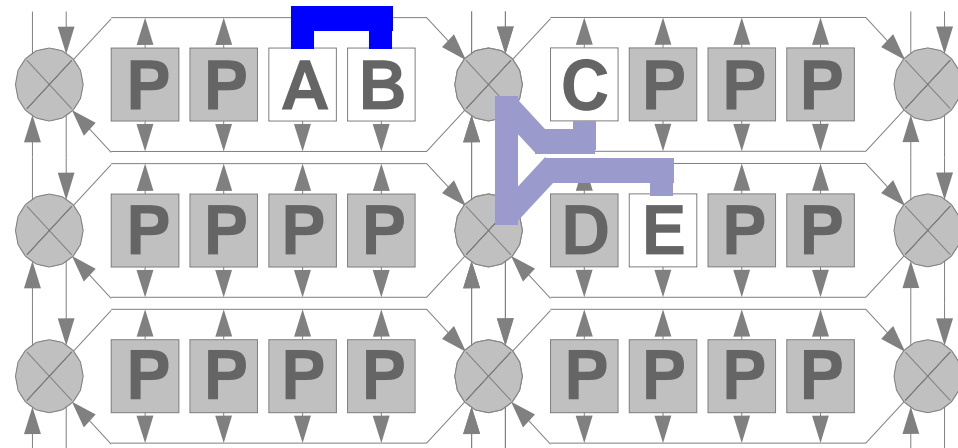
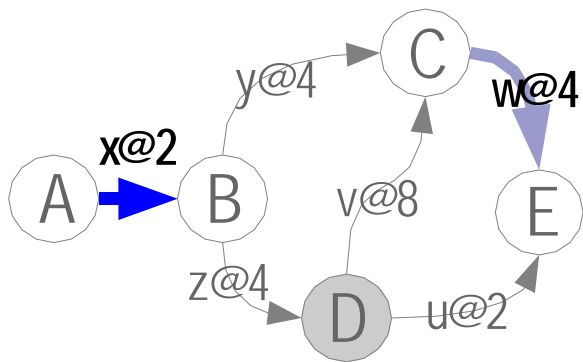
	0	1	2	3	4	5	6	7
U		Red		Red	Grey	Red		Red
V					Grey			Cyan
W			Purple		Grey		Purple	
X	Blue		Blue		Blue		Blue	
Y		Yellow			Grey	Yellow		
Z	Green				Green			

picoBus – statically defined TDM bus



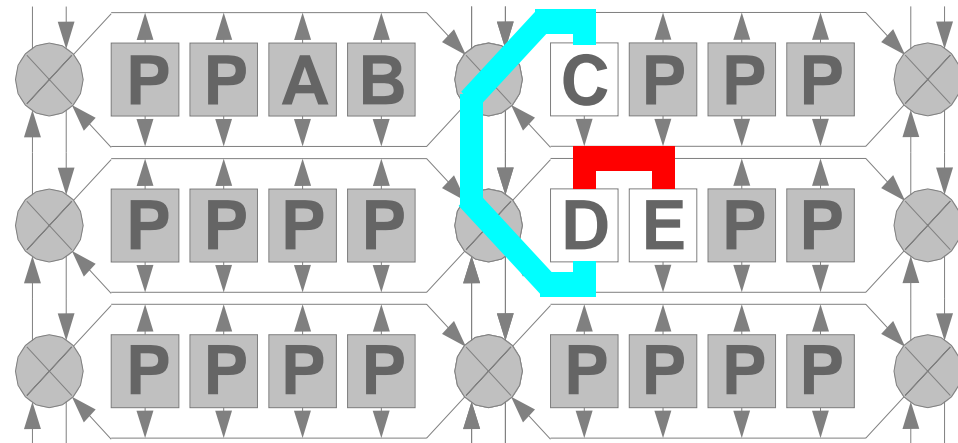
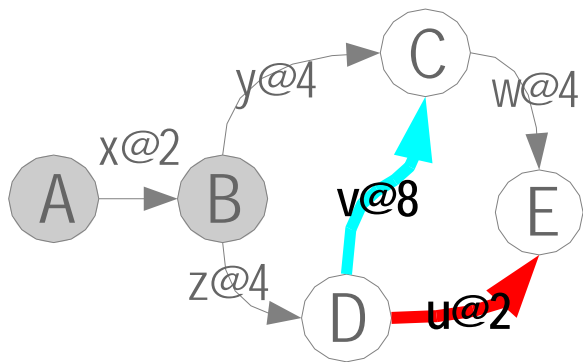
	0	1	2	3	4	5	6	7
U		Red		Red		Red		Red
V						Grey		Cyan
W			Purple			Grey	Purple	
X	Blue		Blue		Blue	Grey	Blue	
Y		Yellow				Yellow		
Z	Green				Green	Grey		

picoBus – statically defined TDM bus



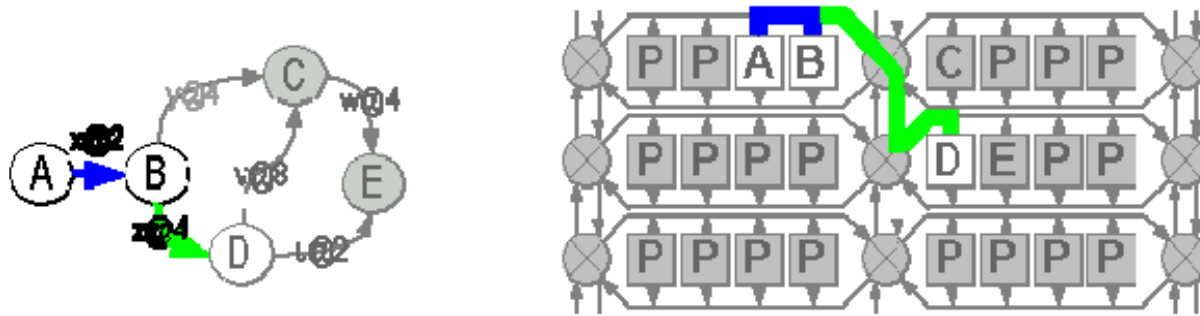
	0	1	2	3	4	5	6	7
U								
V								
W								
X								
Y								
Z								

picoBus – statically defined TDM bus



	0	1	2	3	4	5	6	7
U		Red		Red		Red		Red
V								Cyan
W			Purple				Purple	
X	Blue		Blue		Blue		Blue	
Y		Yellow				Yellow		
Z	Green				Green			

picoBus – statically defined TDM bus



	0	1	2	3	4	5	6	7
U		Red		Red		Red		Red
V								Cyan
W			Purple				Purple	
X	Blue		Blue		Blue		Blue	
Y		Yellow				Yellow		
Z	Green				Green			

picoArray code example

```

entity NCO is
    generic (FREQ_BASE : integer16);
    port (oscillator:out blocking integer16pair@16);
end entity NCO;

architecture C of NCO is
begin ANY
CODE
int main() {
    integer16pair out_phase;
    int sine_phase, sine;

    while (1) {
        if (quadrant & 1)
            sine_phase = ((1 << 6) - 1) - sine_phase ;
            sine = sine_lookup[sine_phase];

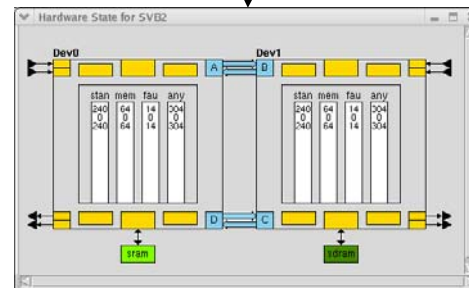
        // invert sine?
        if (quadrant & 2) sine = -sine;
        out_phase.e11 = 0;
        out_phase.e12 = sine;
        putoscillator(out_phase);
    }
}
ENDCODE
end NCO
    
```

Specify process IO

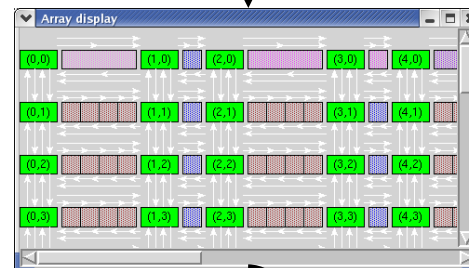
Specify process function in C

compiler/assembler

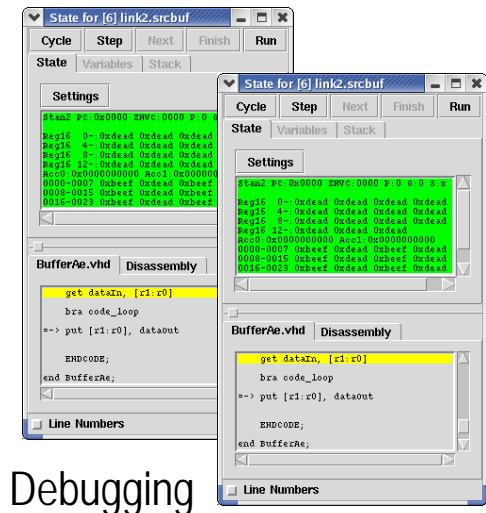
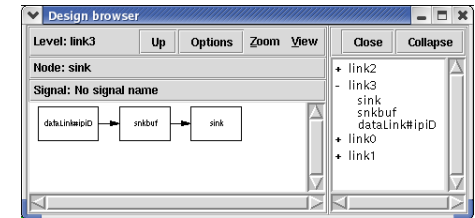
Compiling



Partitioning



Automatic placement & routing



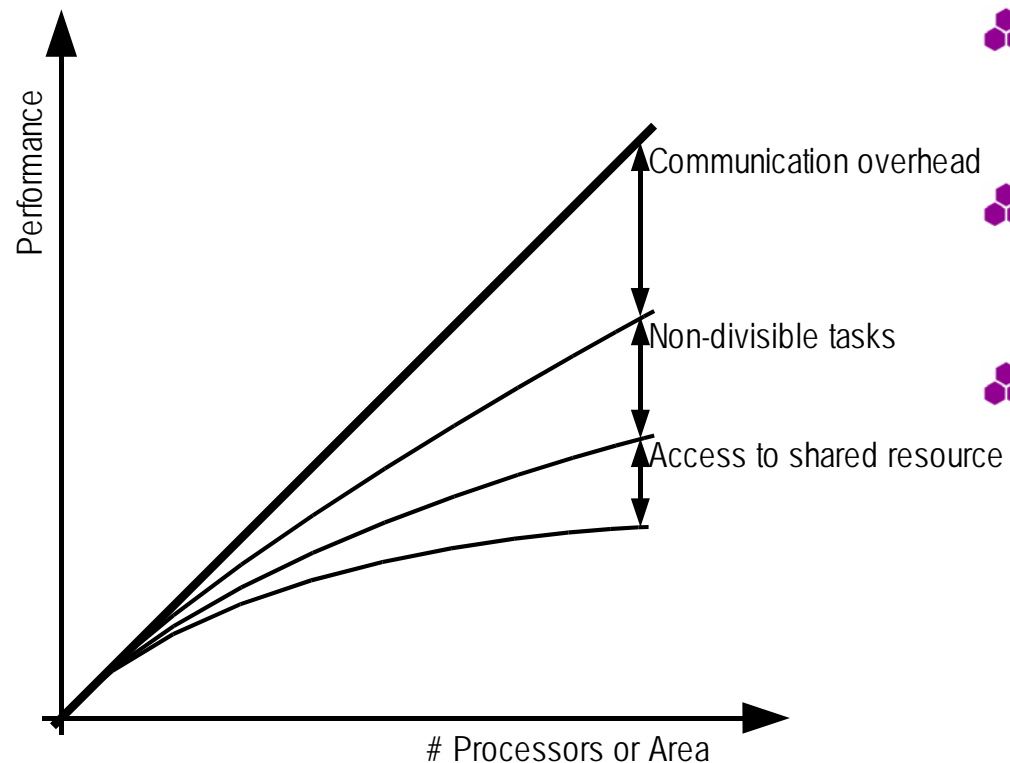
Debugging



Parallel debugger run in simulation or on picoArrays

Tools chain functionality

Gains from parallel processing

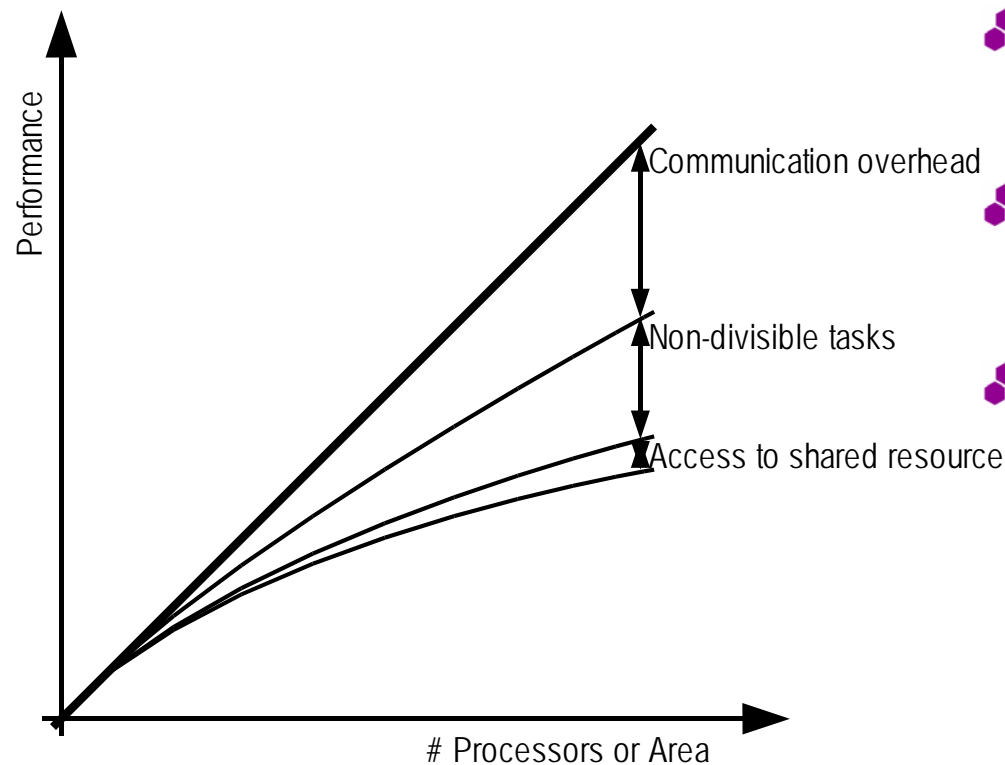


Access to shared resource

Non-divisible tasks

Communication overhead

Gains from parallel processing



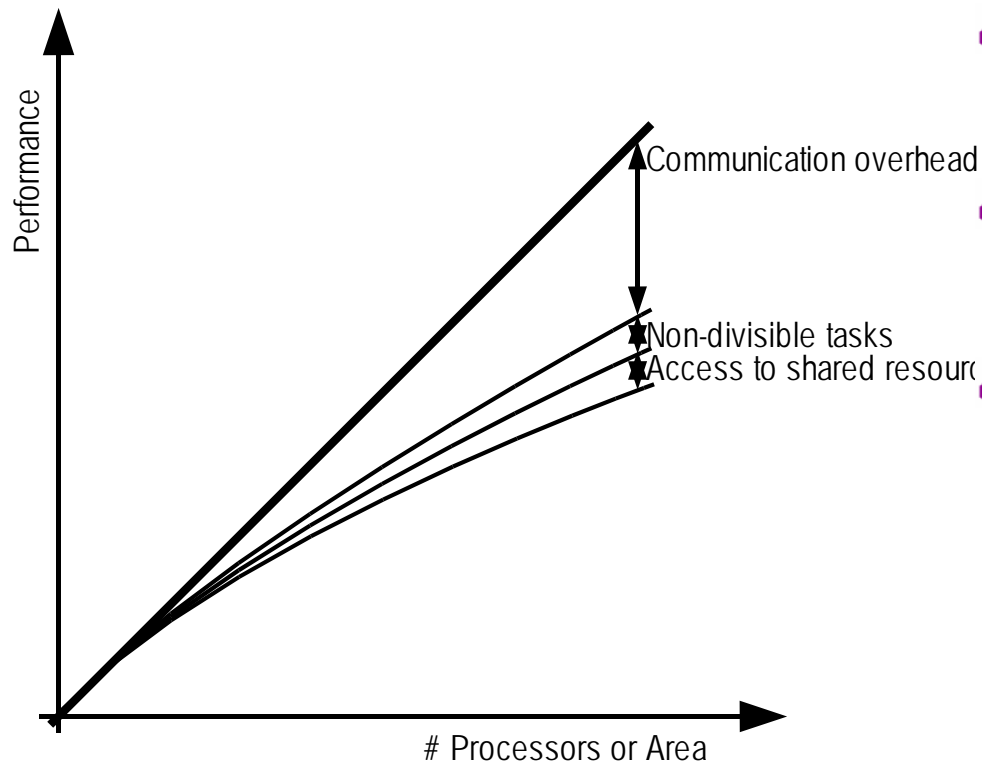
⬢ Access to shared resource

⬢ Data passed along the flow

⬢ Non-divisible tasks

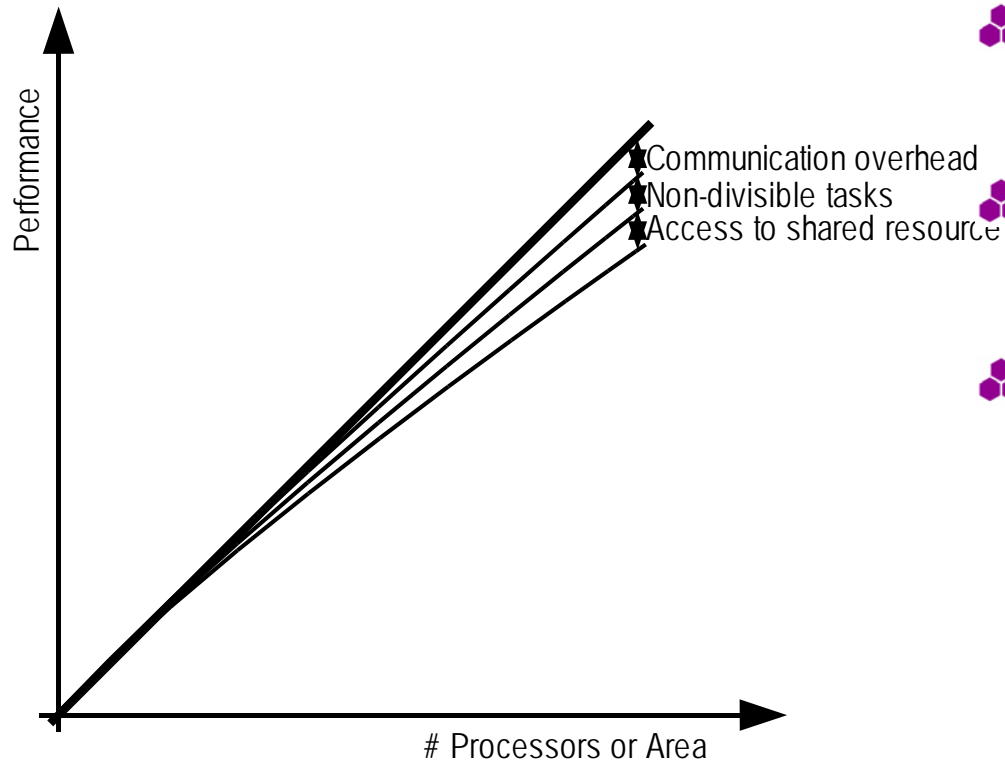
⬢ Communication overhead

Gains from parallel processing



- ❖ Access to shared resource
 - ❖ Data passed along the flow
- ❖ Non-divisible tasks
 - ❖ Inherently parallel
- ❖ Communication overhead

Gains from parallel processing



Access to shared resource

- Data passed along the flow

Non-divisible tasks

- Inherently parallel

Communication overhead

- picoBus allows 100's processors to communicate

It is real

It exists now

- You can buy the tool chain
- You can buy the chips
- You can buy the reference designs
- You can buy the base stations

Summary

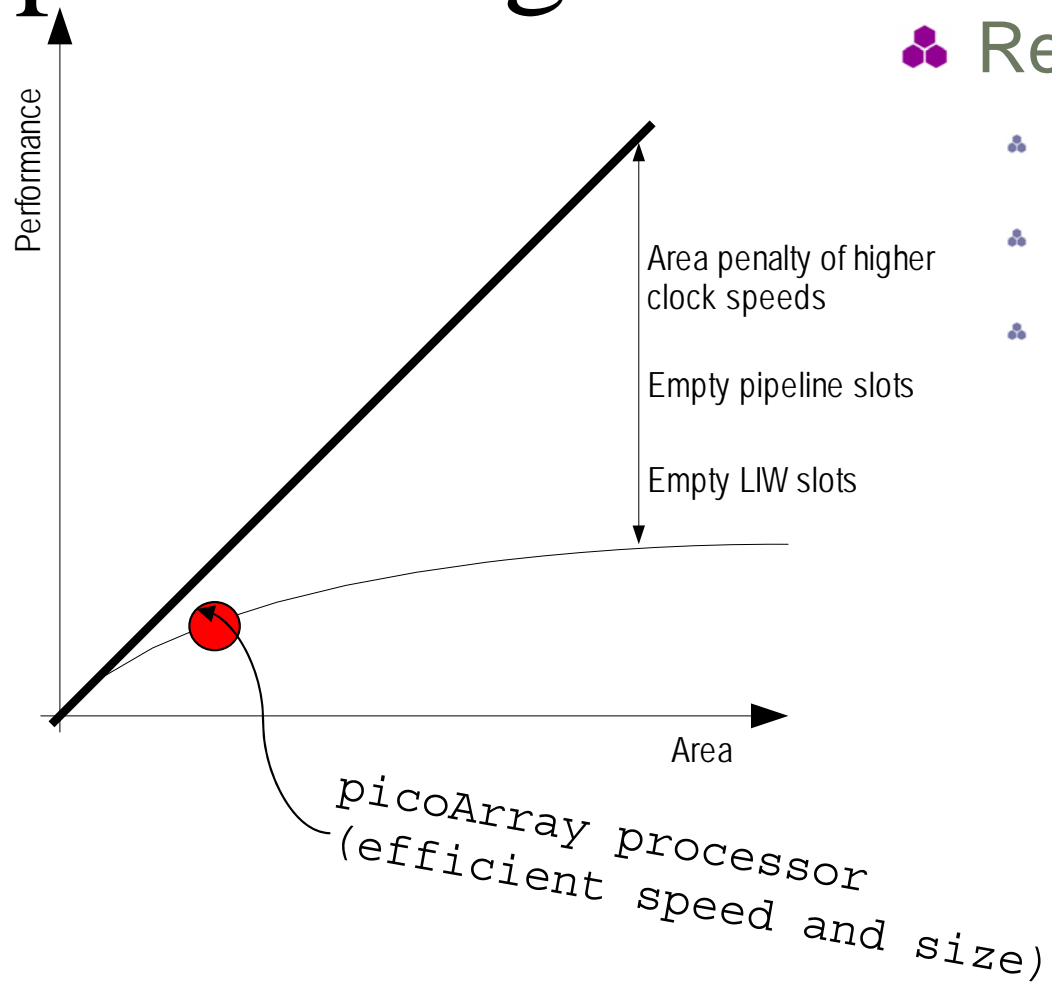
- ❖ NO parallel universe – it's application dependent
- ❖ DSP/flow processing gains from massively parallel
- ❖ picoChip have made this work for real systems

No flies .. on the menu

- ❁ picoChip have fished the flies from the soup of parallel processing... leaving it.
 - ❁ Palatable for your programmers
 - ❁ Nourishing for your application
 - ❁ Tasty for your customers
- ❁ Parallel processing is on the menu



Reduced gains from serial processing



❖ Reducing gains of serial tricks

- ❖ Higher clock speeds
- ❖ Deeper pipelines
- ❖ Wider LIW