CS294-7: Reconfigurable Computing

Day 6: Thursday, February 6

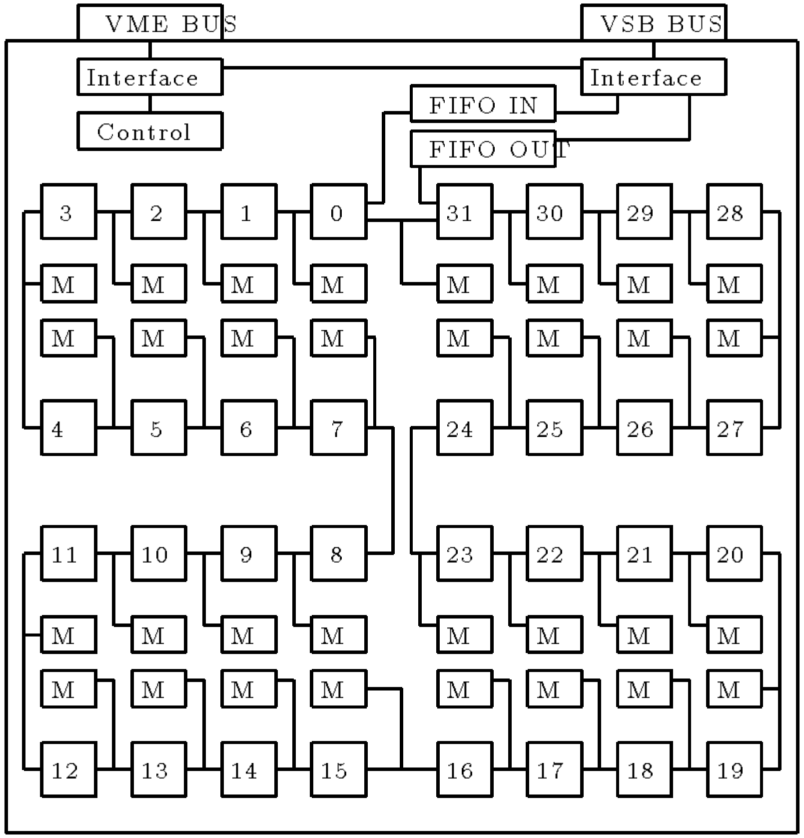
- Admin
- SPLASH – edit distance calculation

Admin

- General questions on assignment?
- Still fighting w/ CAD tool situation...
- This Friday, 2 pm, RC talk at ICSI
- Next Class: Tuesday, 5 pm, 405

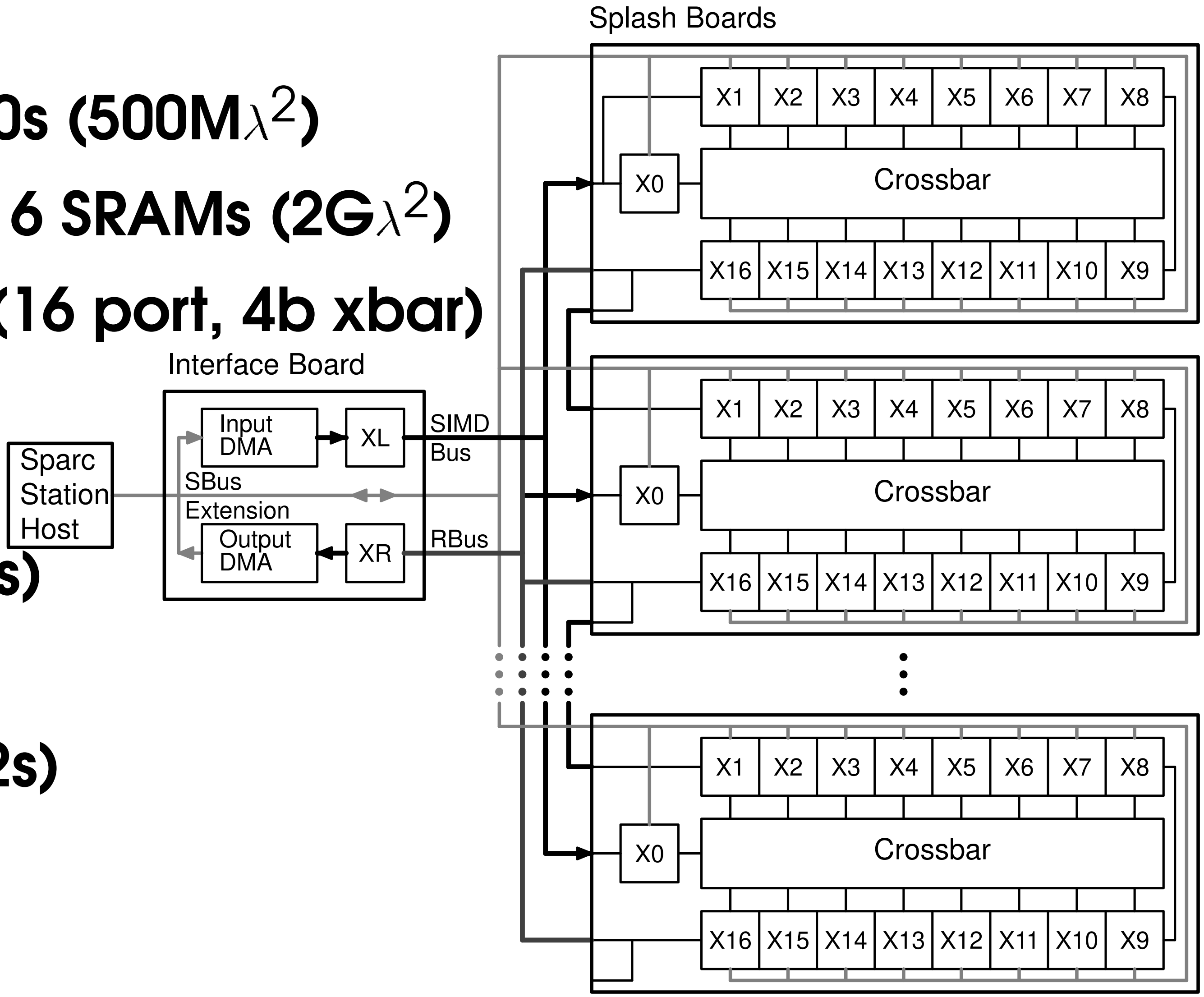
SPLASH Architecture

- 32 XC3090s (420Mλ^2)
- 32 128K$\times$8 SRAMs (600Mλ^2)
- 33Gλ^2
- VMEbus interface (FIFO 1MHz clock)
- Operational 1989



SPLASH-2 Architecture

- 17 XC4010s (500Mλ^2)
- 17 256K$\times$16 SRAMs (2Gλ^2)
- 9 TI8841s (16 port, 4b xbar)
- 43Gλ^2
- SBUS ($<$100MB/s)
- 1992
- (3 XC4062s)



Edit Distance Problem

Problem: What is the cost of transforming s_1 into s_2?

Given: Costs for insertion, deletion, and substitution.

Relevance:

Cataloging "similarity" of two DNA sequences.

- suggests evolutionary relationship?
- structure is a predictor of function

When Biologists characterize a new sequence, want to compare it to the (growing) database of known sequences.

Alphabet and Costs

4 Letter "Alphabet" of bases:

- A (Adenine)
- C (Cytosine)
- G (Guanine)
- T (Thymine)

Simple cost function:

- insertion – 1
- deletion – 1
- substitution – 2
- match – 0

Example

SHOMOO		
$\Downarrow$	insert E	1
SHMOOE		
$\Downarrow$	delete M	1
SHOOE		
$\Downarrow$	substitute R	2
SHORE		
	Total Cost:	4

Dynamic Programming Solution

Source sequence: $s_1, s_2, \ldots s_m$
Target sequence: $t_1, t_2, \ldots t_n$
$d_{i,j} \equiv$ **distance between subsequence** $s_1, s_2, \ldots s_i$
and subsequence $t_1, t_2, \ldots t_j$**.**

$$
\begin{aligned}
d_{0,0} &= 0 \\
d_{i,0} &= d_{i-1,0} + \textbf{Delete}(s_i) \\
d_{0,j} &= d_{0,j-1} + \textbf{Insert}(t_j) \\
d_{i,j} &= \min \begin{cases} d_{i-1,j} + \textbf{Delete}(s_i) \\ d_{i,j-1} + \textbf{Insert}(t_j) \\ d_{i-1,j-1} + \textbf{Substitute}(s_i, t_j) \end{cases}
\end{aligned}
$$

Distance(Source,Target) = $d_{m,n}$

Example

		S	H	O	R	E
S						
H						
M						
O						
O						

Example

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

Systolic Array

if (SCin $\neq$ 0) and (TCin $\neq$ 0)

$$\text{PEDist} \leftarrow \min \begin{cases} \text{PEDist} + \text{Substitute(SCin,TCin)} \\ \text{TDin} + \text{Delete(SCin)} \\ \text{SDin} + \text{Insert(TCin)} \end{cases}$$

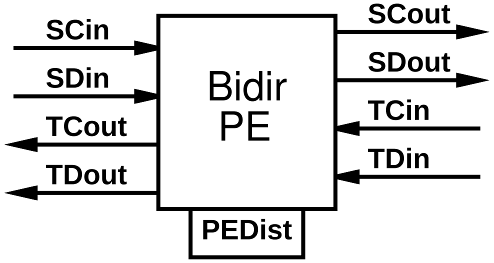
else-if (SCin$\neq$ 0)

PEDist $\leftarrow$ SDin

else-if (TCin$\neq$ 0)

PEDist $\leftarrow$ TDin

endif

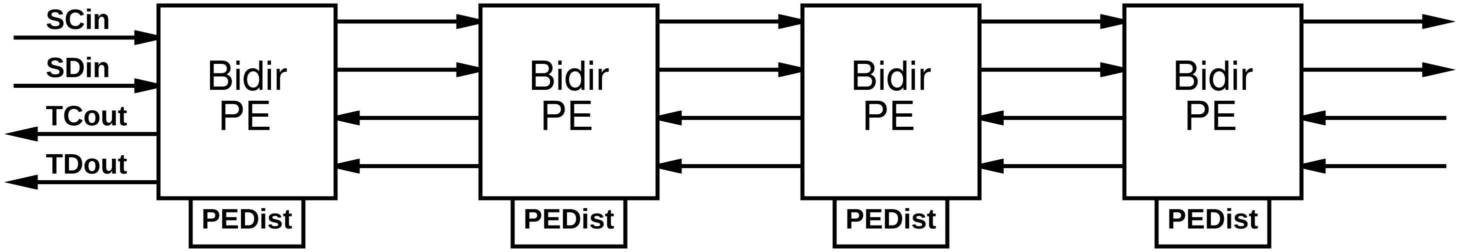


SCout$\leftarrow$SCin

TCout$\leftarrow$TCin

SDout$\leftarrow$PEDist

TDout$\leftarrow$PEDist



Example

<--Timesteps

TC	**O**		**O**		**M**		**H**		**S**										
TD	5		4		3		2		1										
TDout	5		4		3		2		1										
SC											**S**		**H**		**O**		**R**		**E**
SD											1		2		3		4		5
SDout											1		2		3		4		5
PEDist	5	4	4	3	3	2	2	1	1	0	1	1	2	2	3	3	4	4	5
TC		**O**		**O**		**M**		**H**		**S**									
TD		5		4		3		2		1									
TDout		5		4		3		2		0									
SC										**S**		**H**		**O**		**R**		**E**	
SD										1		2		3		4		5	
SDout										0		2		3		4		5	
PEDist	5	5	4	4	3	3	2	2	1	0	1	2	2	3	3	4	4	5	5
TC			**O**		**O**		**M**		**H**		**S**								
TD			5		4		3		2		0								
TDout			5		4		3		1		1								
SC									**S**		**H**		**O**		**R**		**E**		
SD									0		2		3		4		5		
SDout									1		1		3		4		5		
PEDist	5	5	5	4	4	3	3	2	1	0	1	2	3	3	4	4	5	5	5

Example (Continued–2 of 4)

<--Timesteps

TC				**O**		**O**		**M**		**H**		**S**							
TD				5		4		3		1		1							
TDout				5		4		2		0		2							
SC								**S**		**H**		**O**		**R**		**E**			
SD								1		1		3		4		5			
SDout								2		0		2		4		5			
PEDist	5	5	5	5	4	4	3	2	1	0	1	2	3	4	4	5	5	5	5

TC					**O**		**O**		**M**		**H**		**S**						
TD					5		4		2		0		2						
TDout					5		3		1		1		3						
SC							**S**		**H**		**O**		**R**		**E**				
SD							2		0		2		4		5				
SDout							3		1		1		3		5				
PEDist	5	5	5	5	5	4	3	2	1	0	1	2	3	4	5	5	5	5	5

TC						**O**		**O**		**M**		**H**		**S**					
TD						5		3		1		1		3					
TDout						4		2		2		2		4					
SC						**S**		**H**		**O**		**R**		**E**					
SD						3		1		1		3		5					
SDout						4		2		2		2		4					
PEDist	5	5	5	5	5	4	3	2	1	2	1	2	3	4	5	5	5	5	5

Example (Continued–3 of 4)

<--Timesteps

TC							**O**		**O**		**M**		**H**		**S**				
TD							4		2		2		2		4				
TDout							3		2		3		3		4				
SC					**S**		**H**		**O**		**R**		**E**						
SD					4		2		2		2		4						
SDout					4		3		2		3		3						
PEDist	5	5	5	5	4	4	3	2	2	2	3	2	3	4	4	5	5	5	5
TC								**O**		**O**		**M**		**H**		**S**			
TD								3		2		3		3		4			
TDout								2		3		4		3		4			
SC				**S**		**H**		**O**		**R**		**E**							
SD				4		3		2		3		3							
SDout				4		3		2		3		4							
PEDist	5	5	5	4	4	3	3	2	2	3	3	4	3	3	4	4	5	5	5
TC									**O**		**O**		**M**		**H**		**S**		
TD									2		3		4		3		4		
TDout									3		4		4		3		4		
SC			**S**		**H**		**O**		**R**		**E**								
SD			4		3		2		3		4								
SDout			4		3		2		3		4								
PEDist	5	5	4	4	3	3	2	2	3	3	4	4	4	3	3	4	4	5	5

Example (Continued–4 of 4)

<--Timesteps

TC										**O**		**O**		**M**		**H**		**S**	
TD										3		4		4		3		4	
TDout										4		4		4		3		4	
SC		**S**		**H**		**O**		**R**		**E**									
SD		4		3		2		3		4									
SDout		4		3		2		3		4									
PEDist	5	4	4	3	3	2	2	3	3	4	4	4	4	4	3	3	4	4	5

Bidirectional Statistics

Space	16 CLBs / PE (24 PEs / XC4010) (384 PEs/Splash2 board)
Time	5.5M characters/s 91ns $\times$ 2 cycles / character

Bidirectional Shortcomings

- New symbol every other cycle
- Half PEs idle, on average

		S	H	O	R	E
	0	1	2	3	4	5
S	1	0	1	2	3	4
H	2	1	0	1	2	3
M	3	2	1	2	3	4
O	4	3	2	1	2	3
O	5	4	3	2	3	4

Also note:

In practice, comparing large database against a new source

Unidirectional

- Store source string in the array
- Each PE computes one row in distance matrix
- Only requires n PEs
- Can input a new symbol each cycle
- Overlaps startup/unload

		S	H	O	R	E	PR	S	M	U	T
	0	1	2	3	4	5	0	1	2	3	4
S	1	0	1	2	3	4	1	0	1	2	3
H	2	1	0	1	2	3	2	1	2	3	4
M	3	2	1	2	3	4	3	2	1	2	3
O	4	3	2	1	2	3	4	3	2	3	4
O	5	4	3	2	3	4	5	4	3	4	5

Unidirectional Statistics

Space 28 CLBs / PE
(14 PEs / XC4010)
(248 PEs/Splash2 board)

Time 12M characters/s
83ns / character

Comparison:

Bidir	16 CLBs $\times$ 91ns $\times$ 2	$\Rightarrow$ 2912 CLB-ns
Unidir	28 CLBs $\times$ 83ns	$\Rightarrow$ 2324 CLB-ns

(+ factor of 2 loading/unloading)

Datapath Sizes

- Character size depends on alphabet (2 bits w/ AGCT)
- Distances
  - notice adjacent entries always 0,+1,(+2 diagonal)
  - modulo encoding w/ 2 bits
  - accumulator on output to calculate absolute distances

		S	H	O	R	E	PR	S	M	U	T
	0	1	2	3	4	5	0	1	2	3	4
S	1	0	1	2	3	4	1	0	1	2	3
H	2	1	0	1	2	3	2	1	2	3	4
M	3	2	1	2	3	4	3	2	1	2	3
O	4	3	2	1	2	3	4	3	2	3	4
O	5	4	3	2	3	4	5	4	3	4	5

SPLASH Sequence Matching – efficiency

- O(string length) parallelism
- minimal size datapaths (2-bit level granularity)
- orchestrated data movement and retiming
- local control – each PE makes independent decisions

SPLASH Sequence Matching – inefficiency

- **Many elements programmed to do the same thing** (redundant configurations in memory)
- **Long path length** (80-90ns v/s 7ns LUT+interconnect time) $\rightarrow$ **most of the time interconnect and LUTs are idle**
- **unidir: more flexibility than needed each cycle**
  - **seldom used pieces of logic sitting idle**
  - **? generic costs could be specialized?**

Comparisons – Metric

CUPS (Cell Updates Per Second)

An $n \times m$ comparison requires nm CUPs.

Rating hardware/reconfigurable configurations on maximum size configuration (so are solving problems of different sizes)

Software assuming $n = m = 1000$.

Comparisons – Raw

Machine	CUPS	Note
Splash2×16 (unidir)	43,000M	
Splash2×16 (bidir)	17,000M	half Hoang's
Splash2 (unidir)	3,000M	
Splash2 (bidir)	1,050M	half Hoang's
Splash1 (bidir)	370M	i/o limited to 1MHz
P-NAC (34)	500M	IC capabilities
CM-2 (64K)	150M	
CM-5 (32)	33M	
SPARC 10	1.2M	
SPARC 1	0.87M	

Comparisons – Normalized

Processor area only:

Machine	CUPS	λ	Area	CUP/λ^2s
Splash2$\times$16	43,000M	0.60μ	**500M$\lambda^2\times$17$\times$16**	**0.32**
Splash2	3,000M	0.60μ	**500M$\lambda^2\times$16**	**0.38**
Splash1	370M	0.60μ	**420M$\lambda^2\times$32**	**0.028**
P-NAC (34)	500M	2.0μ	**7.8M$\lambda^2\times$34**	**1.9**
CM-2 (64K)	150M	**?**		
CM-5 (32)	33M	32 Sparc2 CPUs		
SPARC 10	1.2M	0.40μ	**1.6Gλ^2**	**0.00075**
SPARC 1	0.87M	0.75μ	**273Mλ^2**	**0.0032**

- FPGAs make up only 20% (40%) of the area on splash2 (splash1).
- SPARC 1/10 CPUs only make up tiny fraction of silicon on workstation motherboard.

Review

- **Edit distance calculation requires $O(mn)$ work**
- **Can identify algorithm with $O(n)$ parallelism and solve in $O(m)$ steps**
- **Systolic implementation w/ $O(n)$ identical processing blocks tuned to problem datasize and dataflow**
- **SPLASH solution**
  - **orders of magnitude more raw performance**
  - **highest programmable performance density**
- **Next Time:** More on empirical measures of performance density