P6 and Native Signal Processing Algorithms

Native Signal Processing, or NSP, is a major industry movement to enhance the base capabilities of the PC platform by running signal processing tasks on a powerful host processor using basic system resources (memory, chip sets) rather than dedicated hardware.

This brief example demonstrates how the P6's Dynamic Execution architecture is particularly good at NSP-type algorithms.

Dynamic Execution Speeds NSP Algorithms

- ***At their core, many NSP algorithms have a tight loop***

```
            i = 0
tightloop:  load data(i)
            process data(i)
            store data(i)
            i = i + 1
            if i<imax goto tightloop
```

Let’s see how the P6 with Dynamic Execution executes this loop

Dynamic Execution Speeds NSP Algorithms

- ***FIRST pass into the loop***

```
            i = 0                              i =   1
tightloop:  load data(i)                             L1
            process data(i)                          -
            store data(i)                            -
            i = i + 1                                x
            if i<imax goto tightloop                 x
```

- *P6 starts first load which is a cache miss*
- *Speculatively executes increment and loop check*
- *Predicts branch back to tightloop*

Dynamic Execution Speeds NSP Algorithms

- ***SECOND pass into the loop***

```
            i = 0
tightloop:  load data(i)
            process data(i)
            store data(i)
            i = i + 1
            if i<imax goto tightloop
```

i =	1	2
	L1	L2
	-	-
	-	-
	x	x
	x	x

- *P6 starts second load which is a cache miss*
- *Speculatively executes increment and loop check*
- *Predicts branch back to tightloop*

Dynamic Execution Speeds NSP Algorithms

- ***THIRD pass into the loop***

```
            i = 0
tightloop:  load data(i)
            process data(i)
            store data(i)
            i = i + 1
            if i<imax goto tightloop
```

i =	1	2	3
	L1	L2	L3
	-	-	-
	-	-	-
	x	x	x
	x	x	x

– *P6 starts third load which is a cache miss*

– *Speculatively executes increment and loop check*

– *Predicts branch back to tightloop*

Dynamic Execution Speeds NSP Algorithms

- ***FOURTH pass into the loop***

```
            i = 0
tightloop:  load data(i)
            process data(i)
            store data(i)
            i = i + 1
            if i<imax goto tightloop
```

i =	1	2	3	4
	L1	L2	L3	L4
	-	-	-	P1
	-	-	-	-
	x	x	x	x
	x	x	x	x

- *P6 starts fourth load which is a cache miss*
- *First data element returns, process it*
- *Speculatively increment, loop check and branch*

Dynamic Execution Speeds NSP Algorithms

- ***FIFTH pass into the loop***

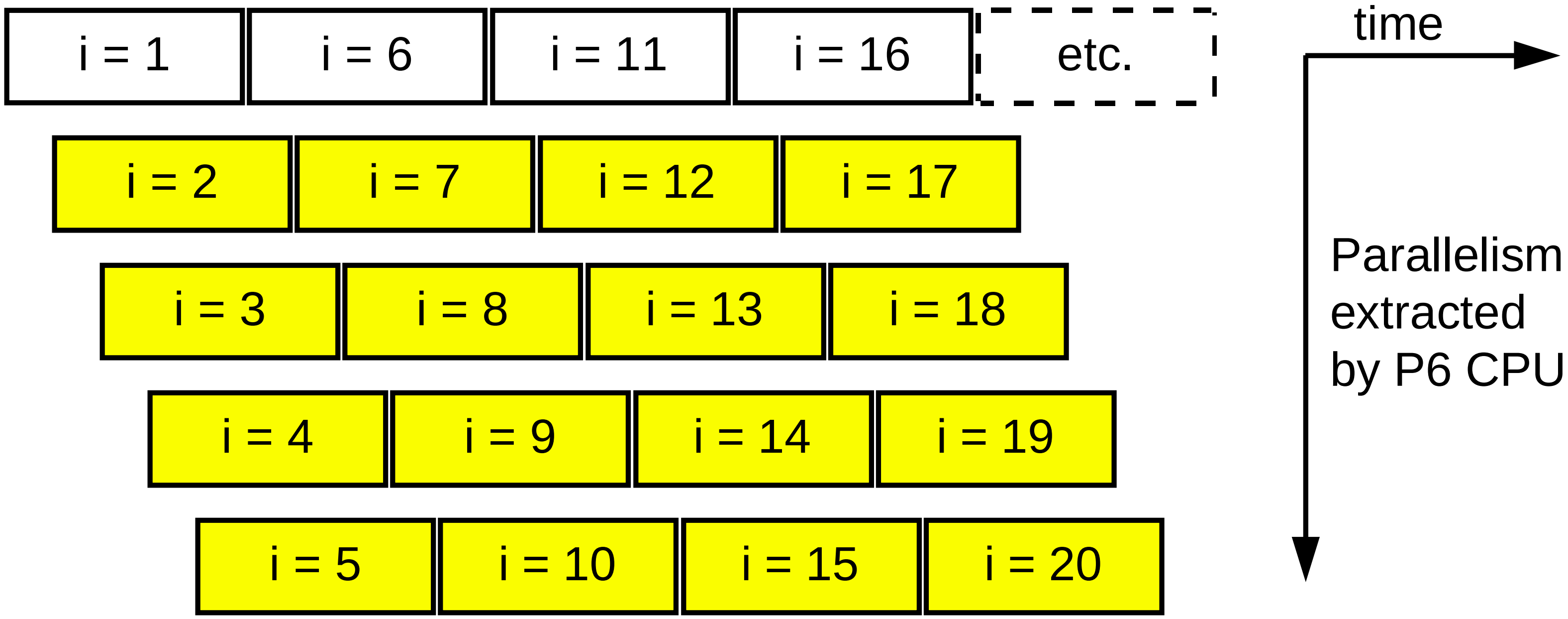
```
            i = 0
tightloop:  load data(i)
            process data(i)
            store data(i)
            i = i + 1
            if i<imax goto tightloop
```

i =	1	2	3	4	5
	L1	L2	L3	L4	L5
	-	-	-	P1	P2
	-	-	-	-	S1
	x	x	x	x	x
	x	x	x	x	x

  - *P6 starts fifth load which is a cache miss*
  - *Second data element returns, process it*
  - *Store the processed first data element*
  - *Speculatively increment, loop check and branch*

Dynamic Execution Speeds NSP Algorithms

- P6, using DE, is automatically UNROLLING LOOPS



i = 1 i = 6 i = 11 i = 16 etc.

i = 2 i = 7 i = 12 i = 17

i = 3 i = 8 i = 13 i = 18

i = 4 i = 9 i = 14 i = 19

i = 5 i = 10 i = 15 i = 20

time

Parallelism
extracted
by P6 CPU

- *Elements i=1,2,3,4,5 are processed in parallel*
- *P6 does useful work while waiting for cache miss*
- *In this example, got a 5x execution speed up*