# Dynamic Dependency Graphs

## *How much parallelism is out there?*

UNIVERSITY OF CAMBRIDGE

Jonathan Mak & Alan Mycroft
University of Cambridge
Computer Laboratory

### End of the "Free Lunch"

The advance of multi-core architectures is forcing the programming community to think about extracting parallelism from programs as they won't automatically speed up with newer processors.

### The Dynamic Dependency Graph

As a measure of the parallelism that can possibly be exploited during the execution of a program, we built an analyser to examine the execution traces and construct their "Dynamic Dependency Graphs". From such graphs we can extract the critical path and hence calculate the limit of parallelism (extending [Wall 1993]), or the time taken in an ideal world given an unlimited number of processor cores and zero communication costs.

**True dependency (Read-after-Write)**

```
add r4, r5, r6

sub r2, r3, r4
```

**Anti-dependency (Write-after-Read)**

```
add r4, r5, r6

sub r6, r2, r3
```

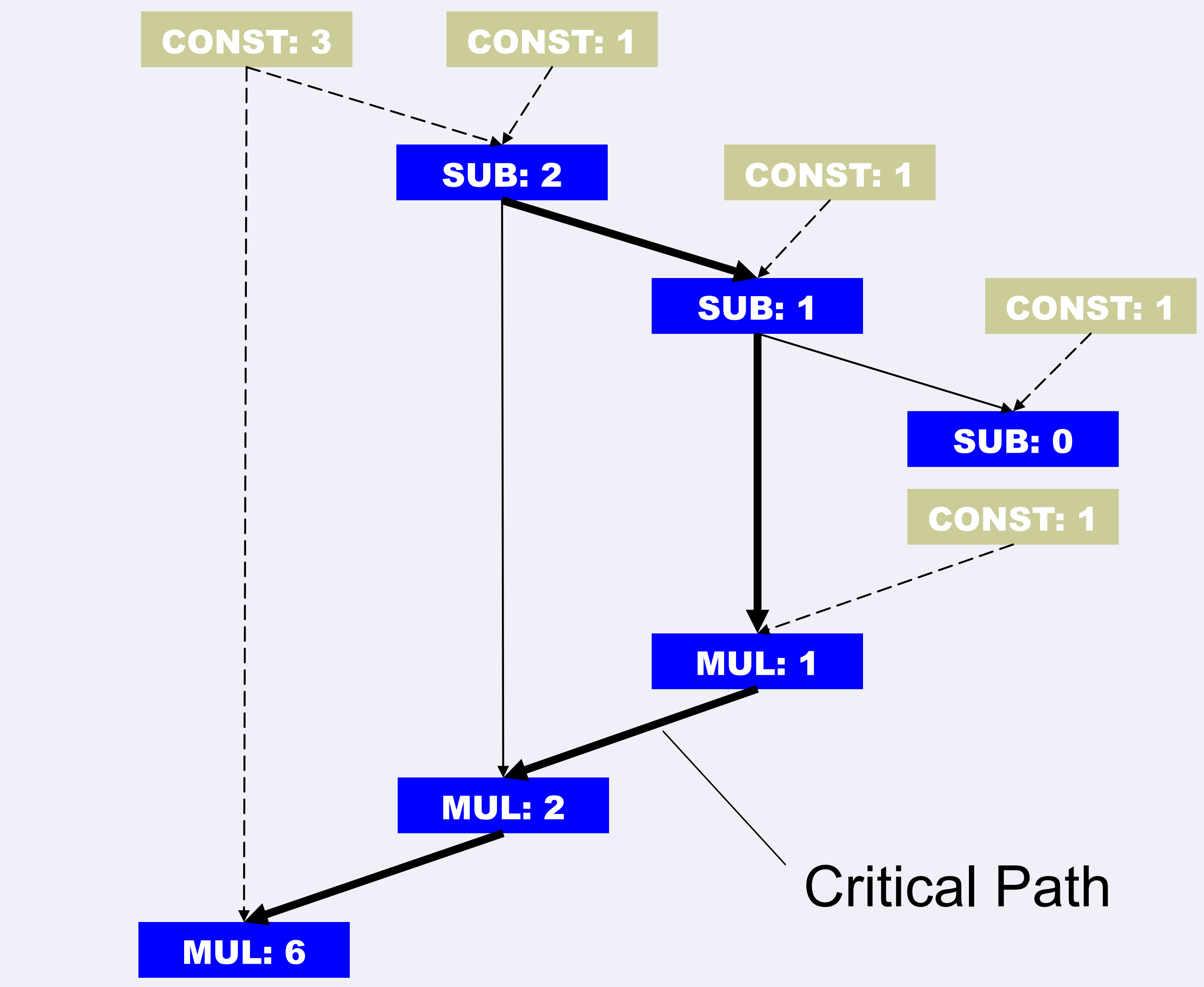
**Output dependency (Write-after-Write)**

```
add r4, r5, r6

sub r4, r2, r3
mul r7, r4, r8
```

**Control dependency (Branch/jump)**

```
  beq r2, r3, L

  add r4, r5, r6
L: ...
```

### Factorial(3)*



* Usual recursive definition, i.e. `Factorial(n) = if (n=0) then 1 else n * Factorial(n-1)`

$$\mathit{Parallelism} = \frac{\mathit{Total\ number\ of\ nodes}}{\mathit{Length\ of\ Critical\ Path}} = \frac{6}{5} = 1.2$$

### Types of dependency

To look at realising some of this parallelism, we examine the various types of dependency that make up the Dynamic Dependency Graph, and the effect they have on the available parallelism*.

### True dependencies on the Stack

On some examples it is found that a lot of true dependencies on the critical path are due to increments and decrements of the Stack/Frame Pointers. By employing a tree-like version of the execution stack known as a "Spaghetti Stack" we can alleviate this problem.

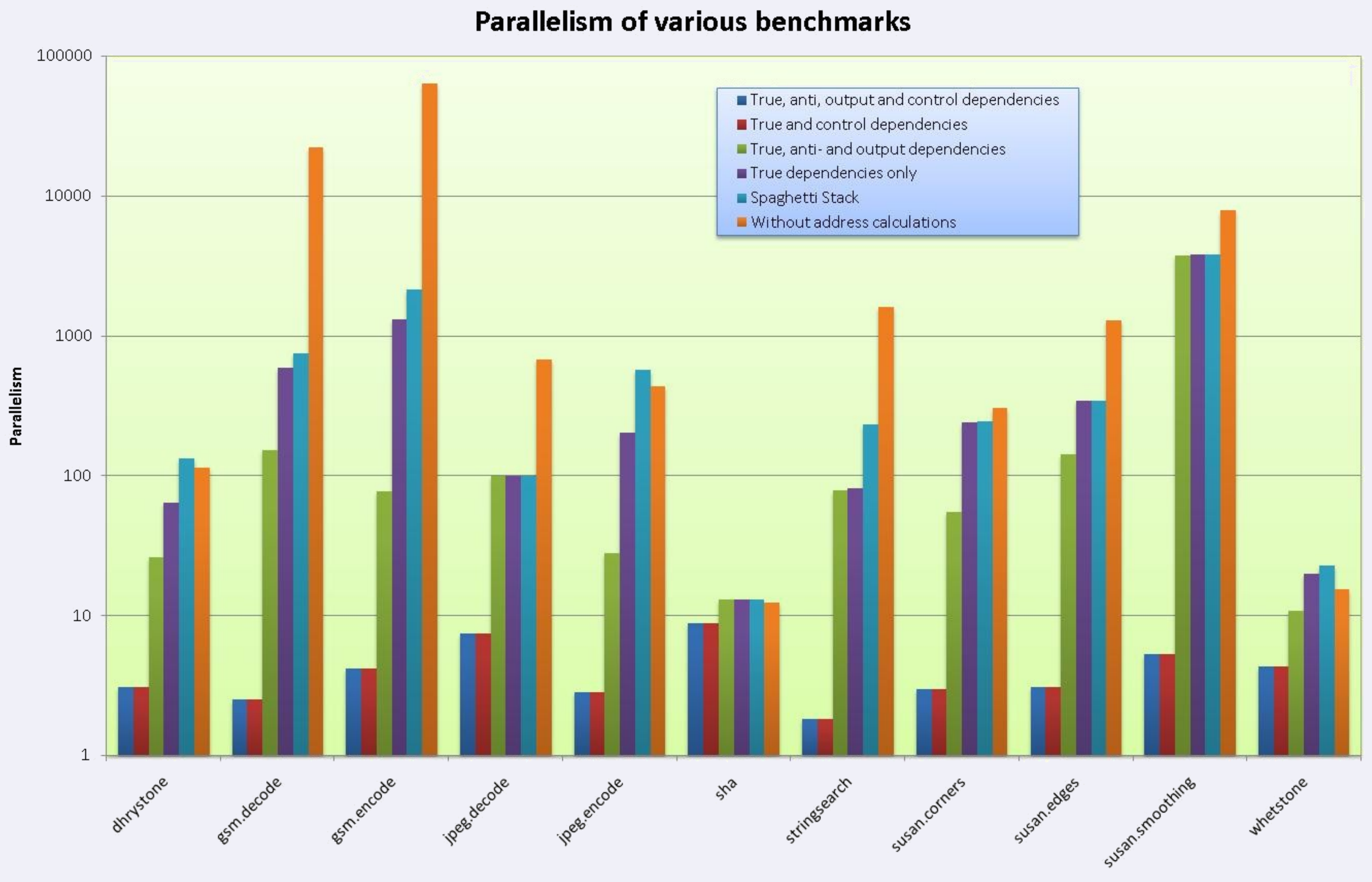
* We consider true dependencies for registers and memory locations, and output and anti-dependencies for memory alone, as register renaming, which can remove output and anti-dependencies for registers, is relatively trivial and standard among today's processors.

### Conclusions

- Many existing programs have lots (100+) of potential implicit parallelism.
- Speculative execution often leads to an over tenfold increase in potential parallelism (🟥🟩 gap on graph)
- Use of "Spaghetti Stack" could further double potential parallelism for some programs (🟪🟦 gap on graph)

### Future Directions

- Bridging theory (this study) and practice (real compilers)
- Transforming sequential programs for Thread-level Speculation


Parallelism of various benchmarks
Parallelism
True, anti, output and control dependencies
True and control dependencies
True, anti- and output dependencies
True dependencies only
Spaghetti Stack
Without address calculations
dhrystone
gsm.decode
gsm.encode
jpeg.decode
jpeg.encode
sha
stringsearch
susan.corners
susan.edges
susan.smoothing
whetstone