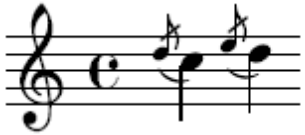
UMassAmherst

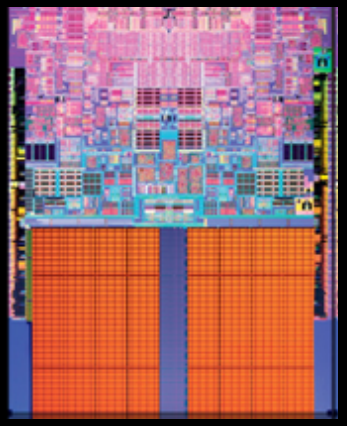
Grace: Safe Multithreaded Programming for C/C++

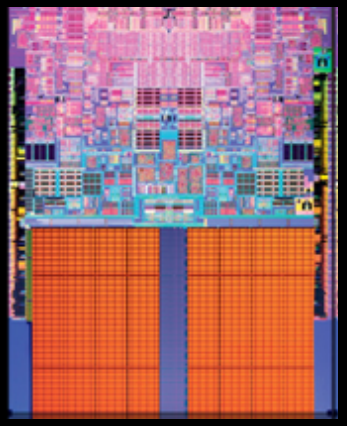


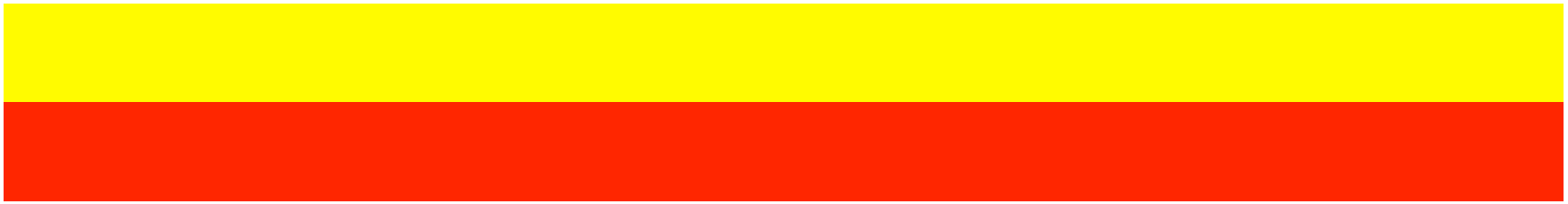
Emery Berger,
Ting Yang, Tongping Liu, Gene Novark

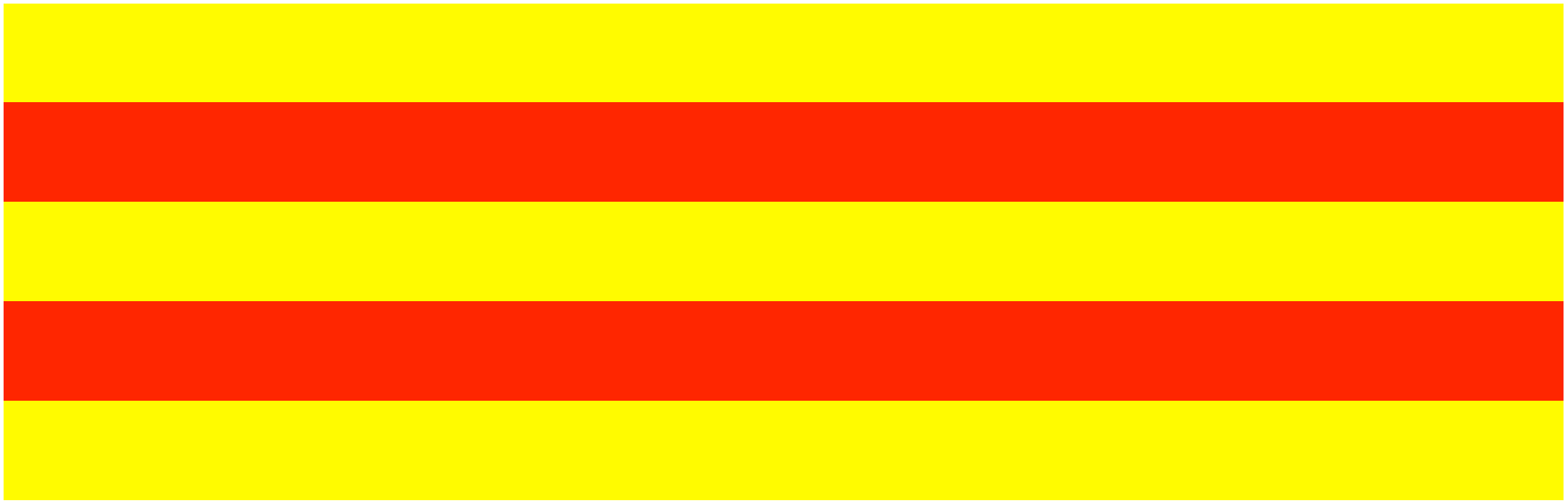
University of Massachusetts, Amherst

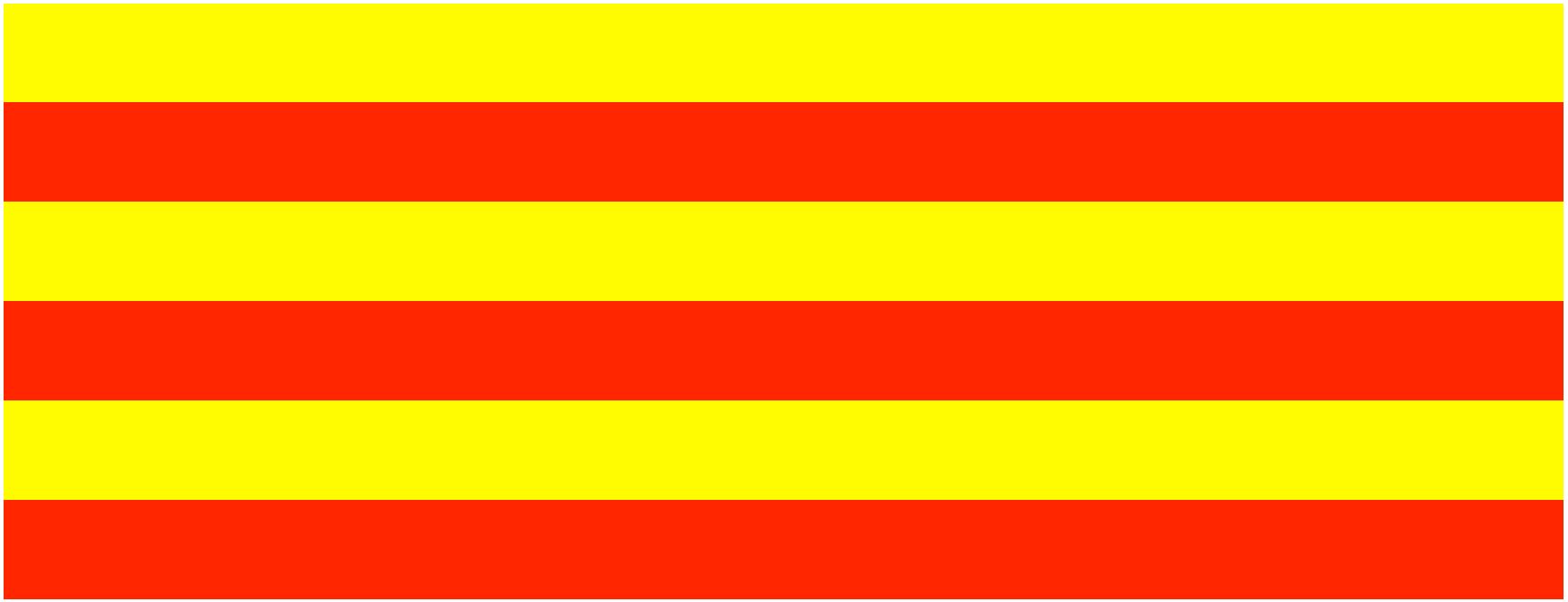
www.cs.umass.edu/~emery

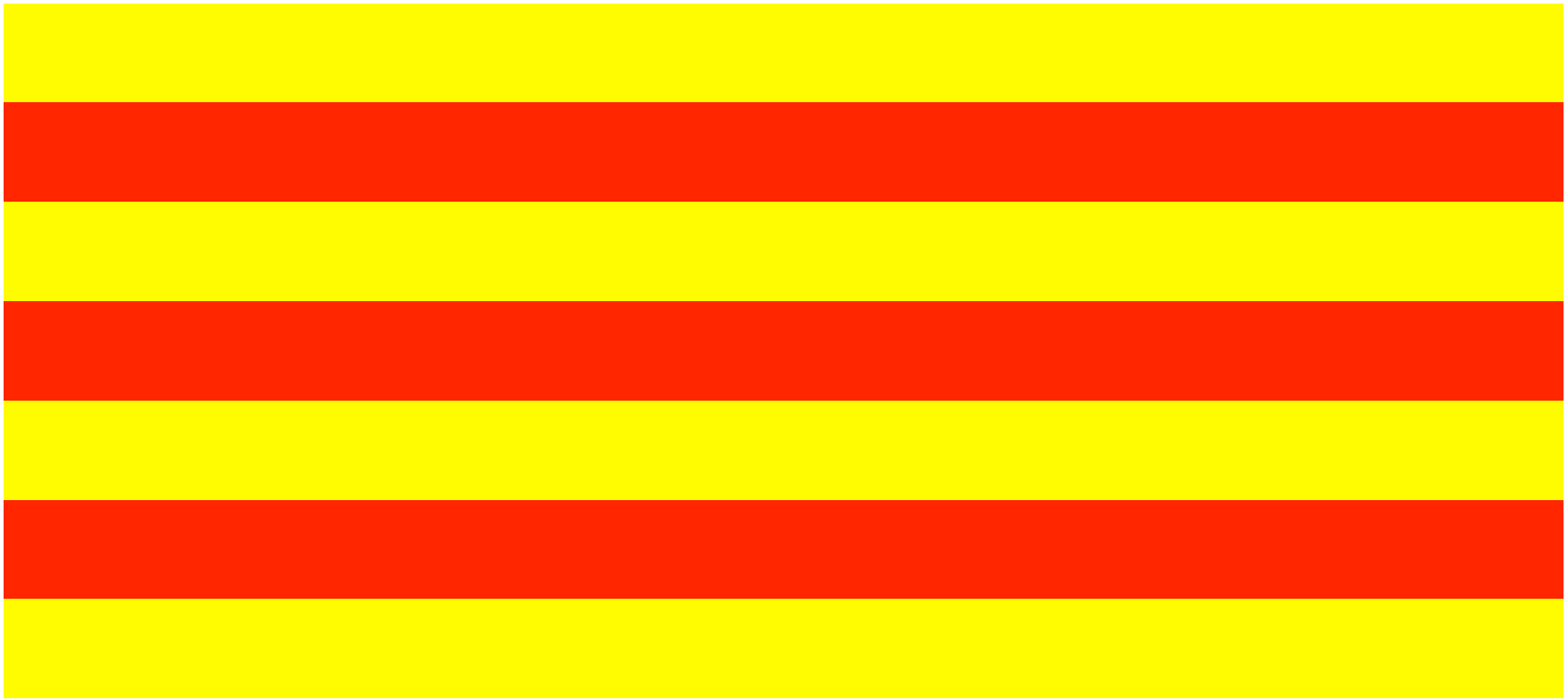


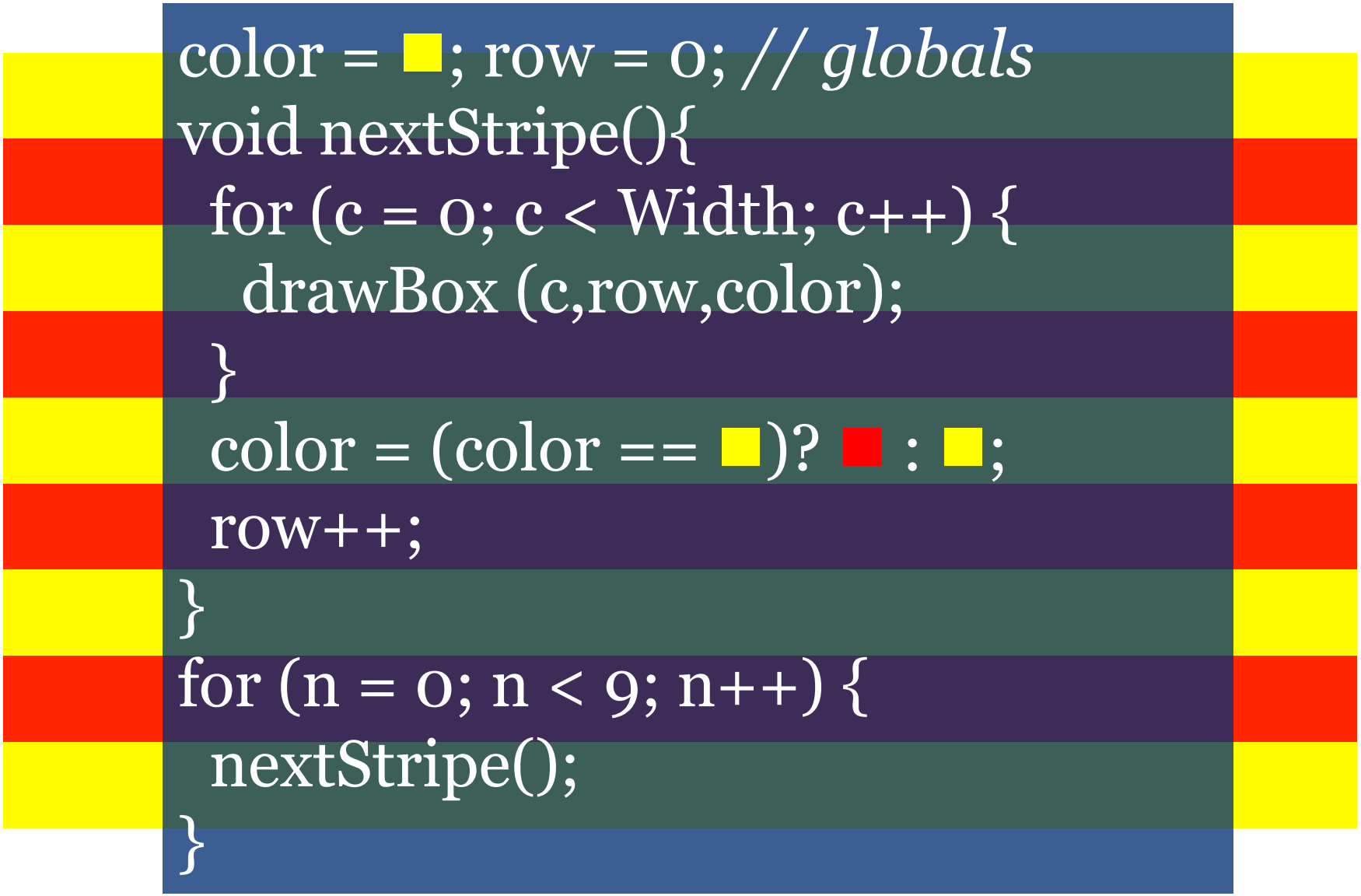


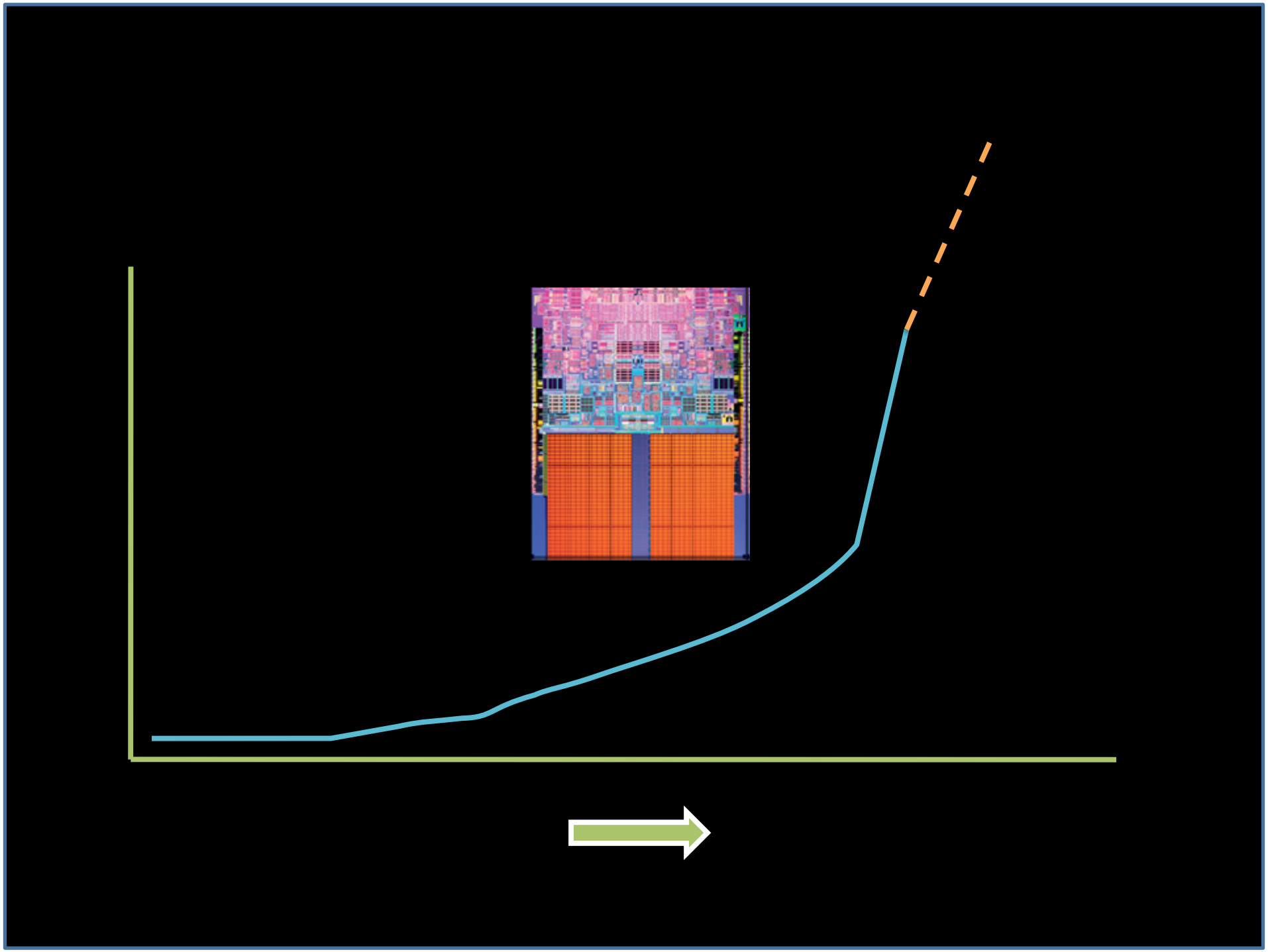


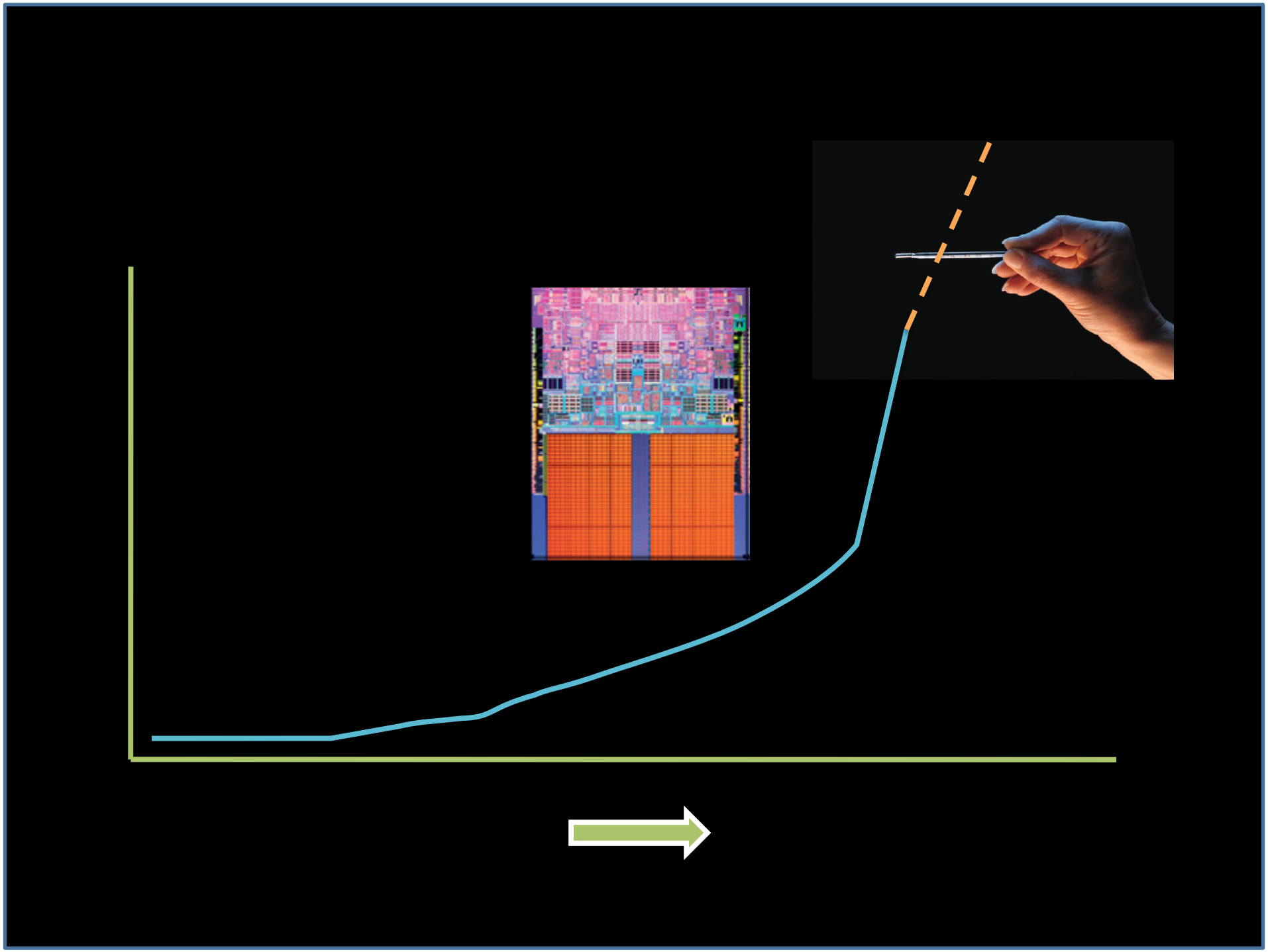


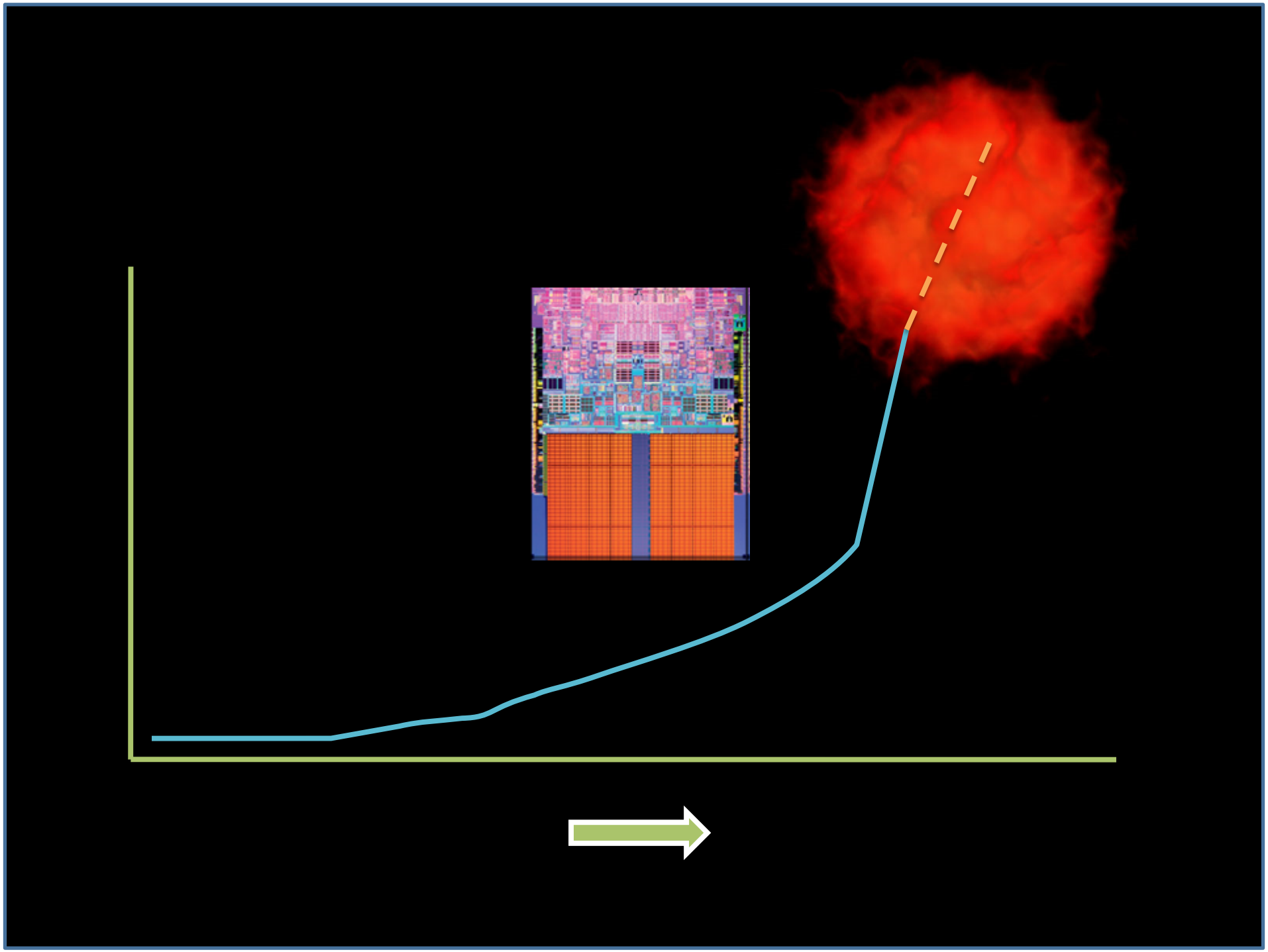


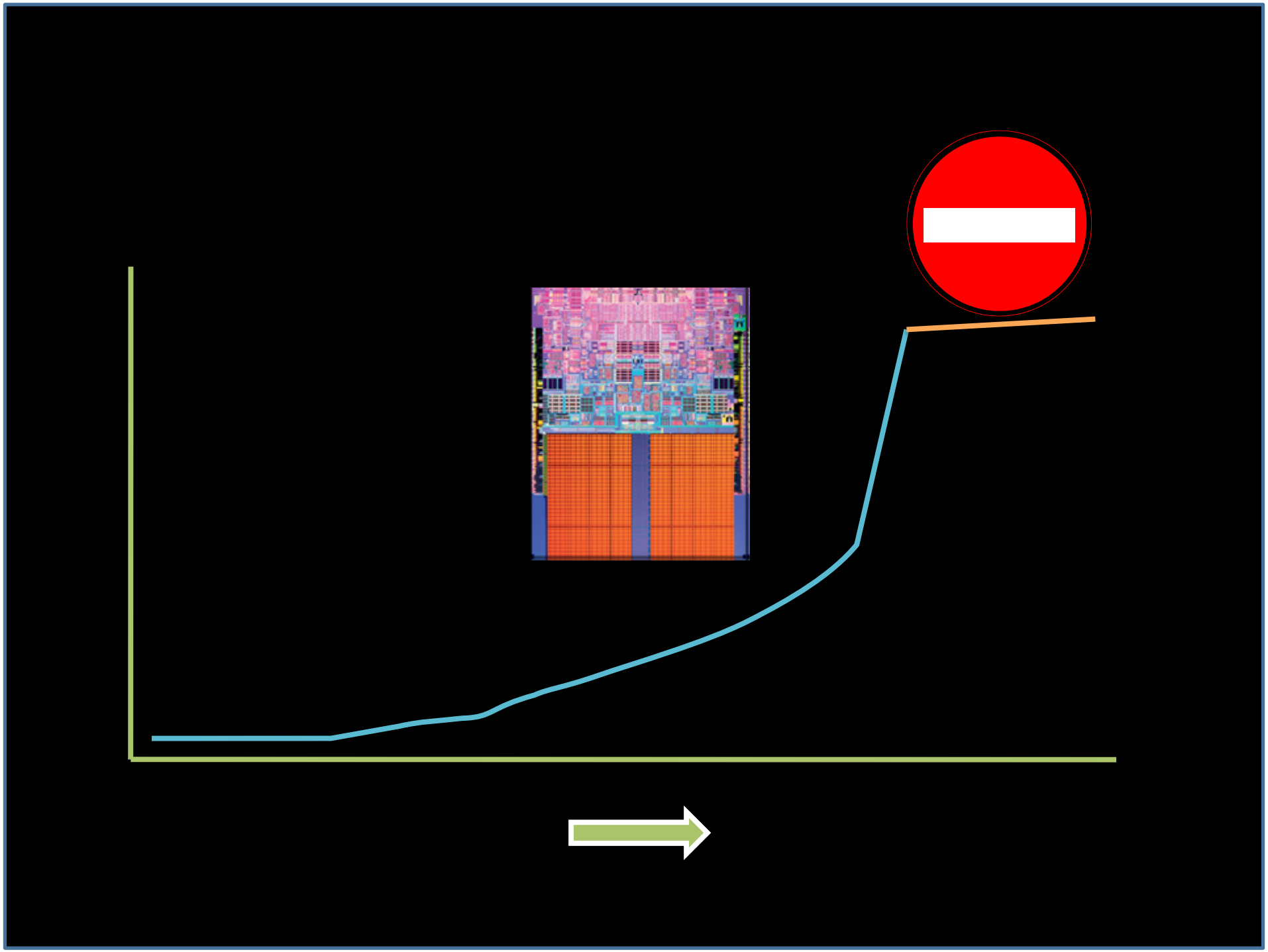


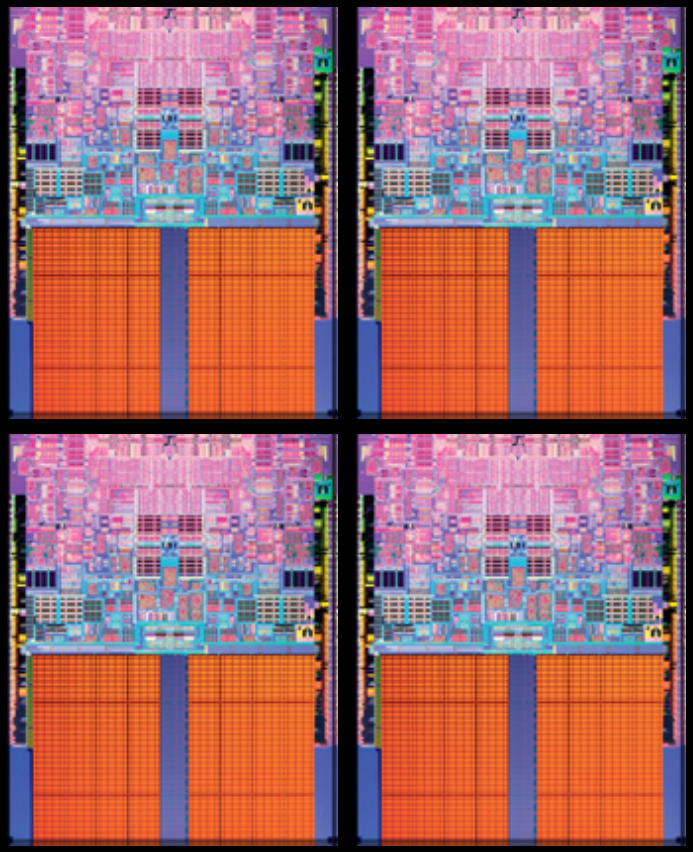

color = ■; row = 0; // globals
void nextStripe(){
for (c = 0; c < Width; c++) {
drawBox (c,row,color);
}
color = (color == ■)? ■ : ■;
row++;
}
for (n = 0; n < 9; n++) {
nextStripe();
}

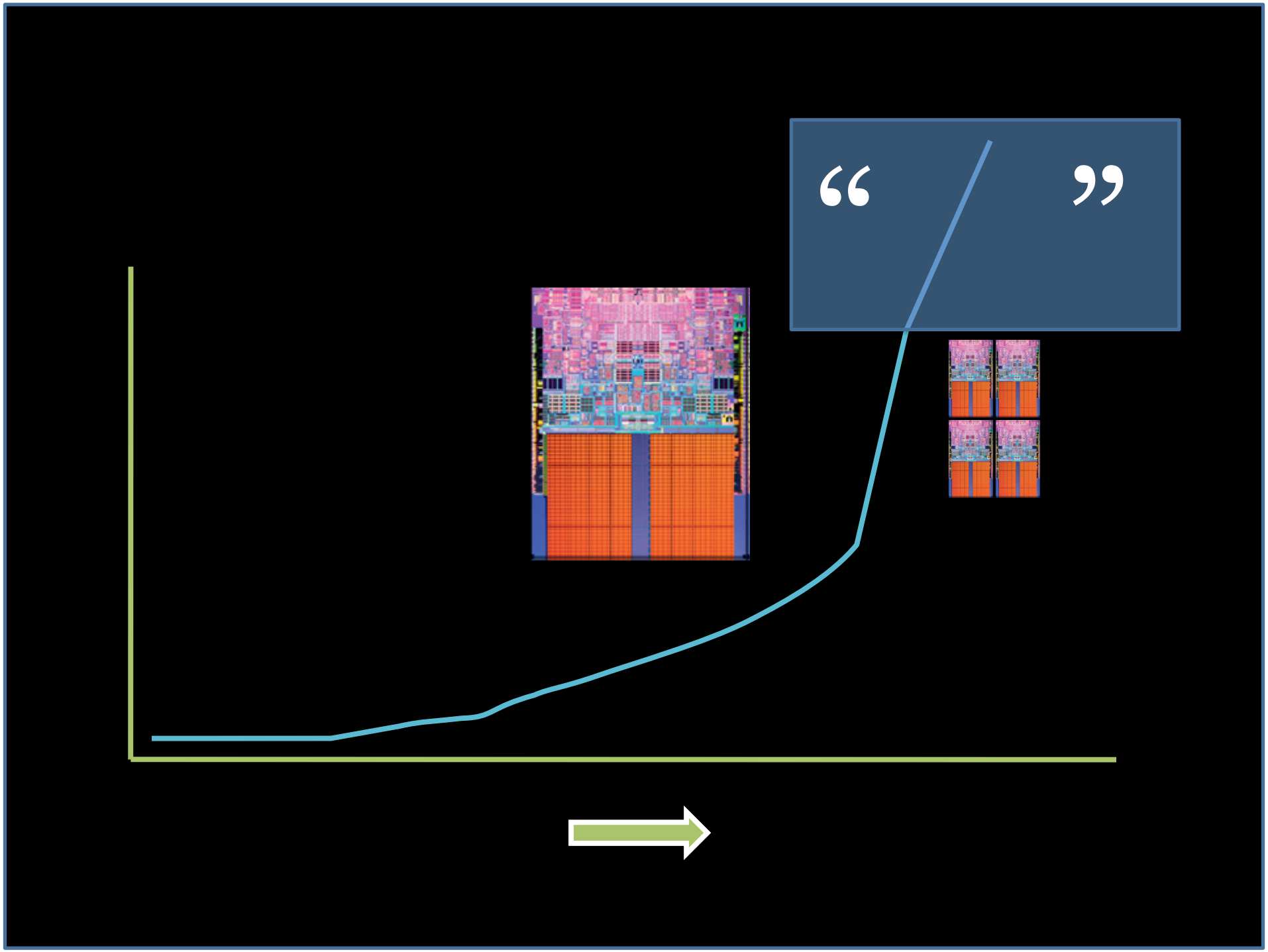


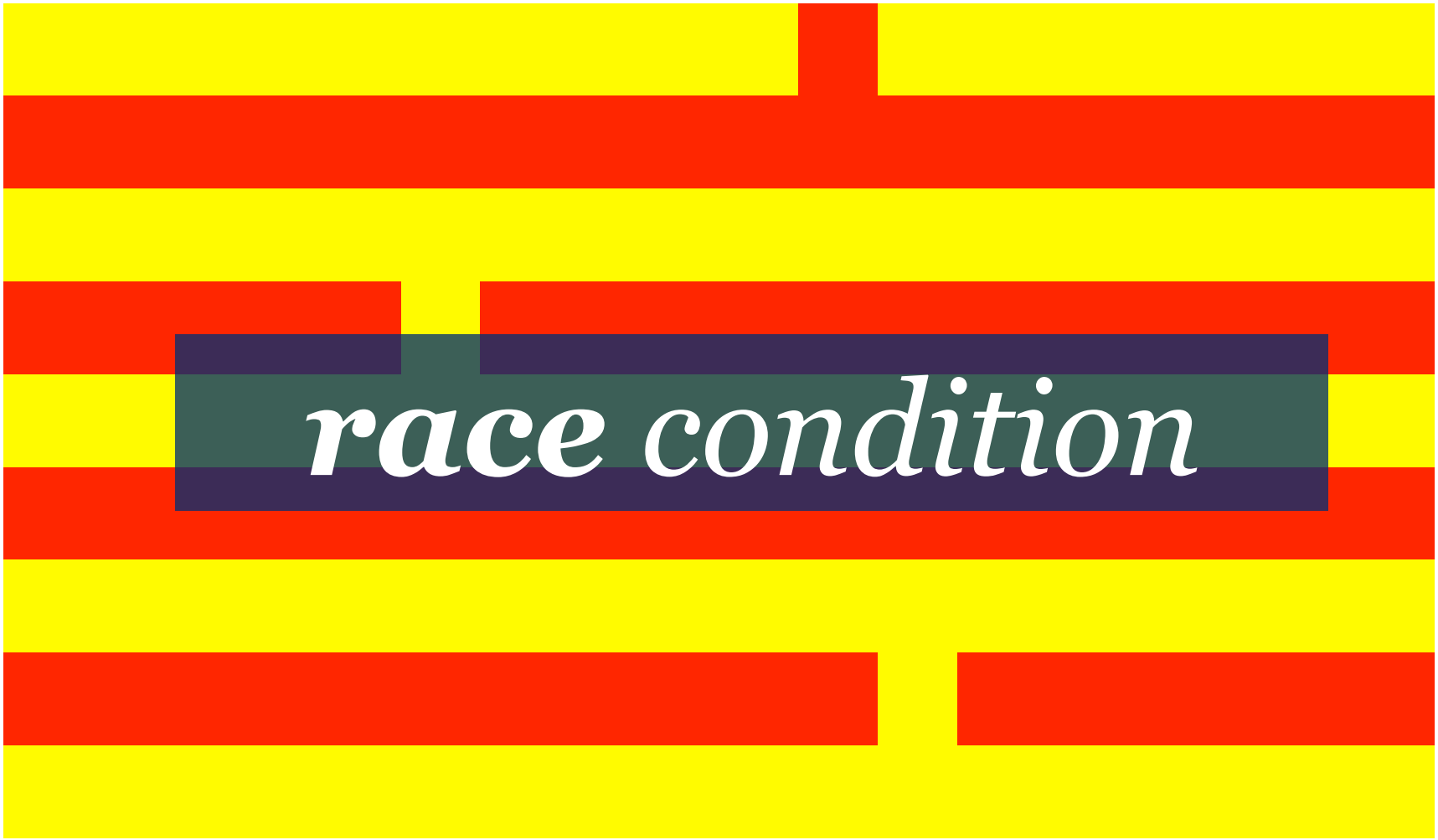


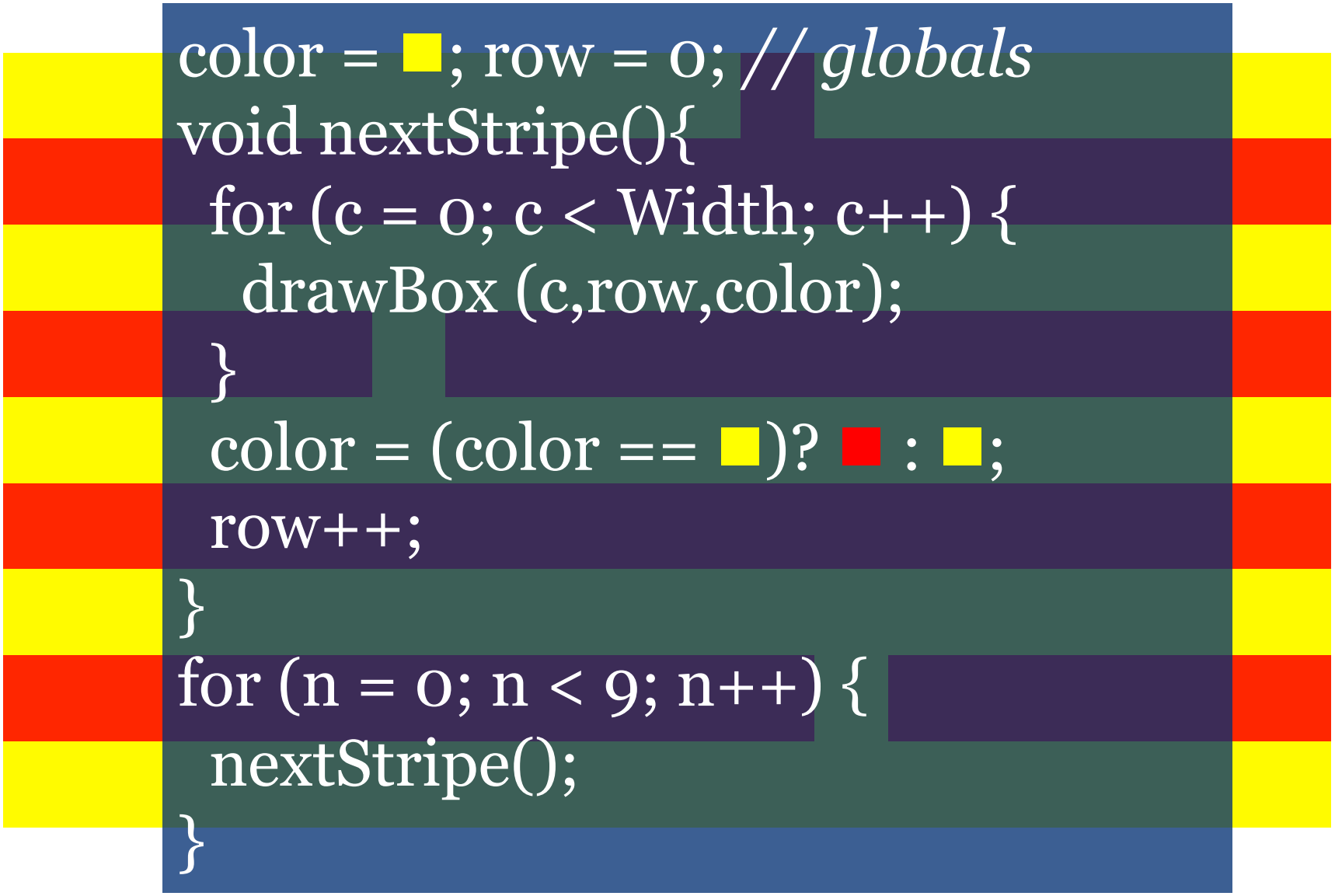


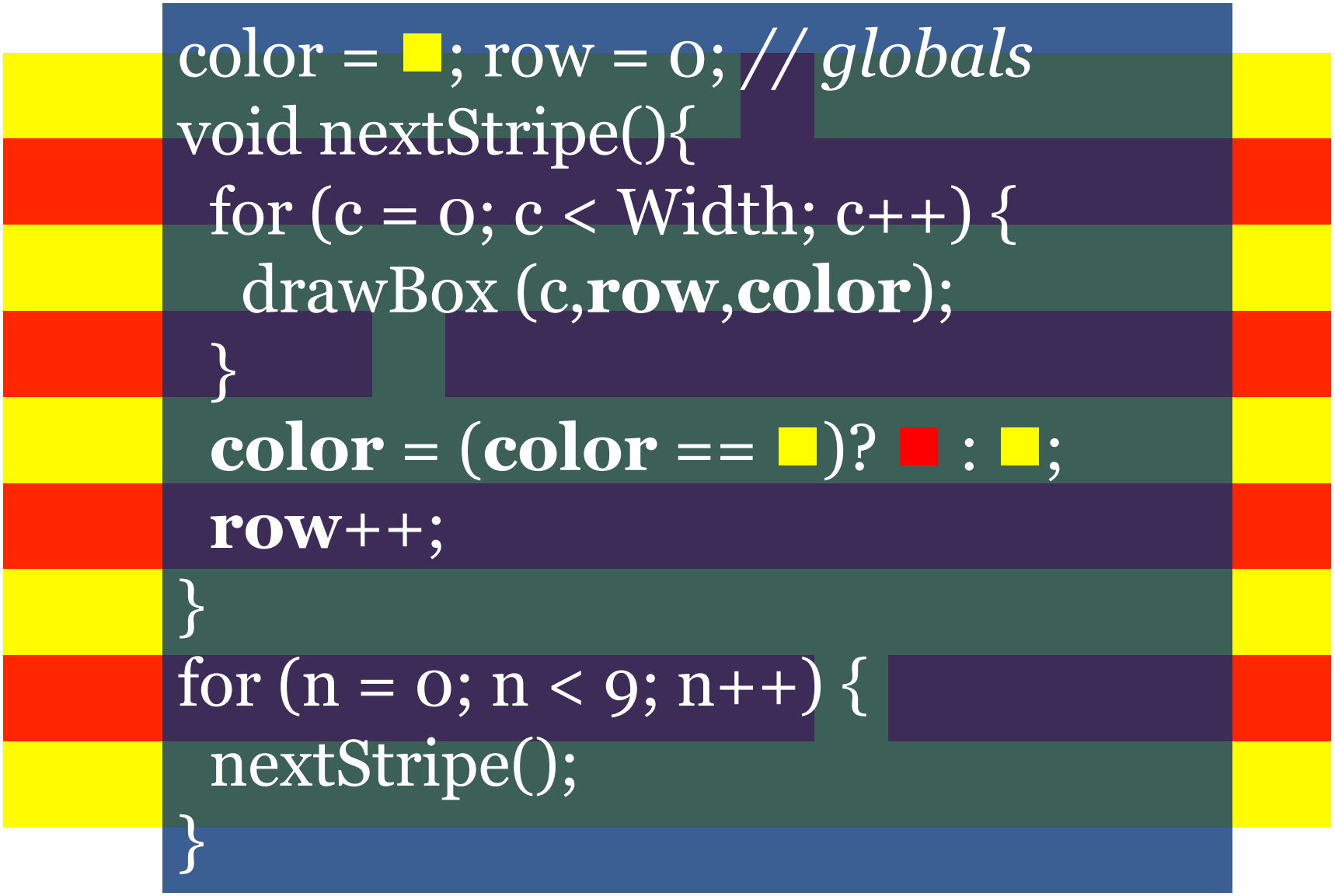


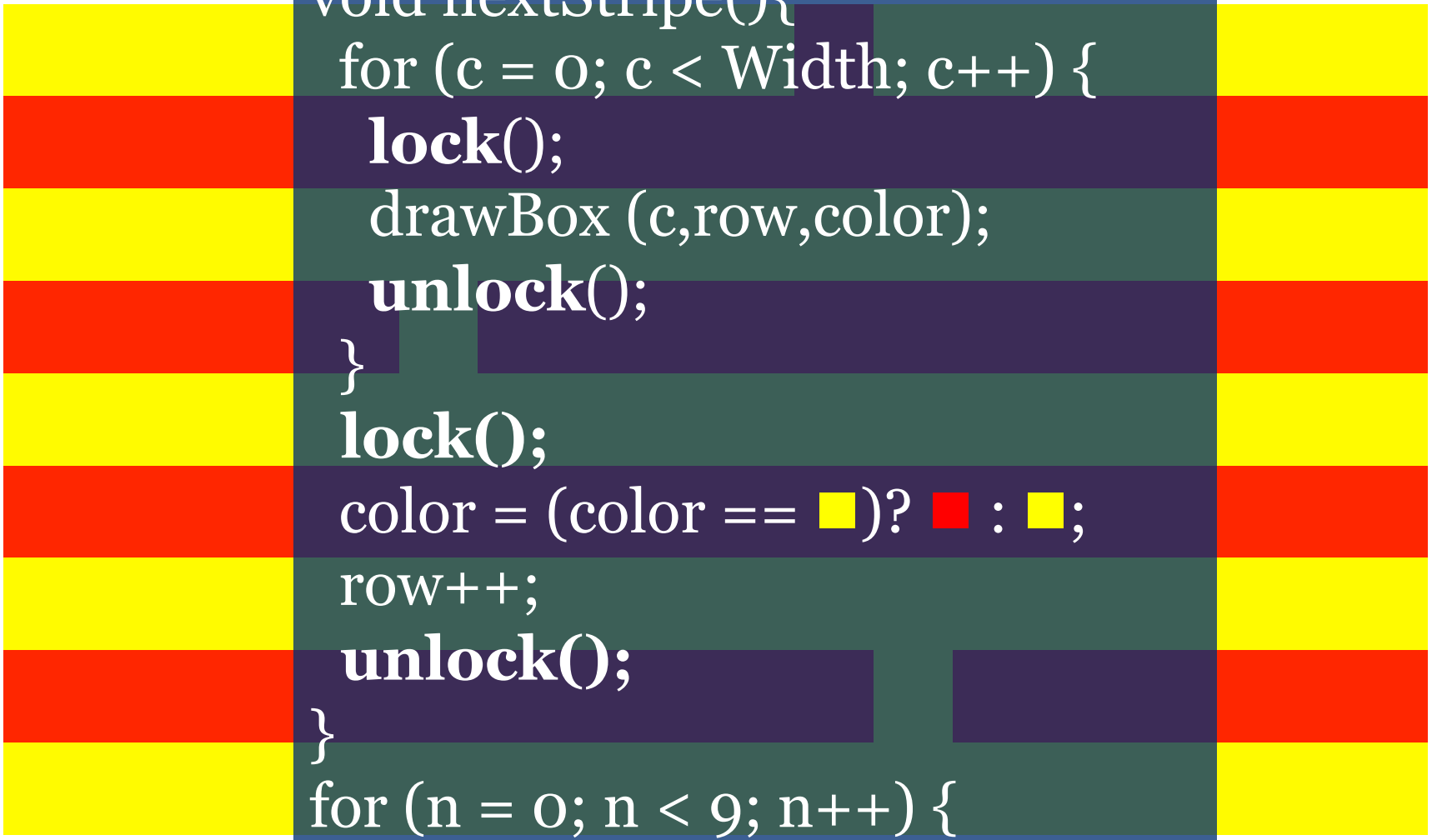


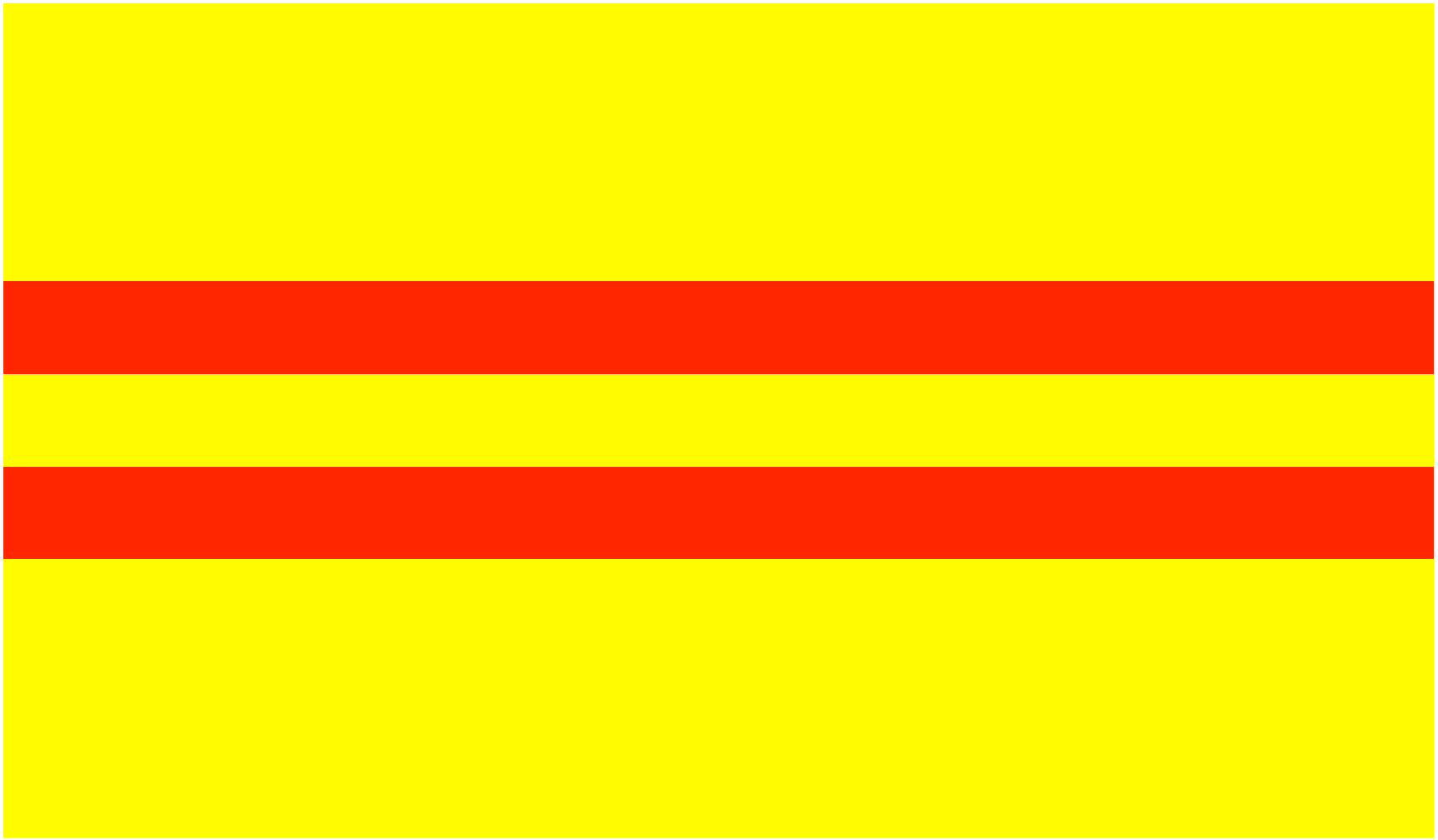


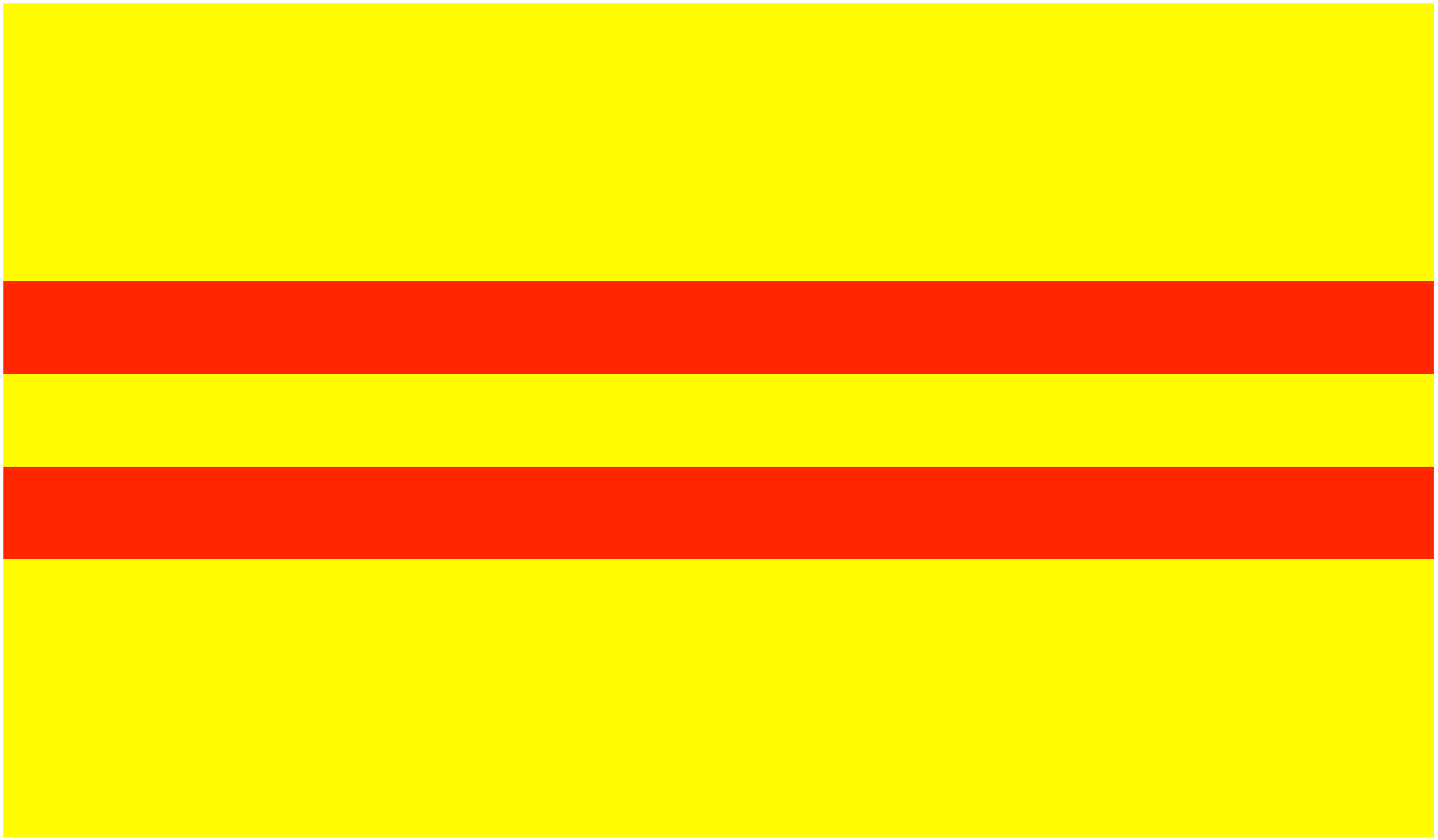

race condition


color = ■; row = 0; // globals
void nextStripe(){
for (c = 0; c < Width; c++) {
drawBox (c,row,color);
}
color = (color == ■)? ■ : ■;
row++;
}
for (n = 0; n < 9; n++) {
nextStripe();
}

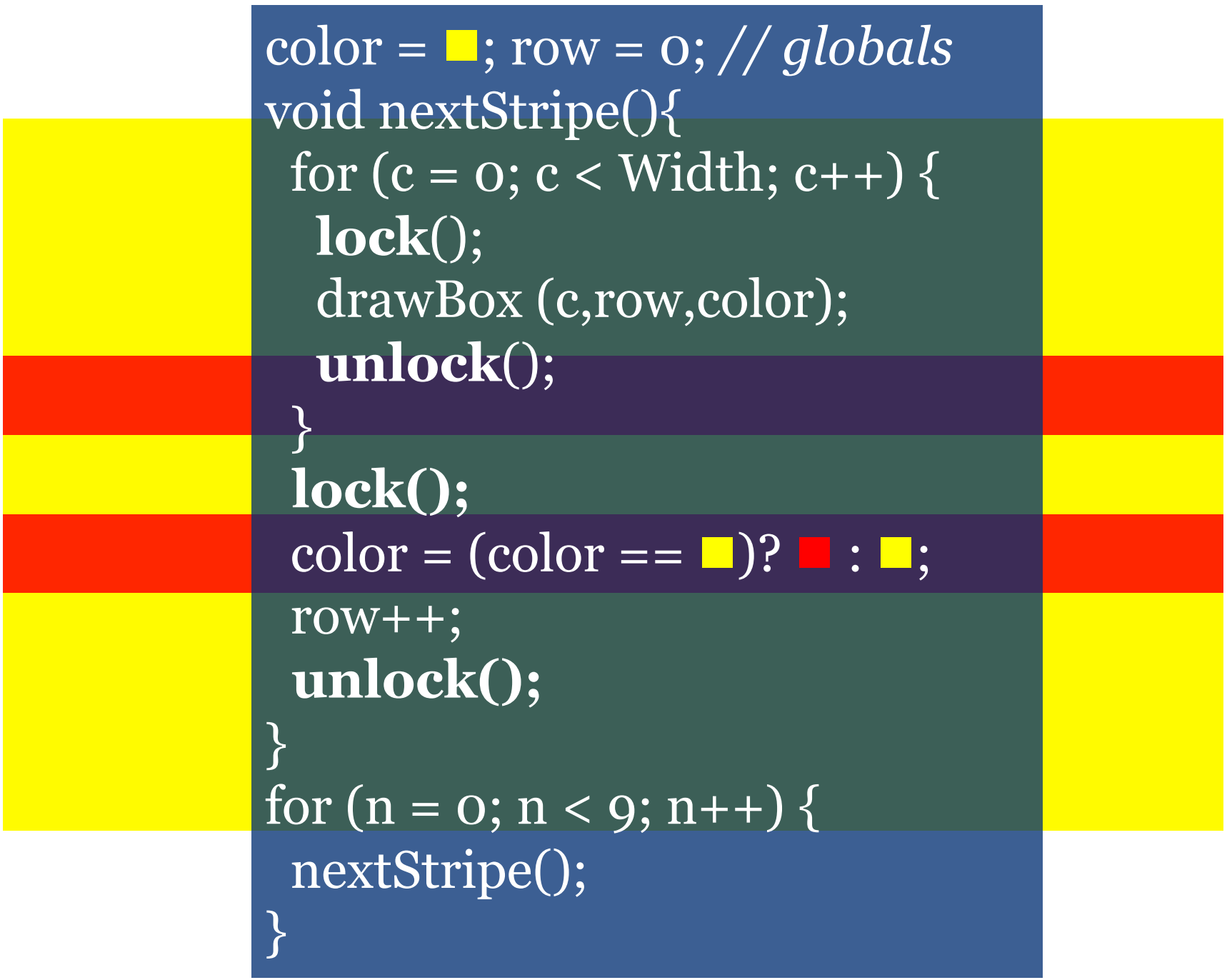

color = ■; row = 0; // globals
void nextStripe(){
for (c = 0; c < Width; c++) {
drawBox (c,row,color);
}
color = (color == ■)? ■ : ■;
row++;
}
for (n = 0; n < 9; n++) {
nextStripe();
}

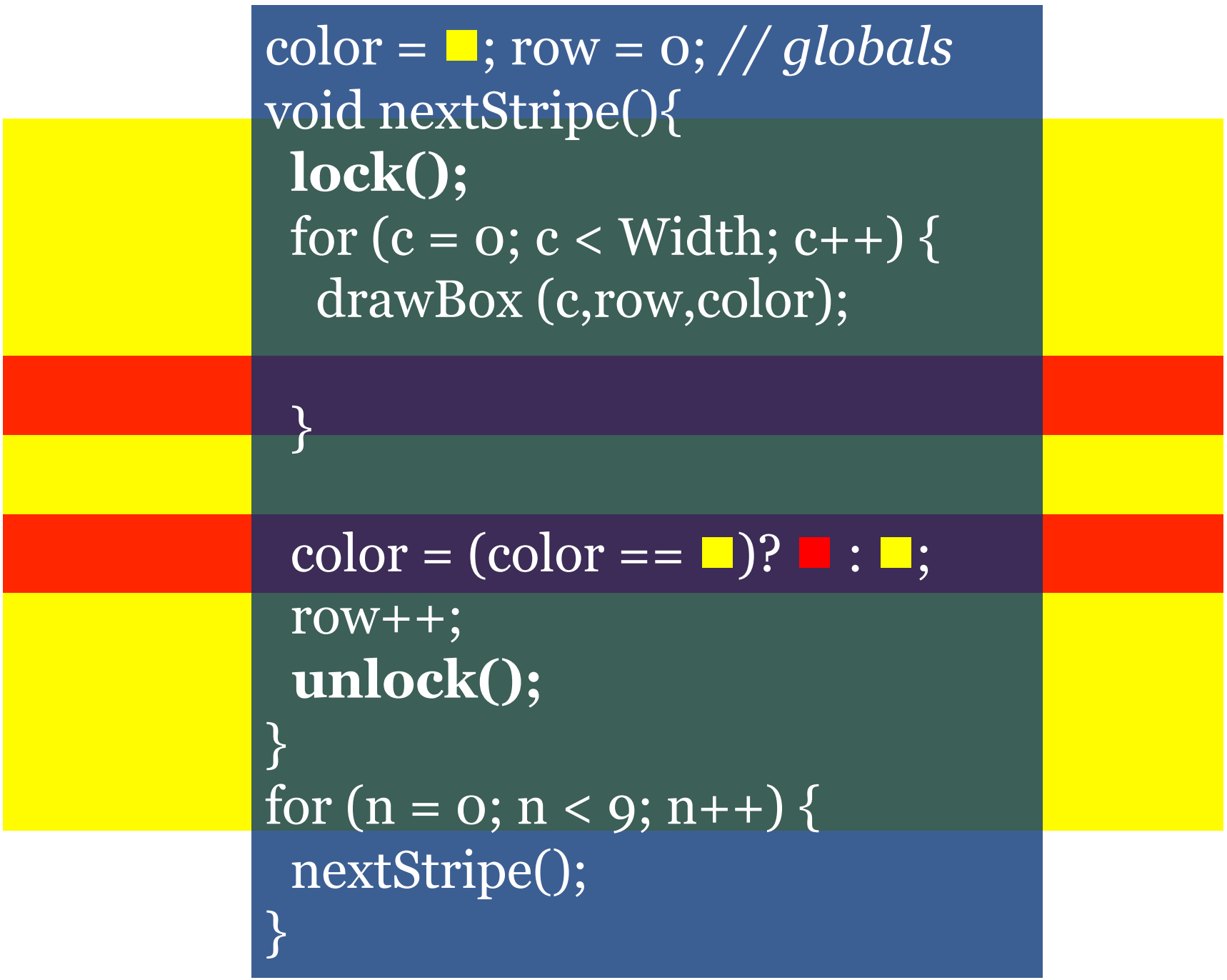

color = ■; row = 0; // globals
void nextStripe(){
for (c = 0; c < Width; c++) {
lock();
drawBox (c,row,color);
unlock();
}
lock();
color = (color == ■)? ■ : ■;
row++;
unlock();
}
for (n = 0; n < 9; n++) {
nextStripe();
}





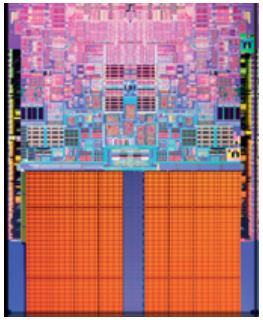

atomicity
violation

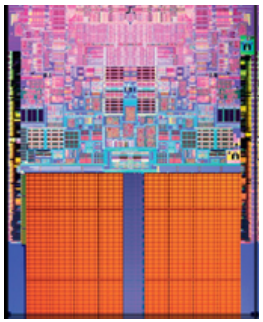

color = ■; row = 0; // globals
void nextStripe(){
for (c = 0; c < Width; c++) {
lock();
drawBox (c,row,color);
unlock();
}
lock();
color = (color == ■)? ■ : ■;
row++;
unlock();
}
for (n = 0; n < 9; n++) {
nextStripe();
}


color = ■; row = 0; // globals
void nextStripe(){
lock();
for (c = 0; c < Width; c++) {
drawBox (c,row,color);
}
color = (color == ■)? ■ : ■;
row++;
unlock();
}
for (n = 0; n < 9; n++) {
nextStripe();
}

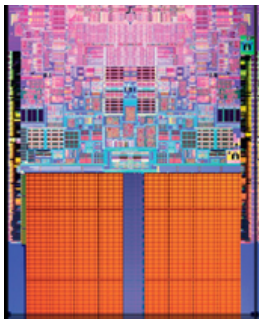

deadlock

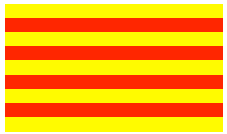

order violation



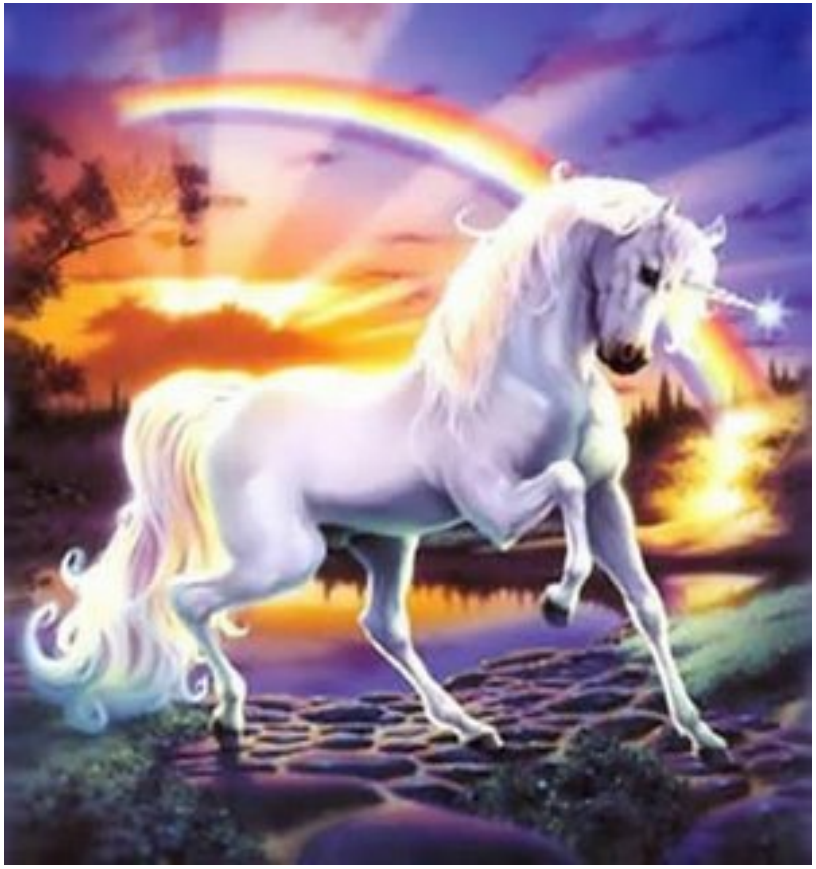


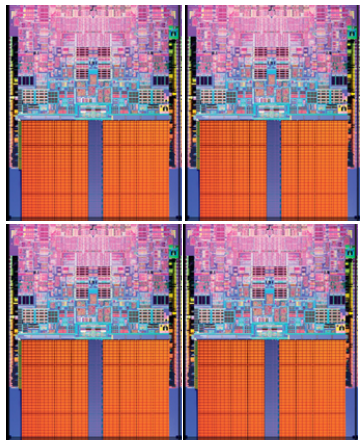
sequential

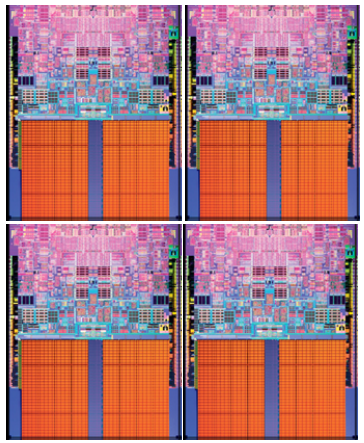




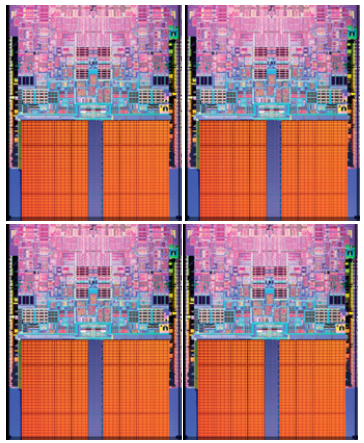
sequential

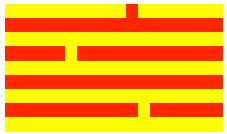






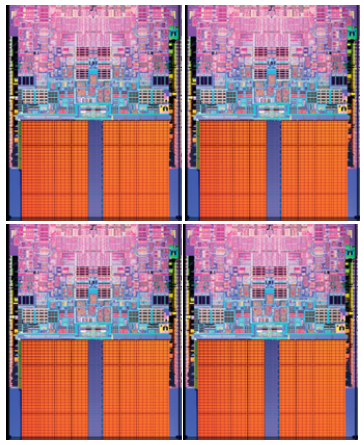
race conditions





race conditions

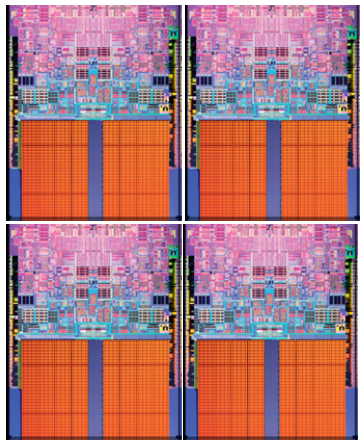
atomicity violations



race conditions

atomicity violations

deadlock

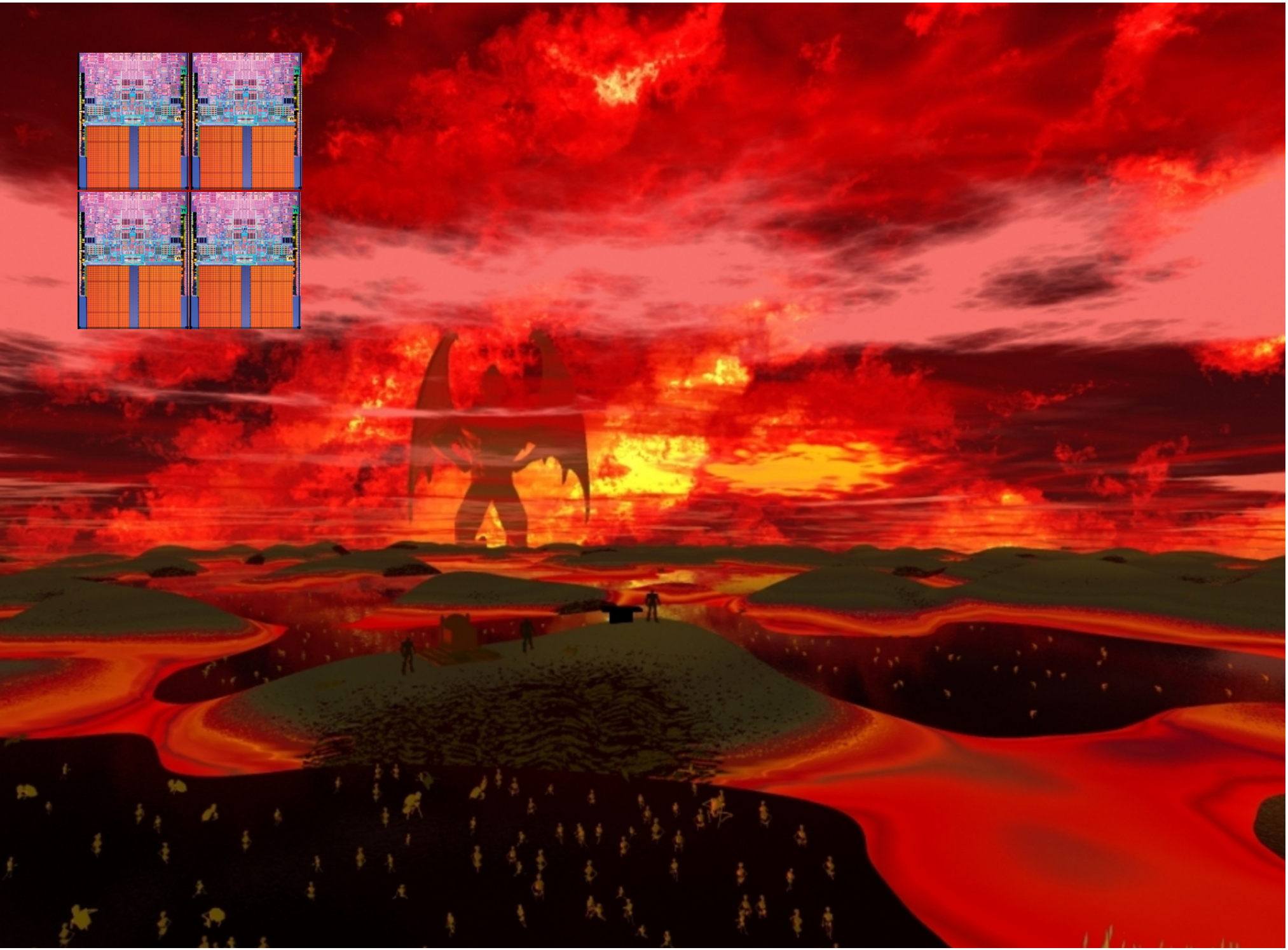


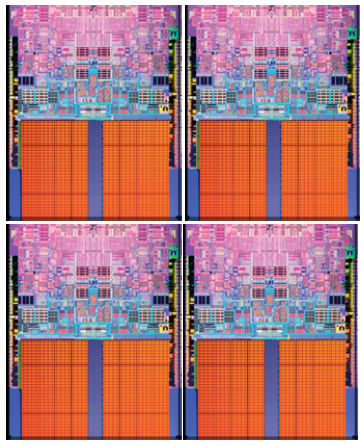
race conditions

atomicity violations

deadlock

order violations





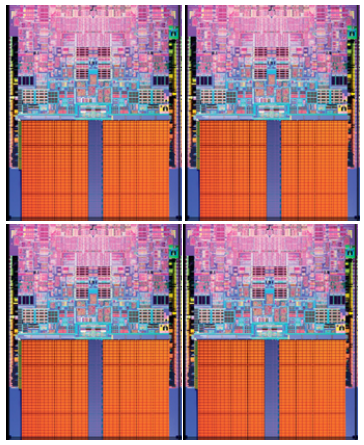
race conditions

atomicity violations

Grace

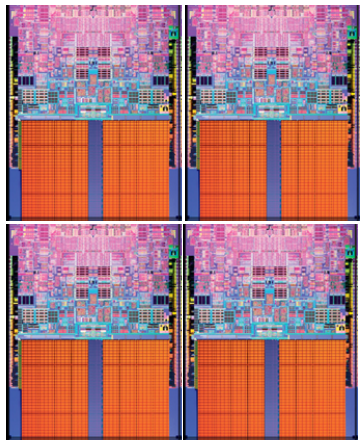
deadlock

order violations



Grace


race conditions
atomicity violations
deadlock
order violations



Grace

sequential

Grace

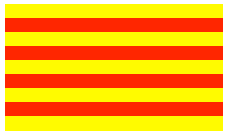
```
% g++ myprog.cpp -lpthread -o myprog
```

Grace

```
% g++ myprog.cpp -lgrace   -o myprog
```

Grace

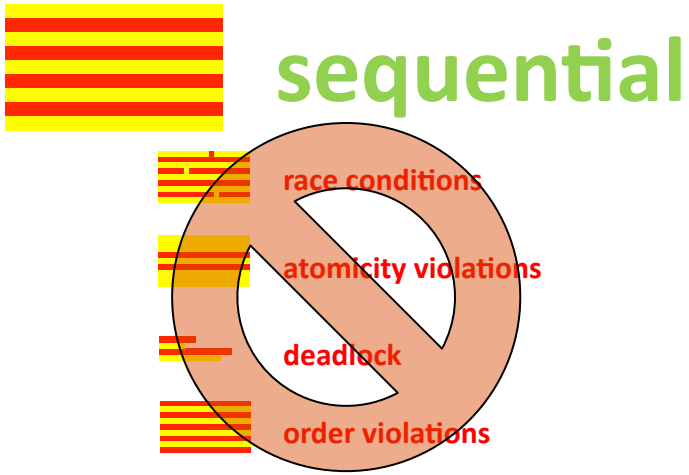
```
% g++ myprog.cpp -lgrace    -o myprog
```



sequential

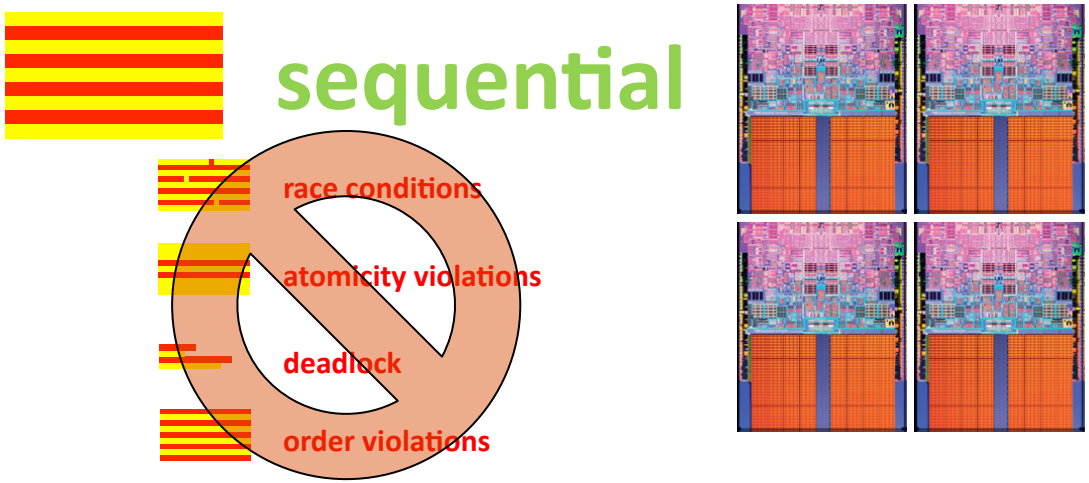
Grace

```
% g++ myprog.cpp -lgrace    -o myprog
```

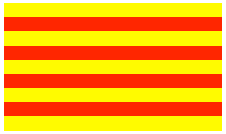
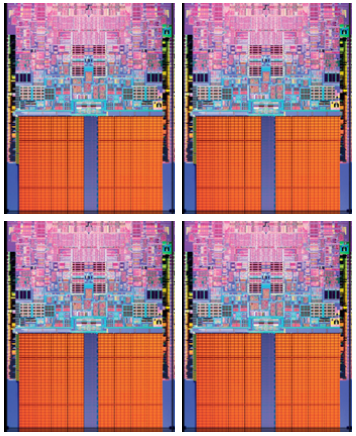


Grace

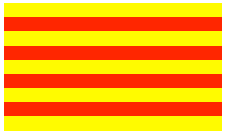
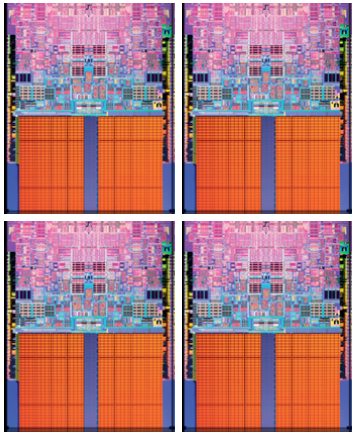
```
% g++ myprog.cpp -lgrace    -o myprog
```



Grace

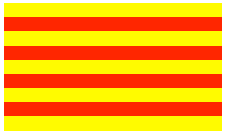
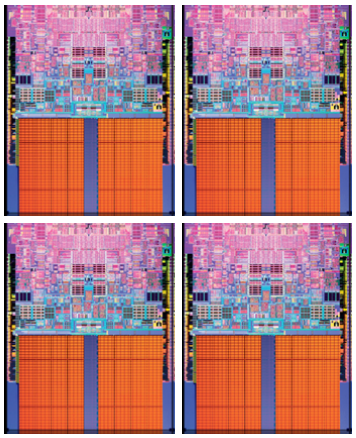
1.  sequential +  ?

Grace

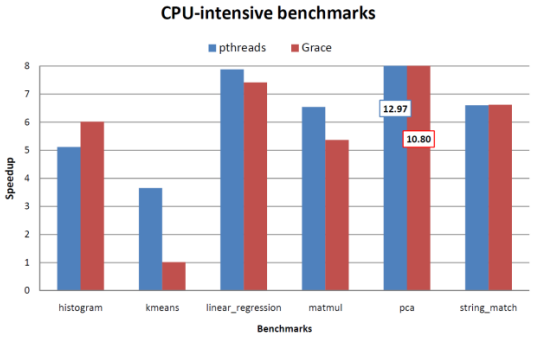
1.  sequential +  ?

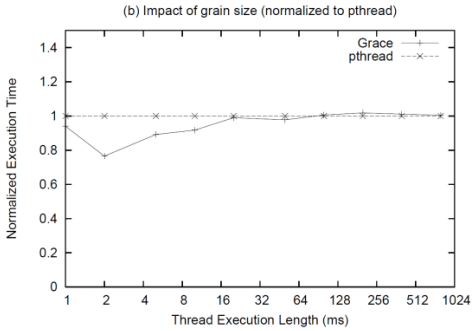
2. 

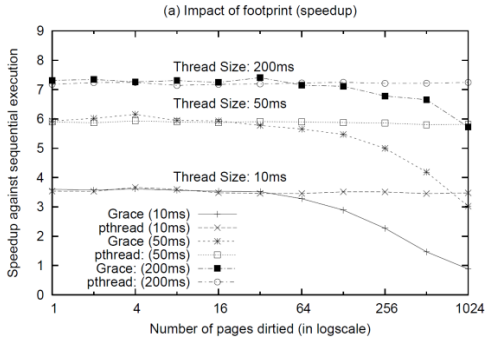
Grace

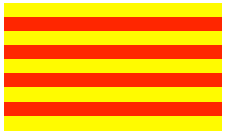
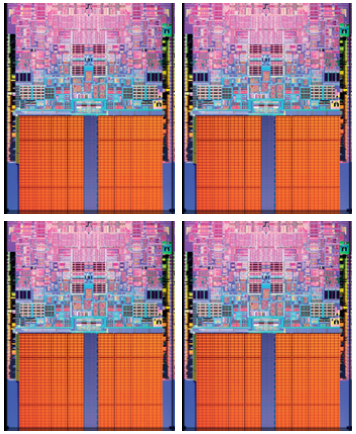
1.  sequential +  ?

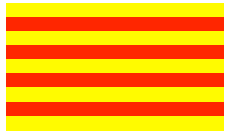
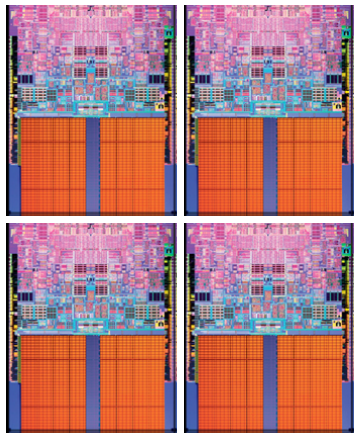
2. 

3. 

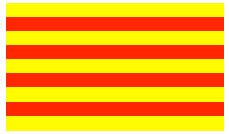
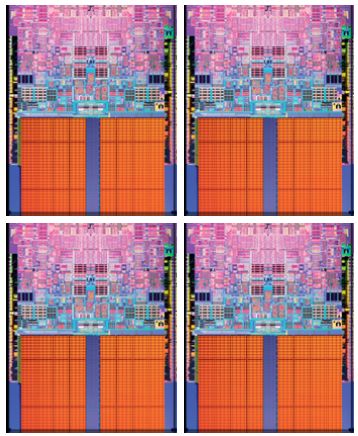




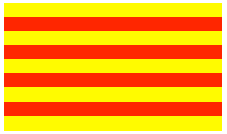
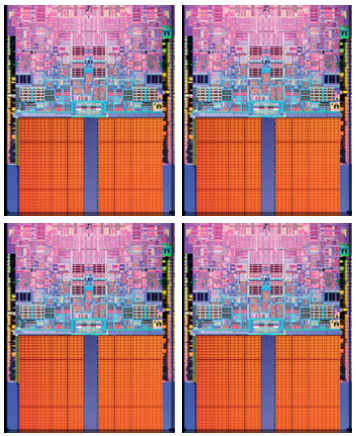
1.  sequential +  ?

1.  sequential +  ?

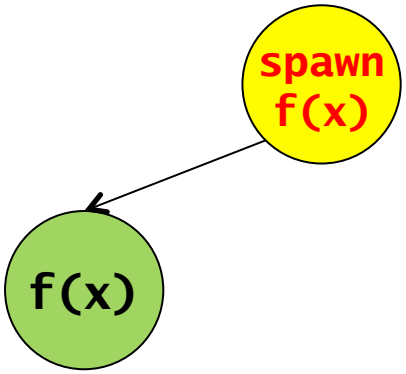
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```

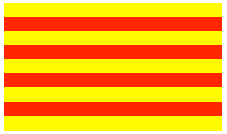
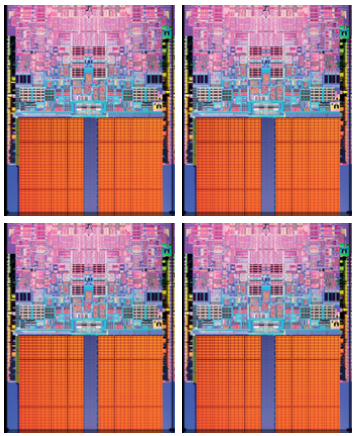
1.  sequential +  ?


```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```

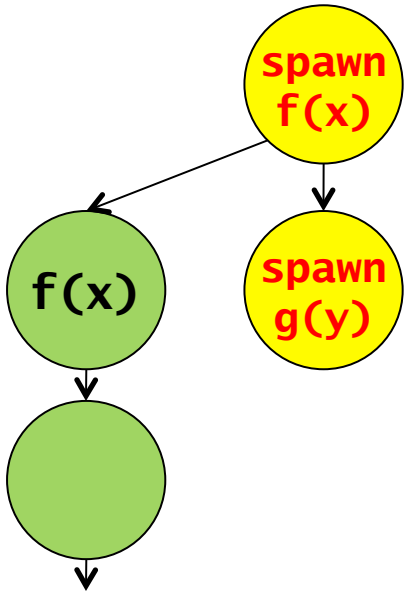
1.  sequential +  ?

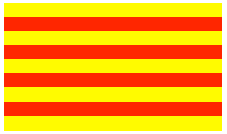
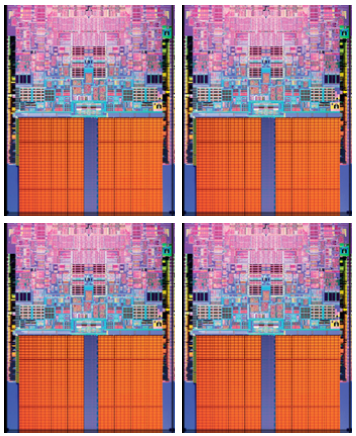
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



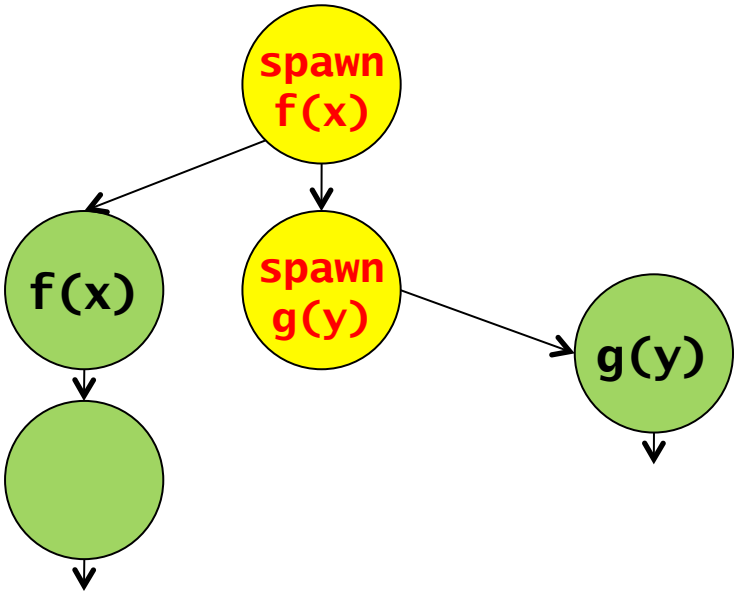
1.  sequential +  ?

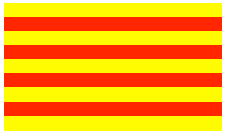
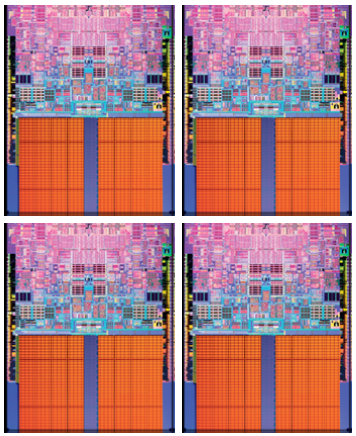
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



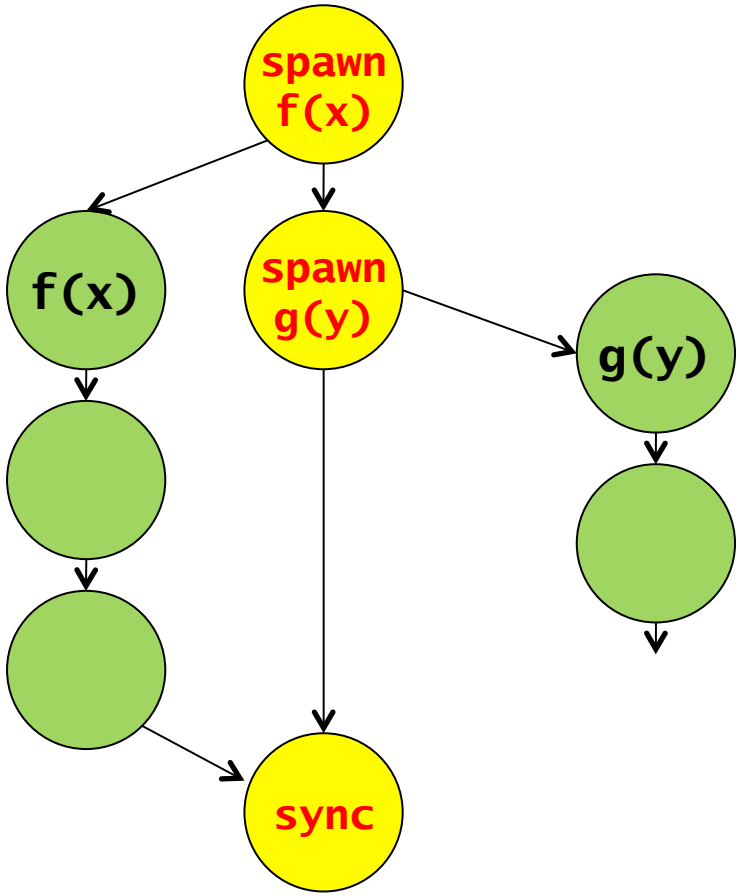
1.  sequential +  ?

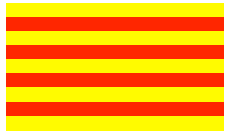
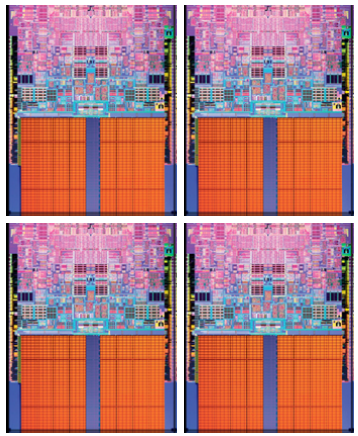
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



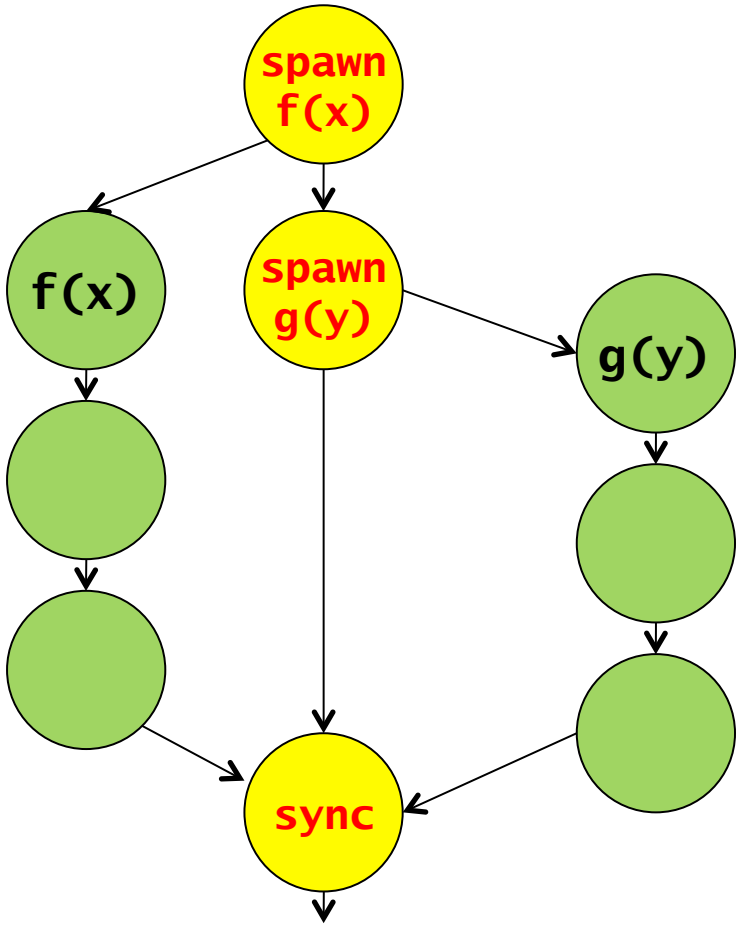
1.  sequential +  ?

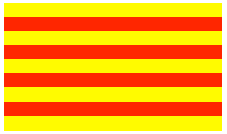
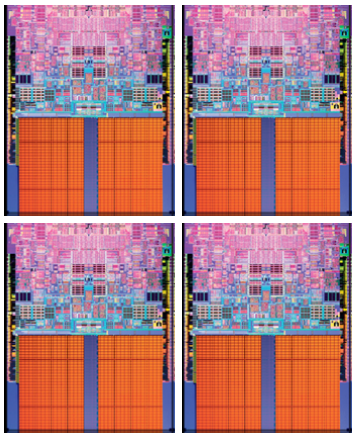
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



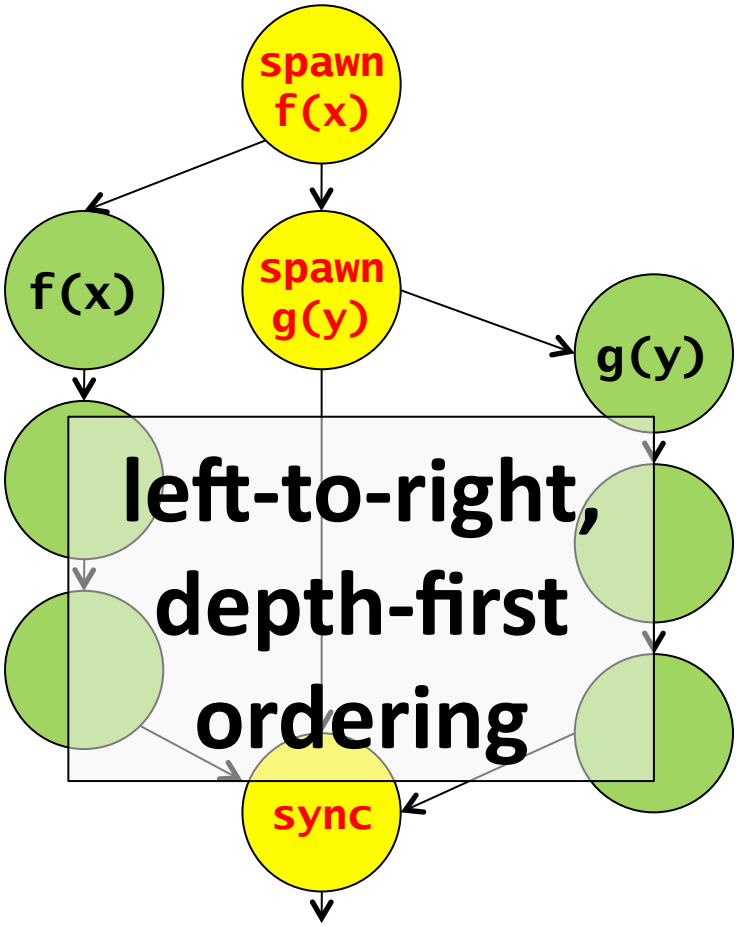
1.  sequential +  ?

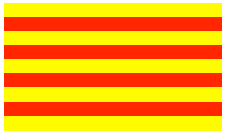
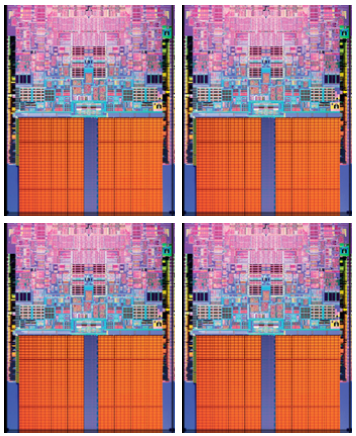
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



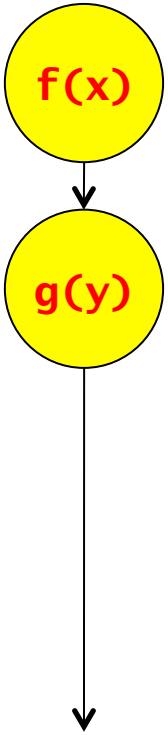
1.  sequential +  ?

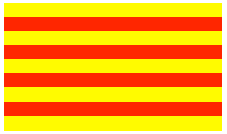
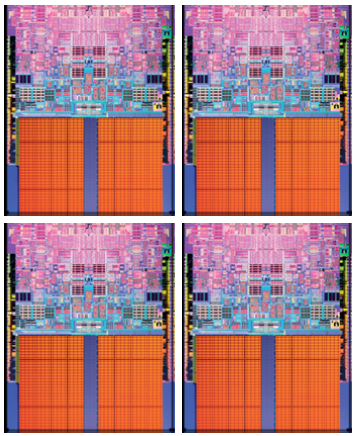
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



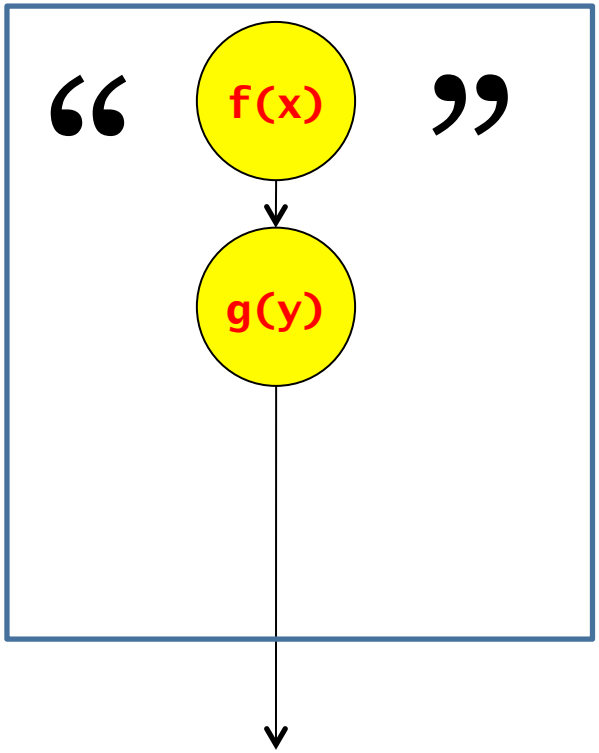
1.  sequential +  ?

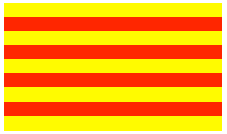
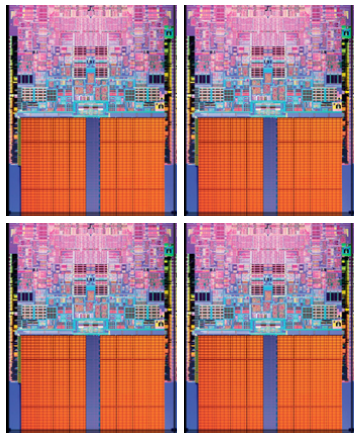
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```

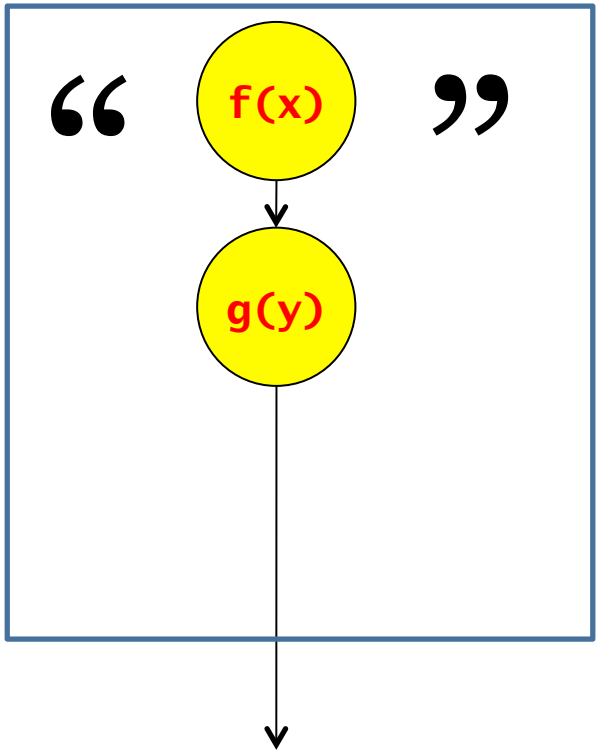


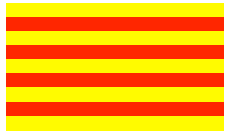
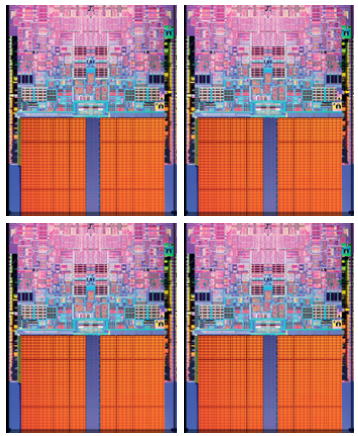
1.  sequential +  ?

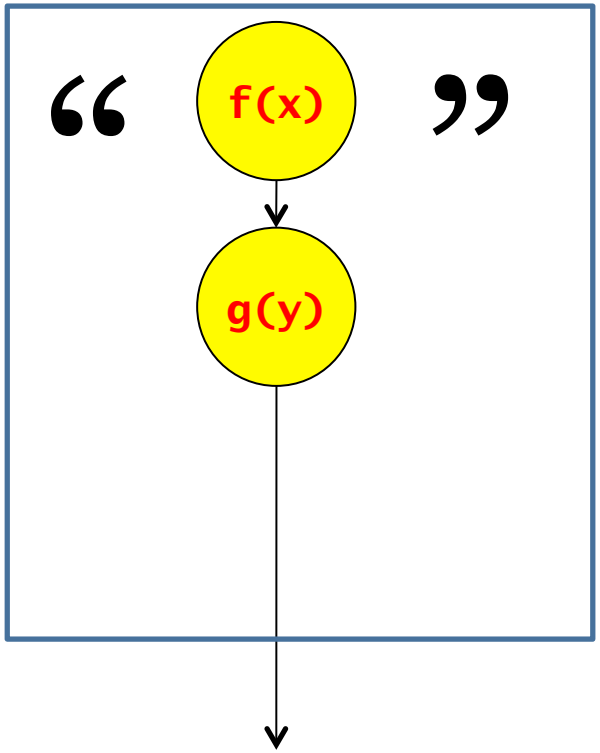
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



1.  sequential +  ?



1.  sequential +  ?

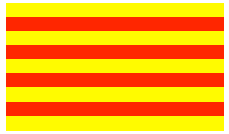
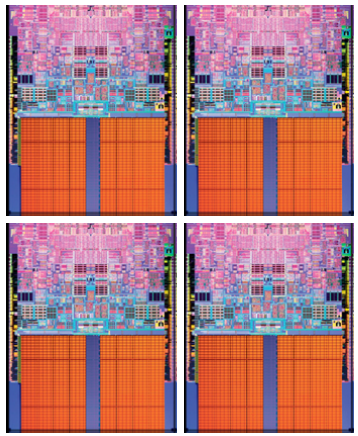


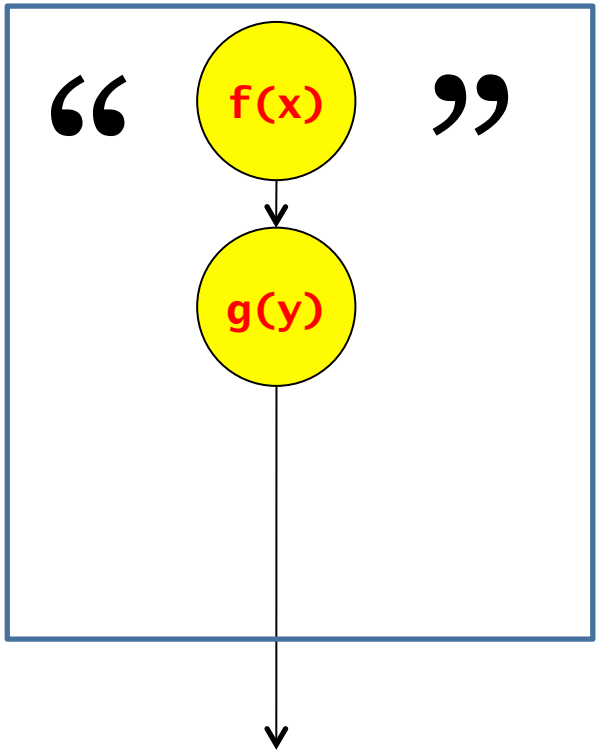
 race conditions

 atomicity violations

 deadlock

 order violations

1.  sequential +  ?

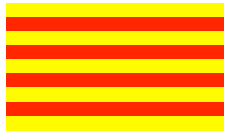
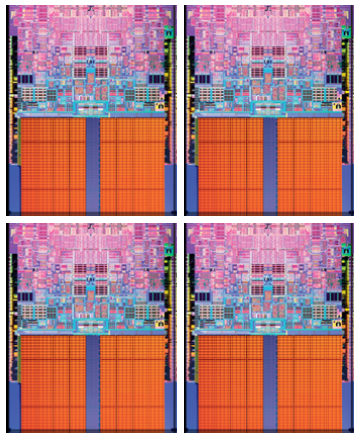


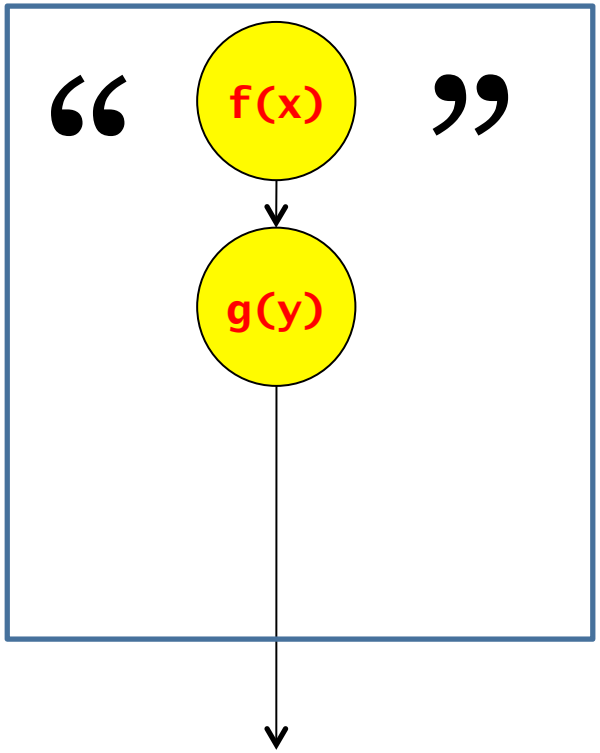
 ~~race conditions~~

 atomicity violations

 deadlock

 order violations

1.  sequential +  ?

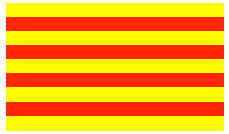
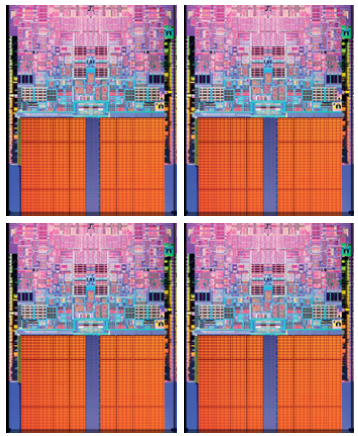


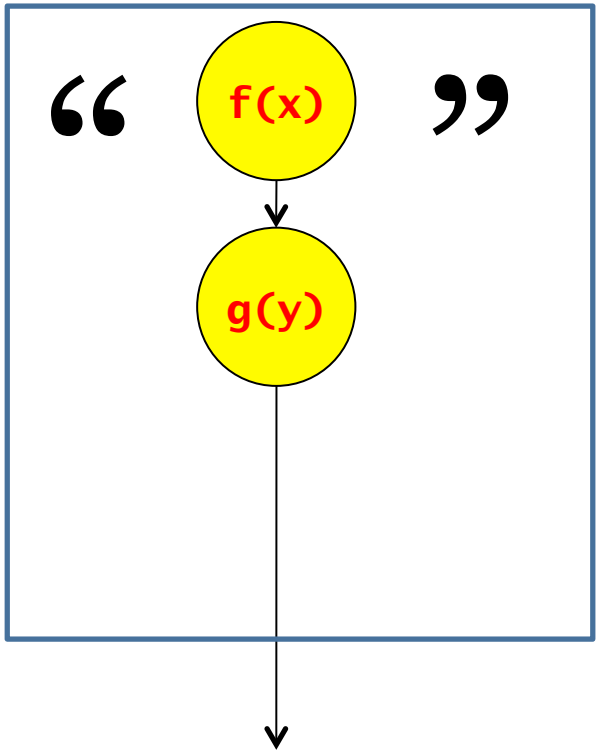
 ~~race conditions~~

 ~~atomicity violations~~

 deadlock

 order violations

1.  sequential +  ?

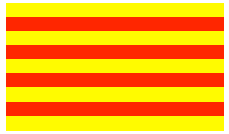
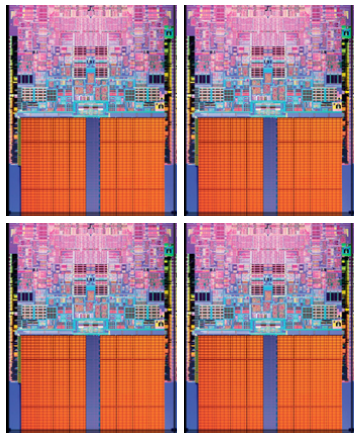


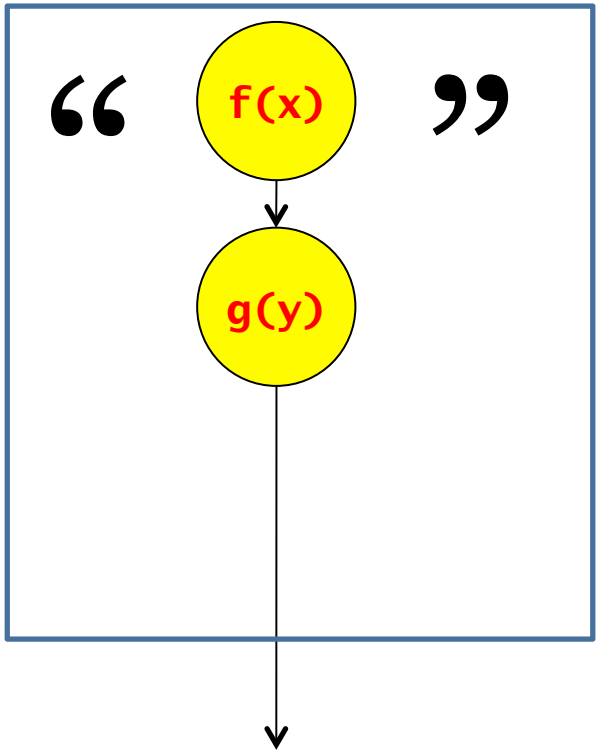
 ~~race conditions~~

 ~~atomicity violations~~

 deadlock

 order violations

1.  sequential +  ?

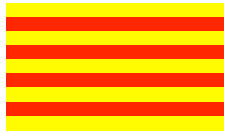
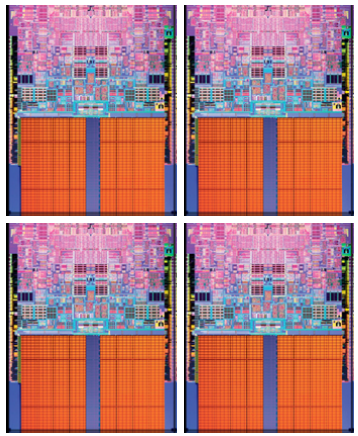


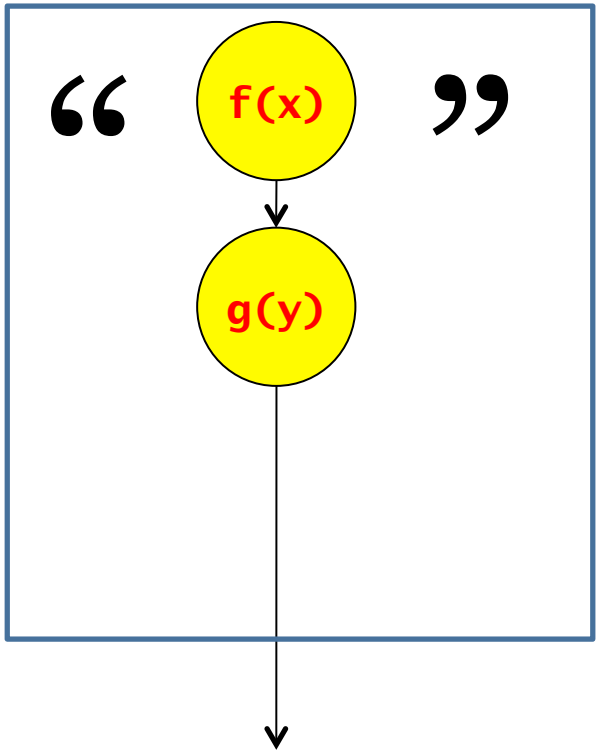
 ~~race conditions~~

 ~~atomicity violations~~

 deadlock

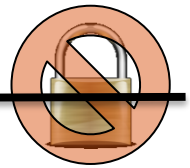
 order violations

1.  sequential +  ?

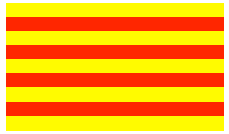
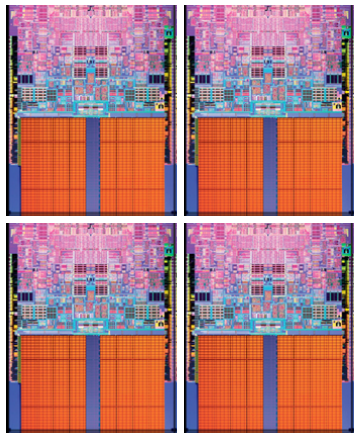


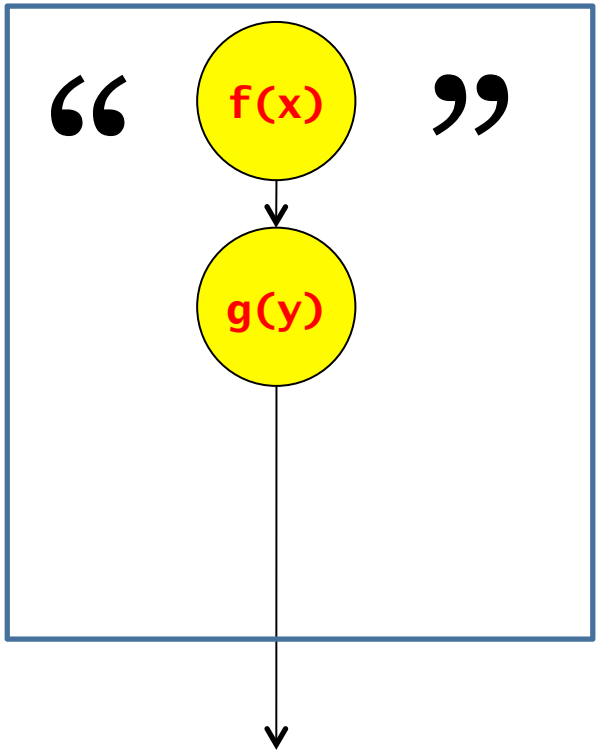
 ~~race conditions~~

 ~~atomicity violations~~

 ~~deadlock~~ 

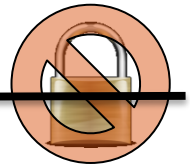
 order violations

1.  sequential +  ?

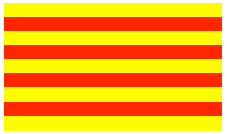
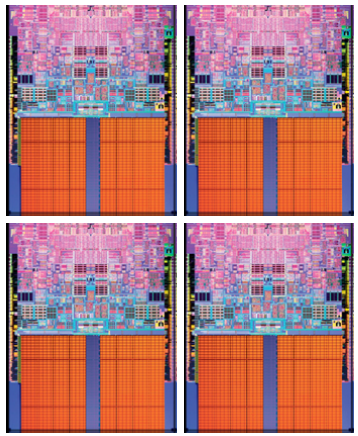


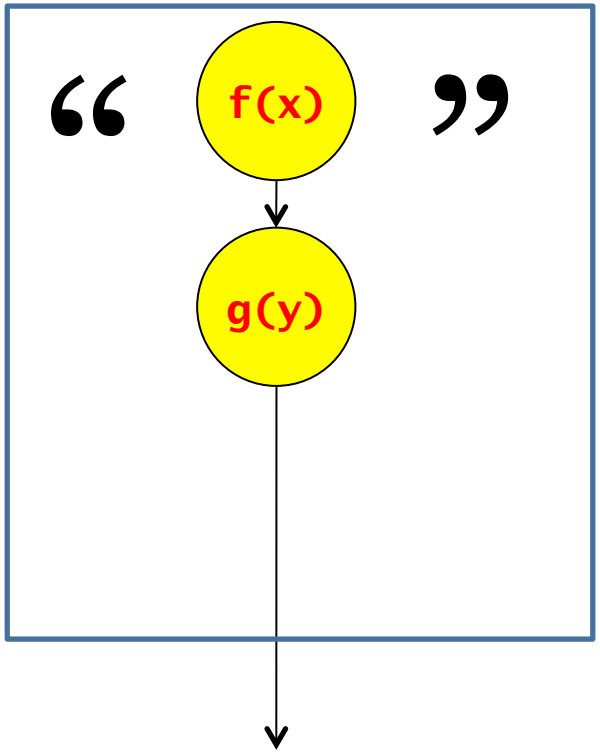
 race conditions

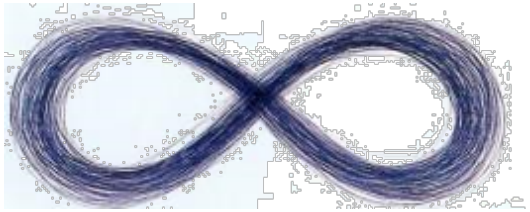
 atomicity violations

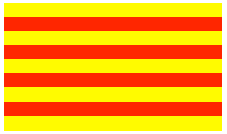
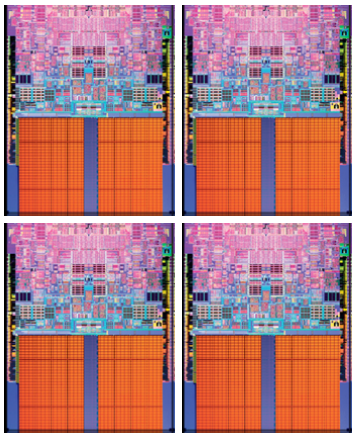
 deadlock 

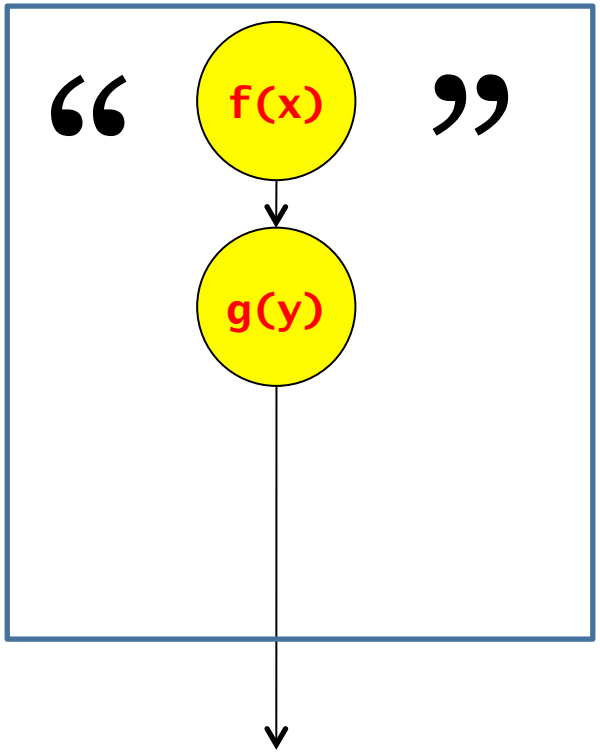
 order violations

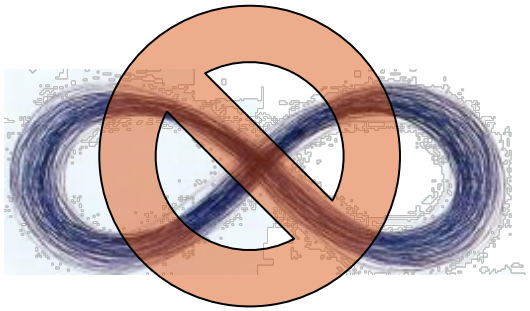
1.  sequential +  ?

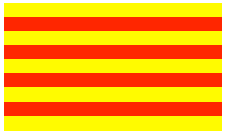
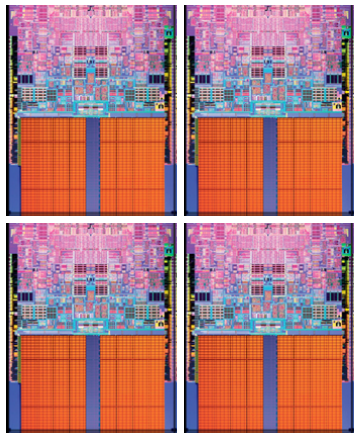


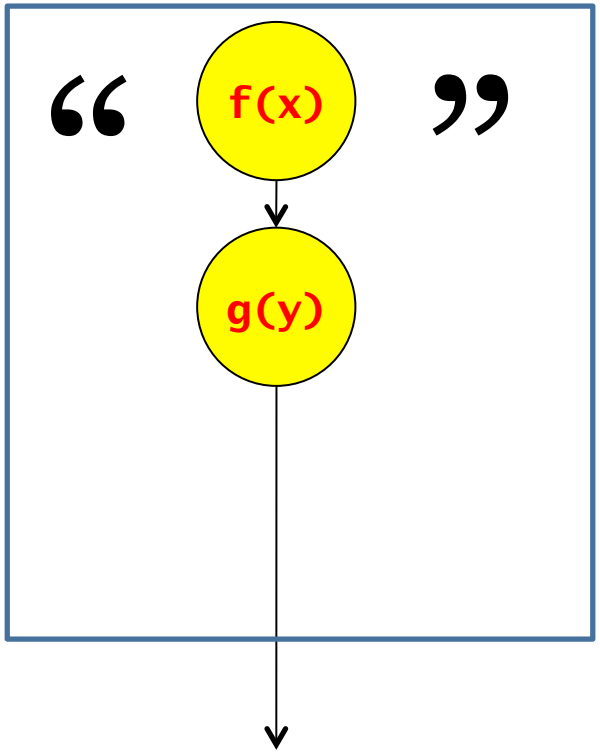


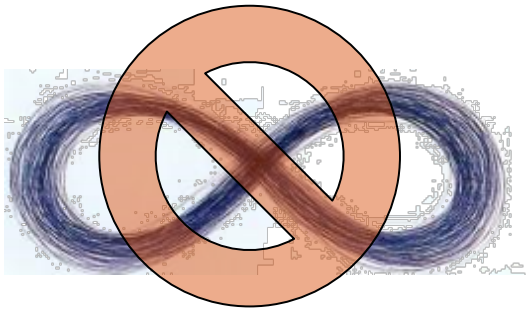
1.  sequential +  ?

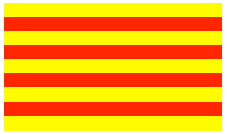
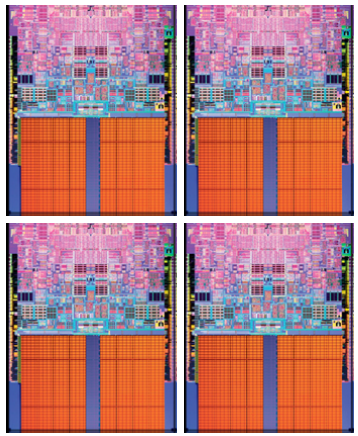


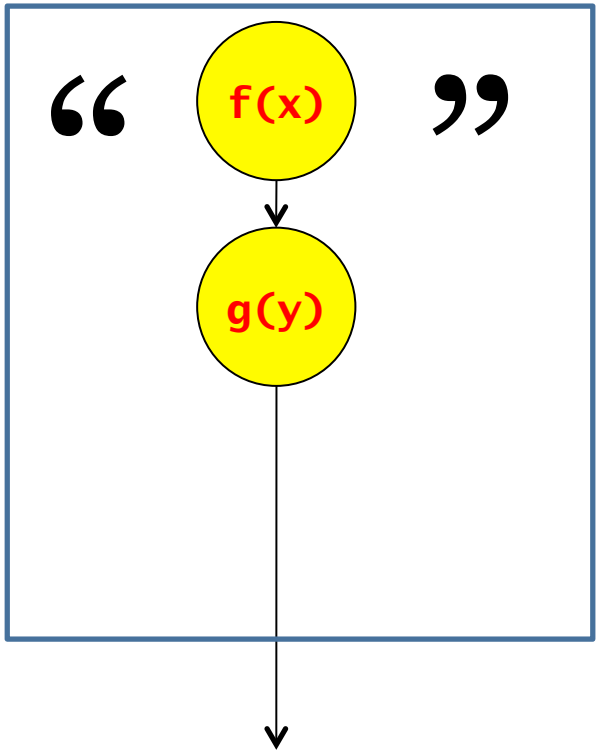


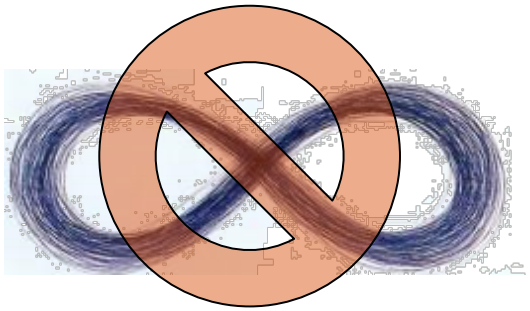
1.  sequential +  ?

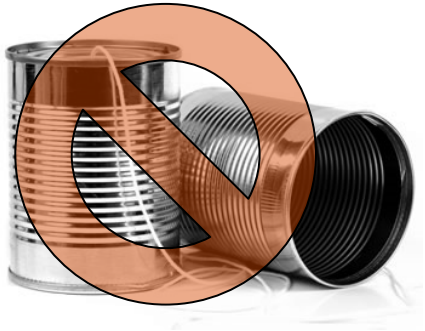


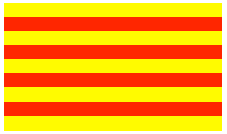
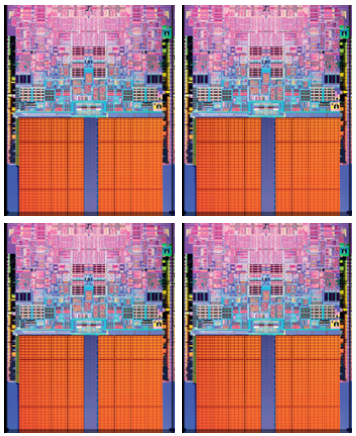


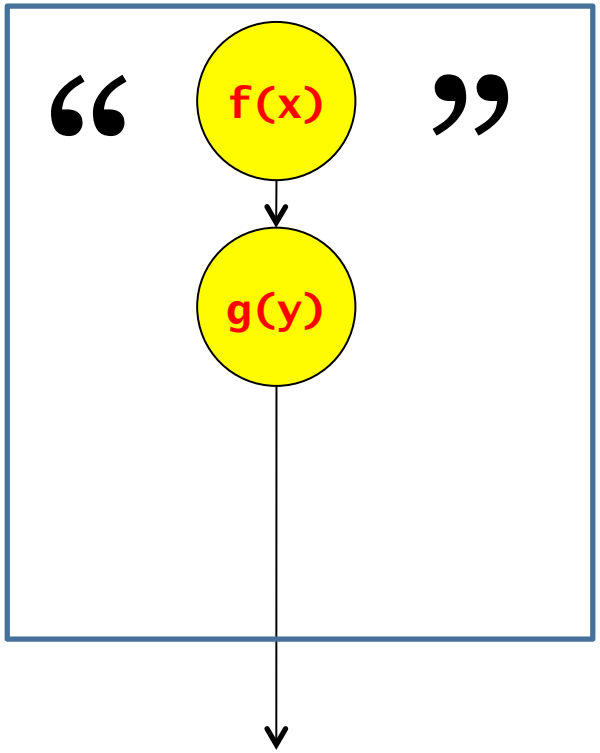
1.  sequential +  ?

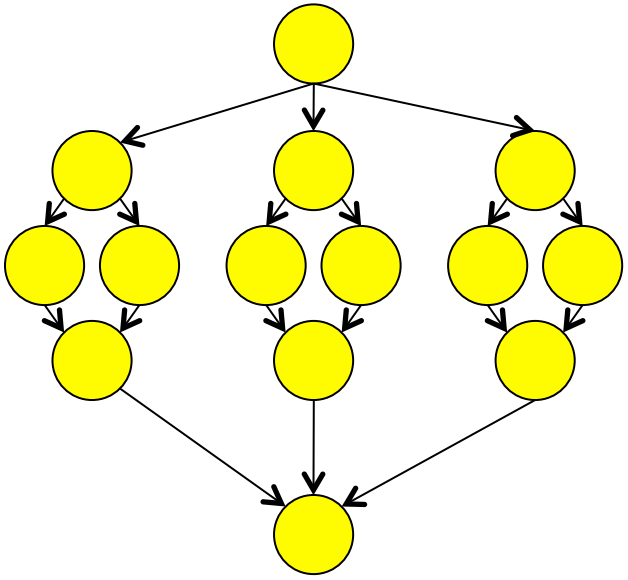


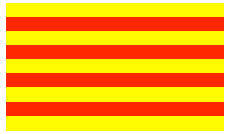
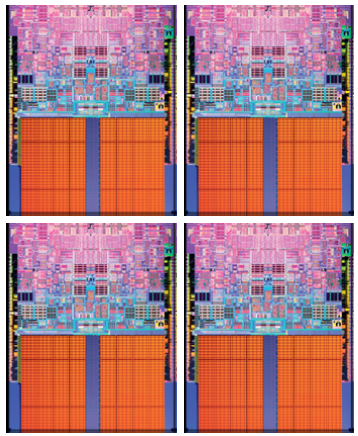


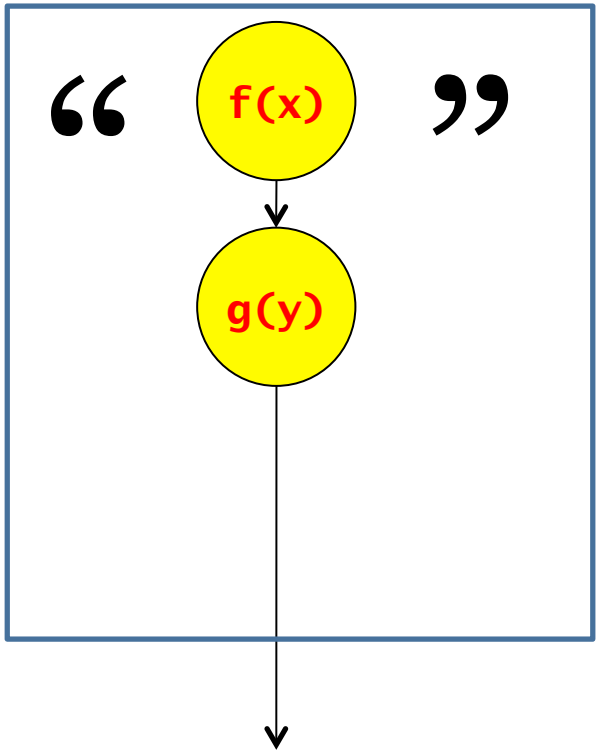


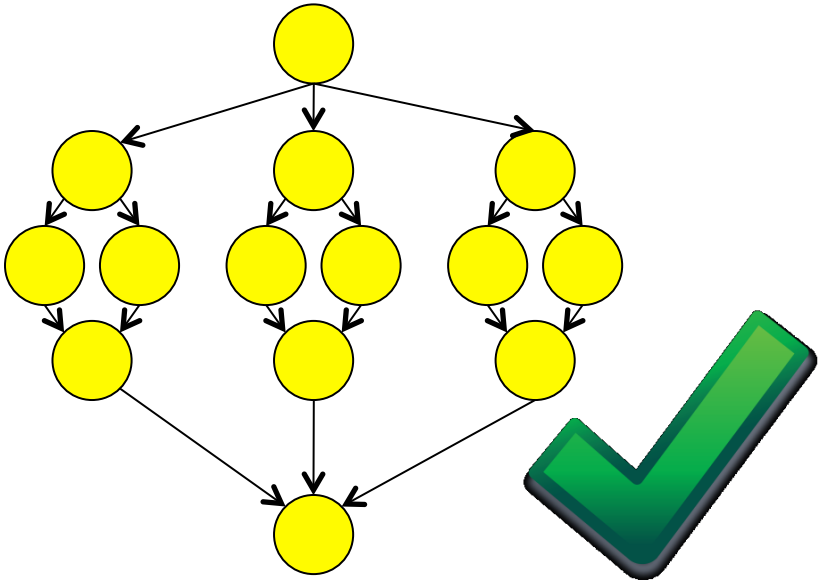
1.  sequential +  ?





1.  sequential +  ?



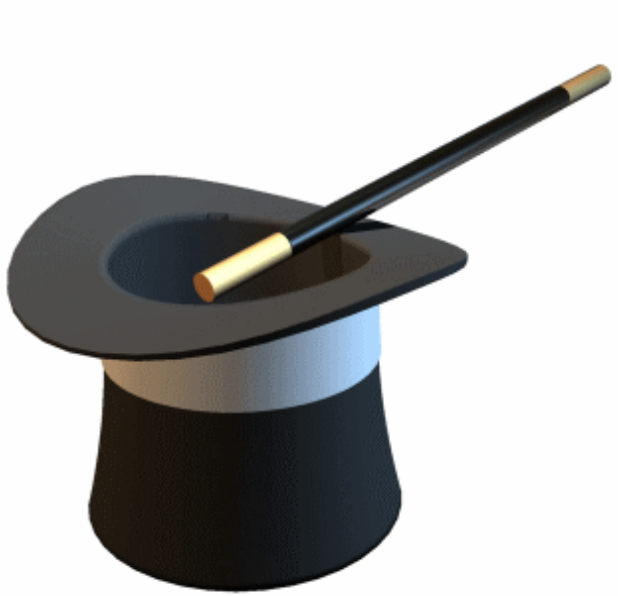


Fork-join
(e.g., map-reduce, Cilk, TBB…)

2.  ?

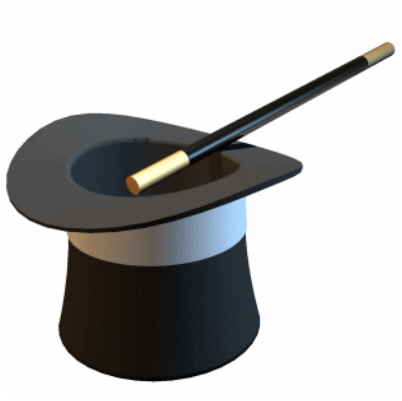
What's under the hood?

2.  **?**

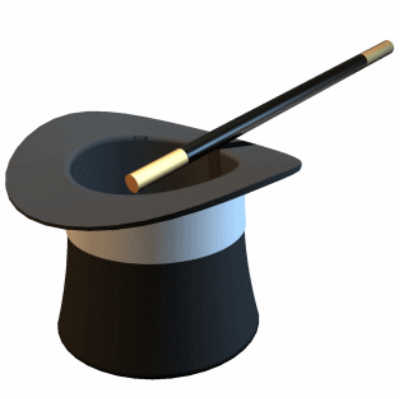


“Magic!”

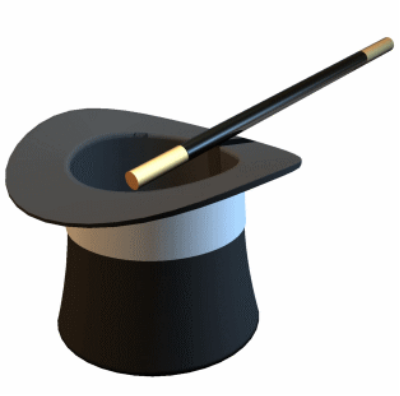
2.  ?

 + 

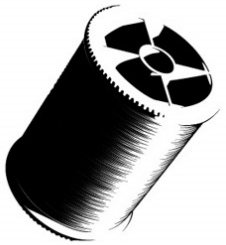
2.  **?**



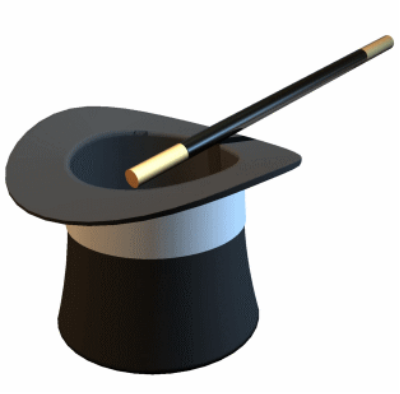
2.  **?**



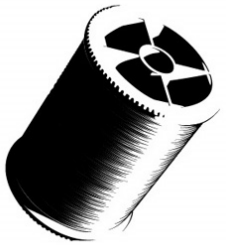
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```



2.  **?**



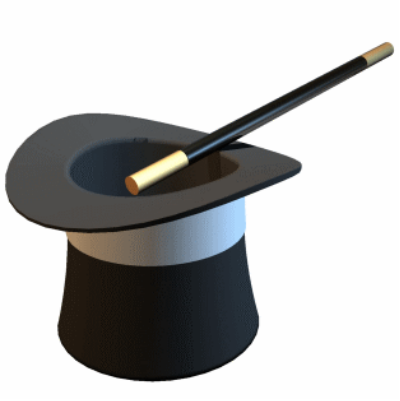
```
t1 = spawn f(x);
t2 = spawn g(y);
sync;
```

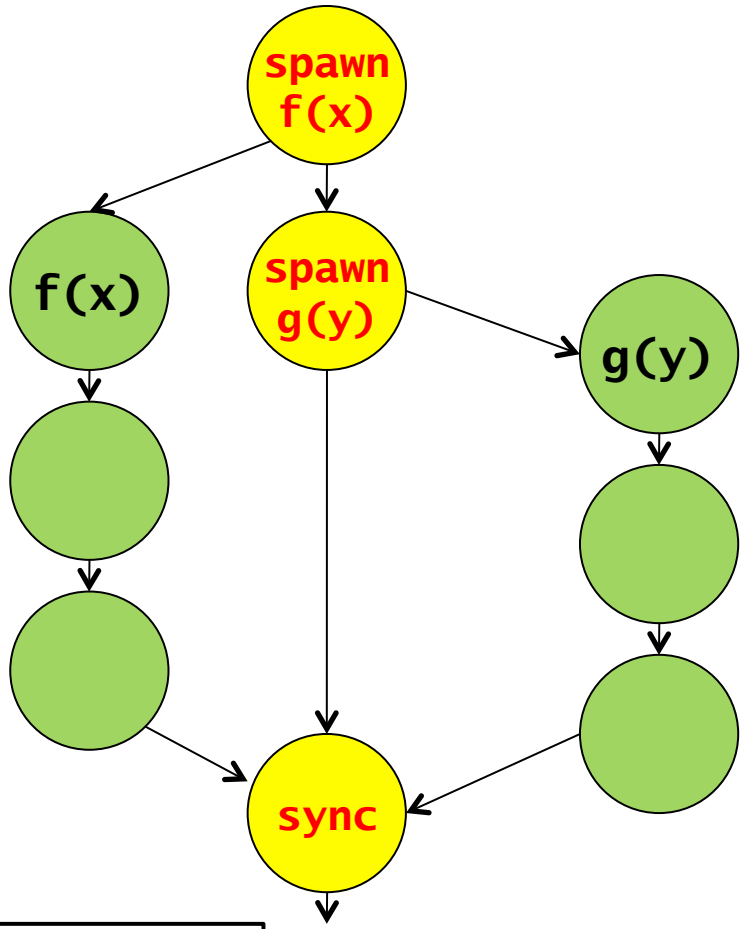


```
if (!fork())
  f(x);
if (!fork())
  g(y);
// check;
```

2.

?

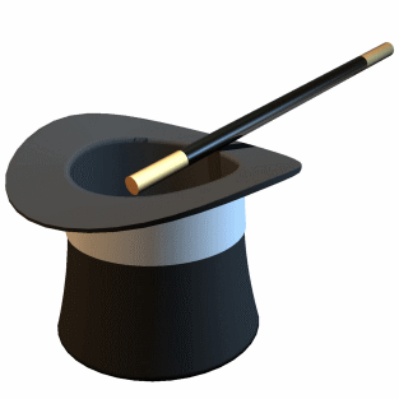


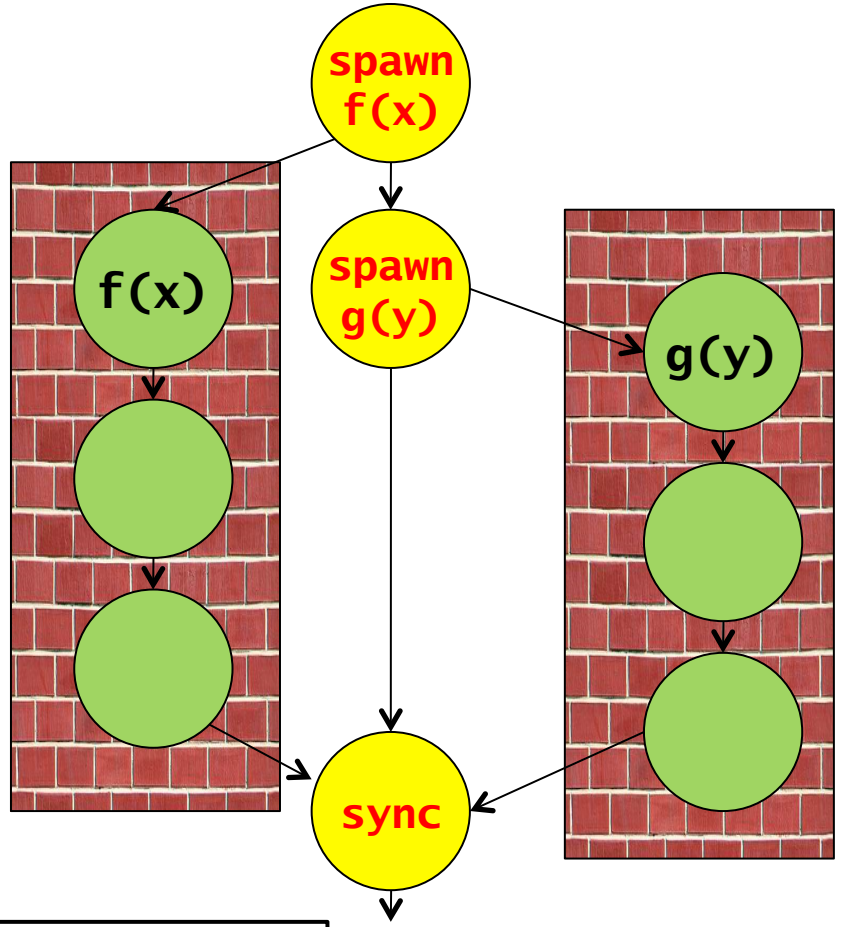



```
if (!fork())
  f(x);
if (!fork())
  g(y);
// check;
```

2.

?

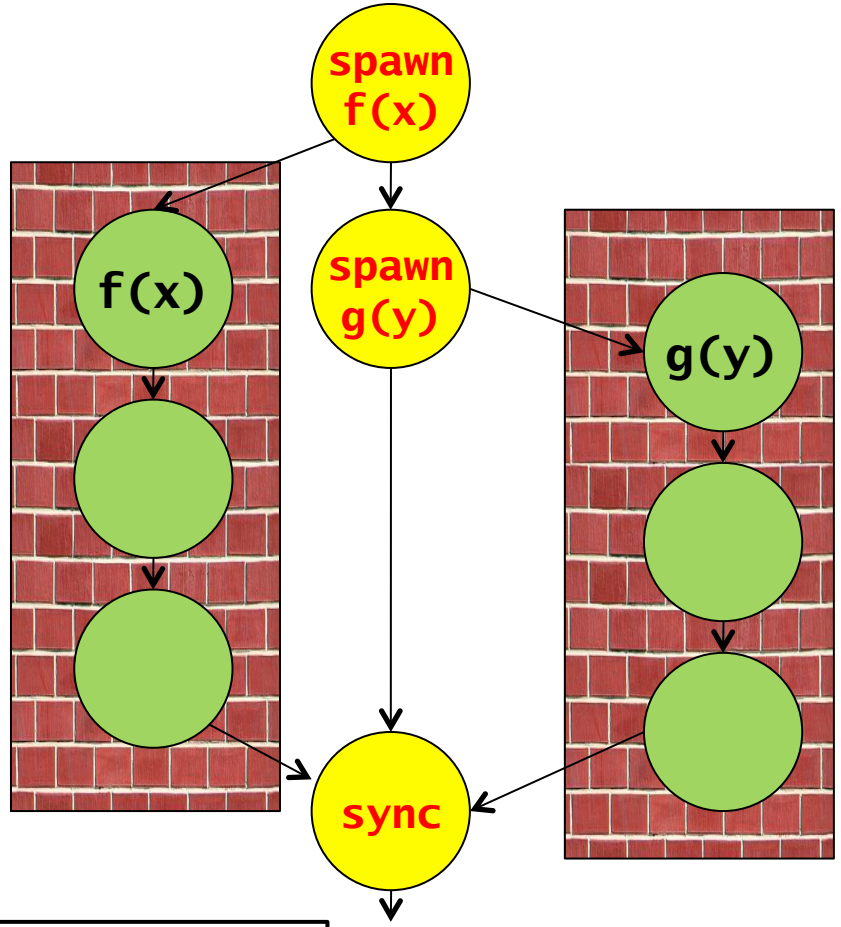


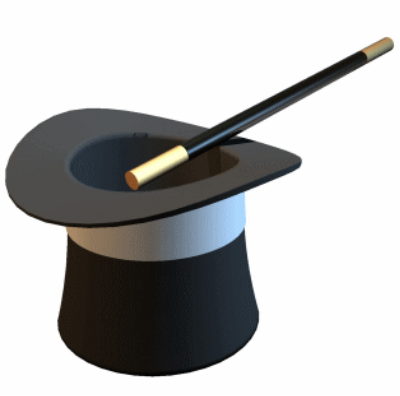



```
if (!fork())
  f(x);
if (!fork())
  g(y);
// check;
```

2.

?





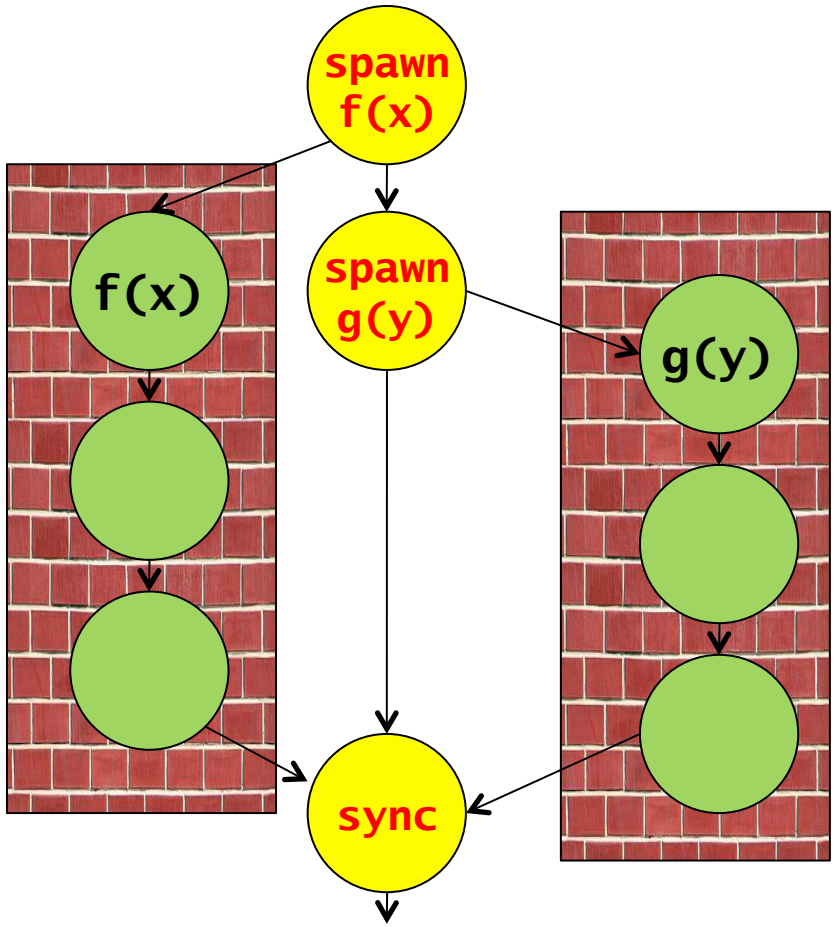
Note:
Nested threads ok

```
if (!fork())
   f(x);
if (!fork())
   g(y);
// check;
```

2.

?

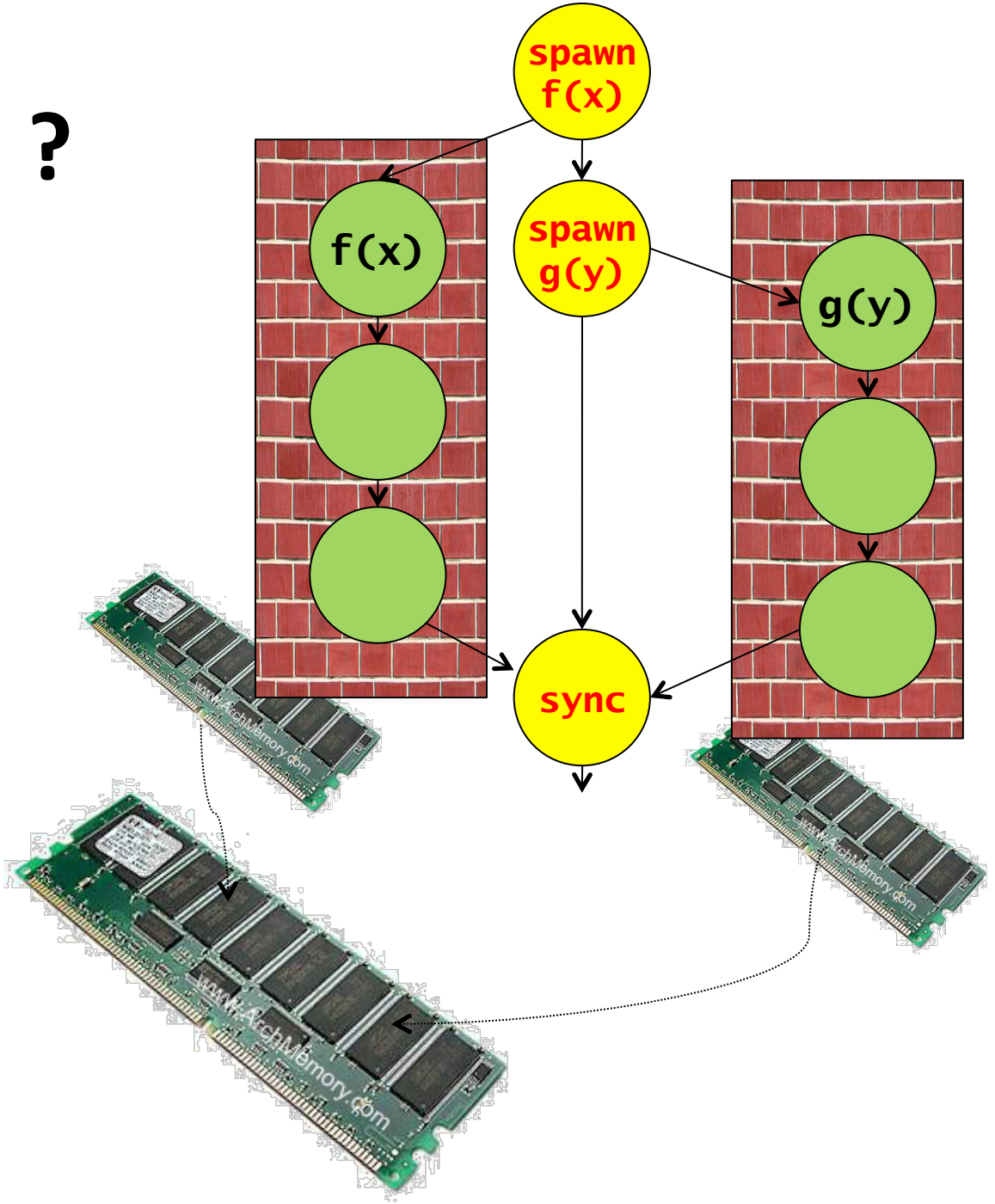
mmap



2.

?

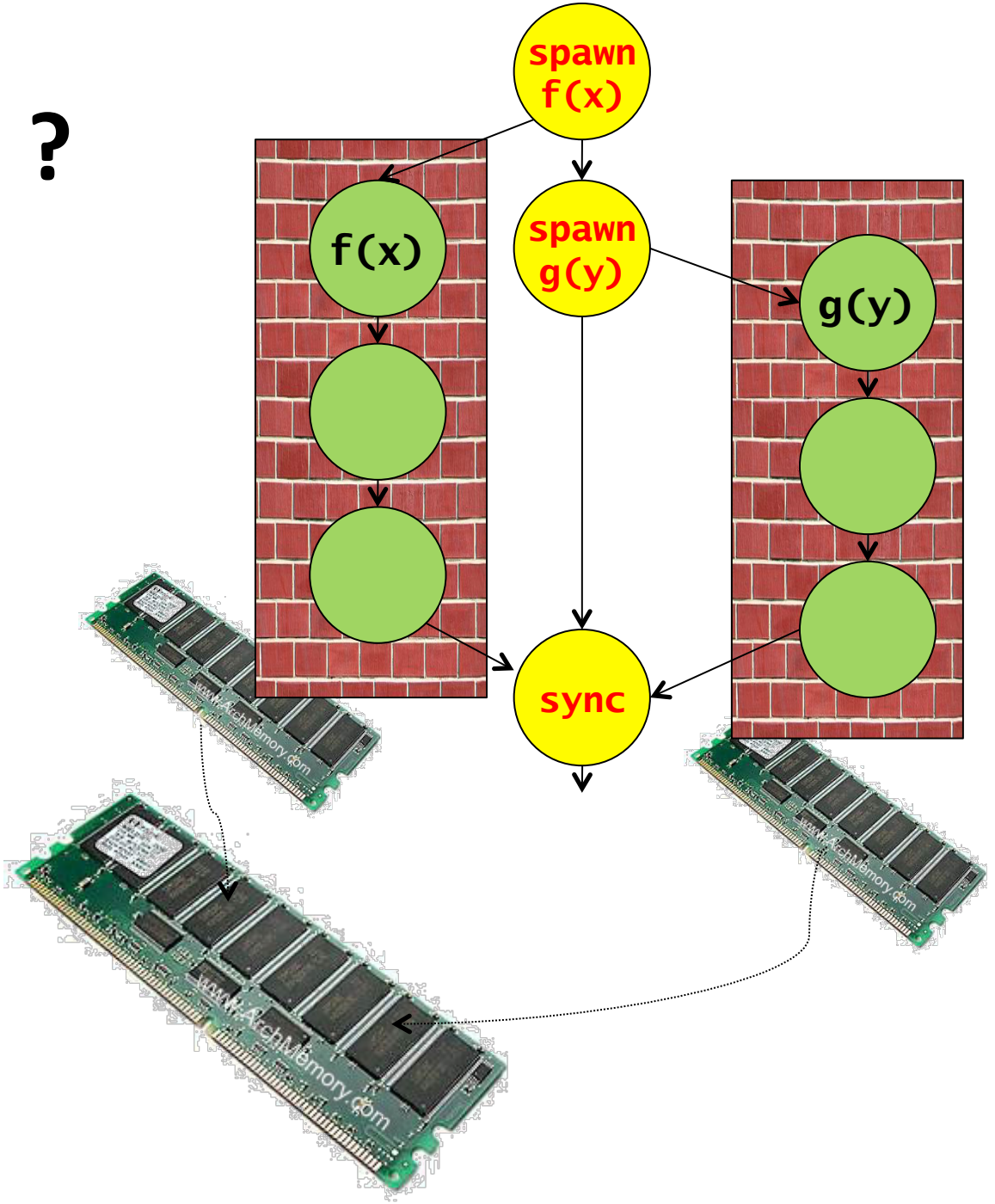
mmap



2.

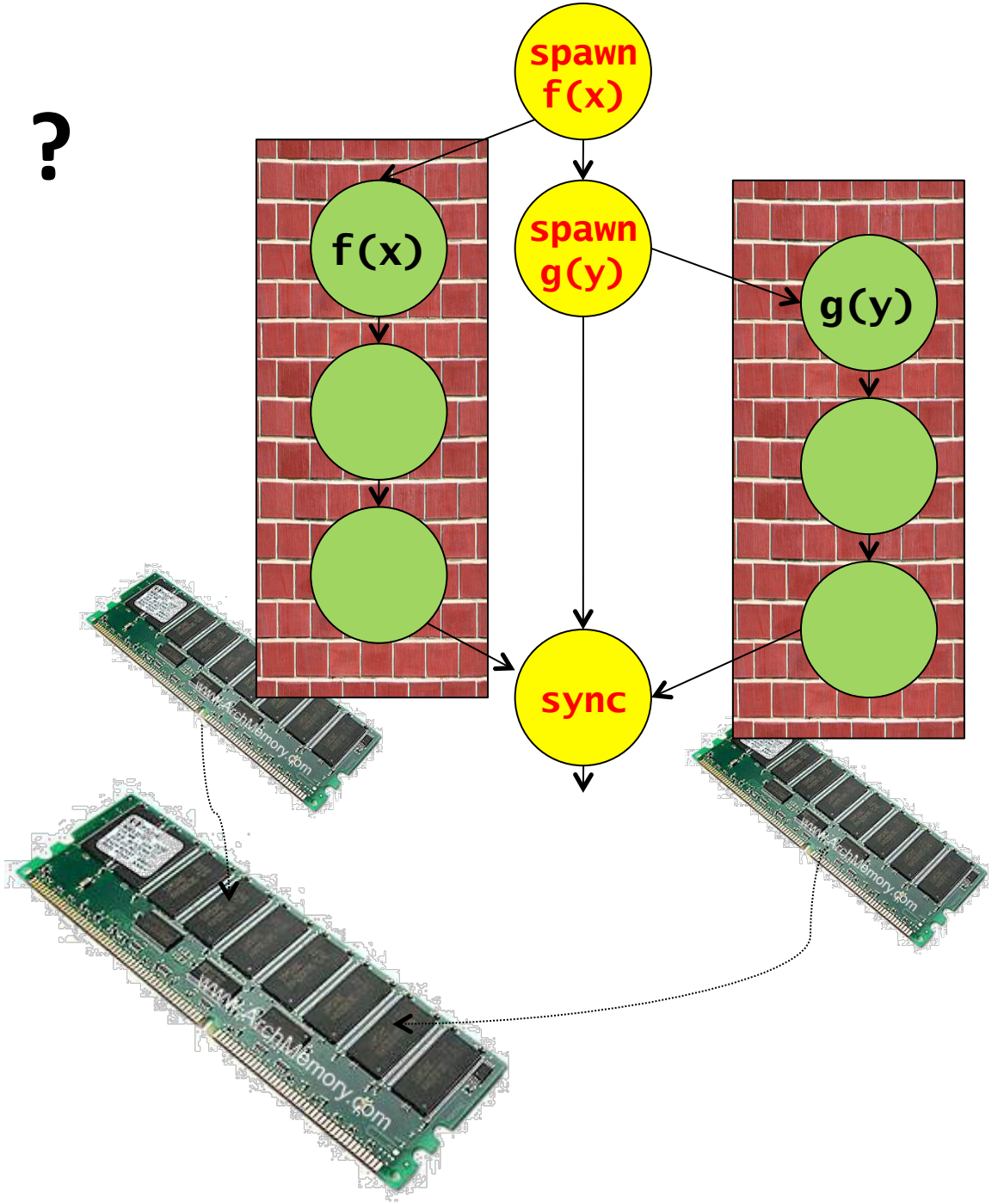
?

mmap



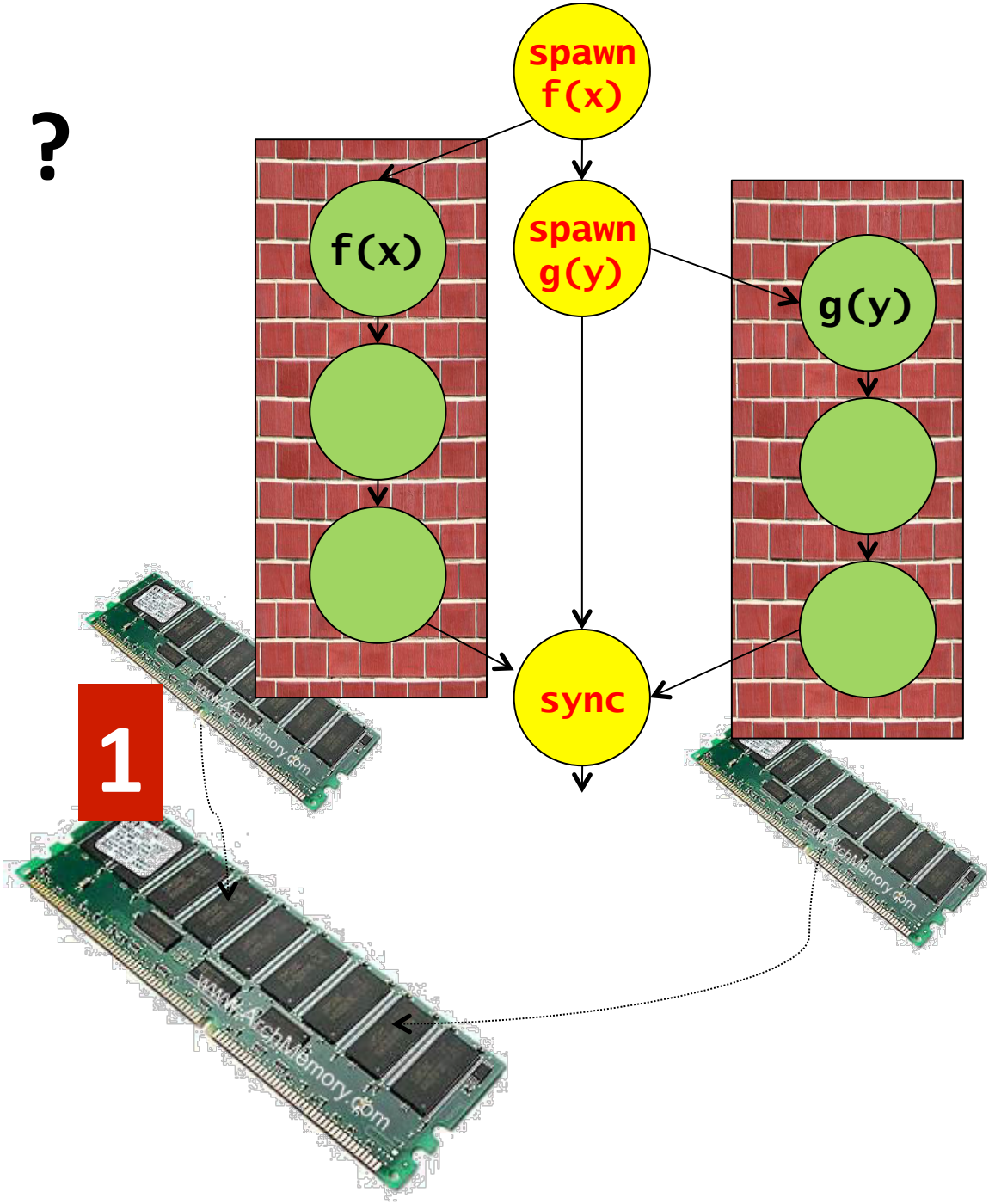
2.

?



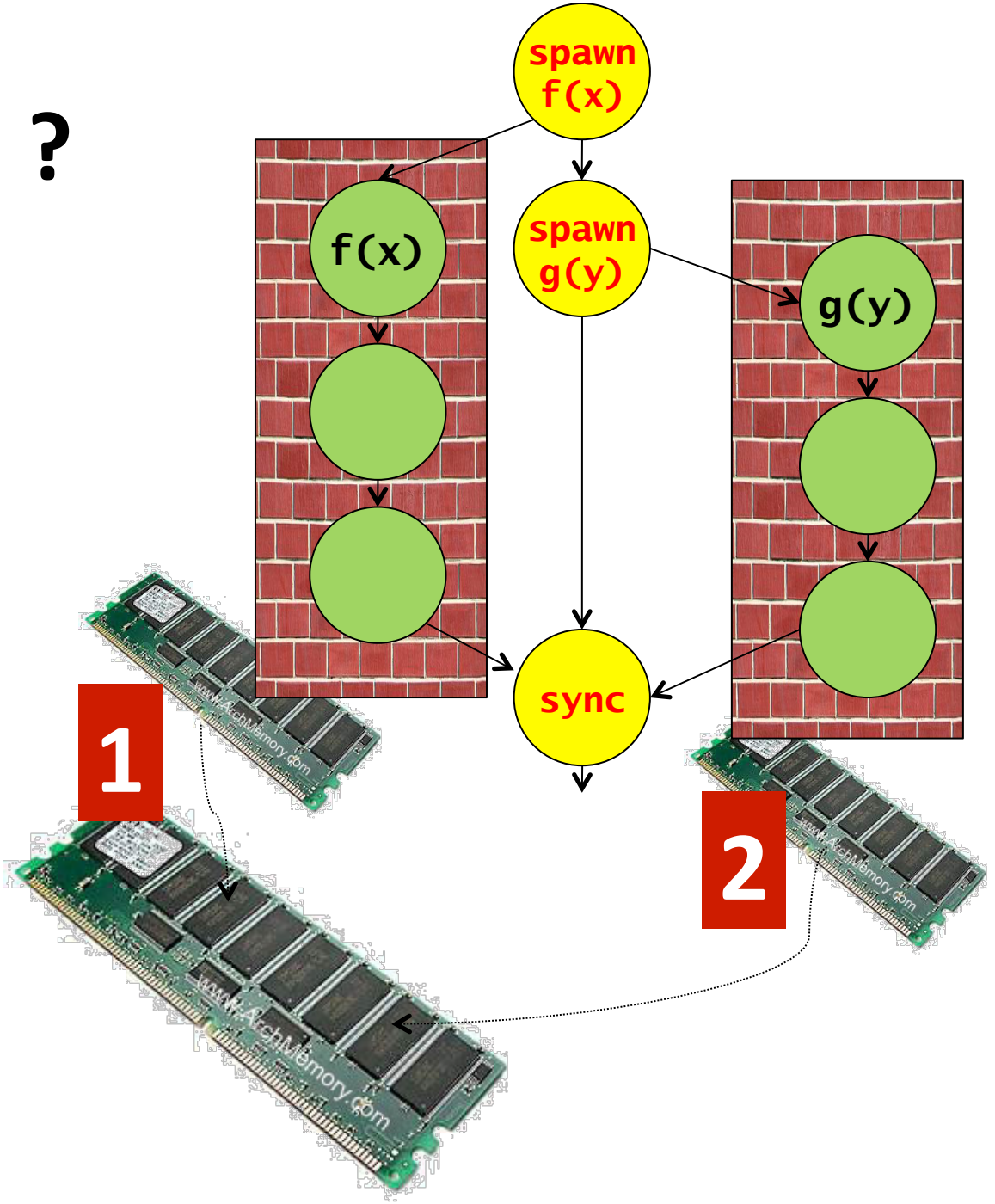
2.

?

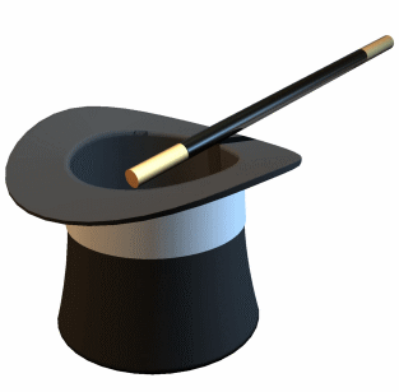


2.

?



2.  ?

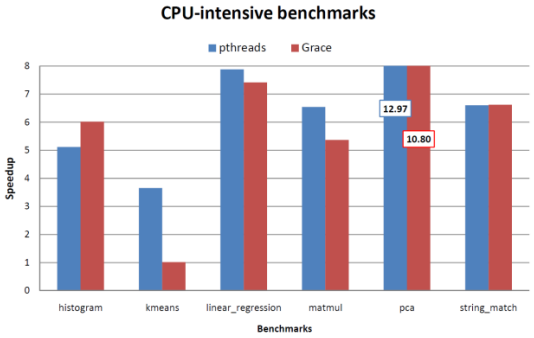


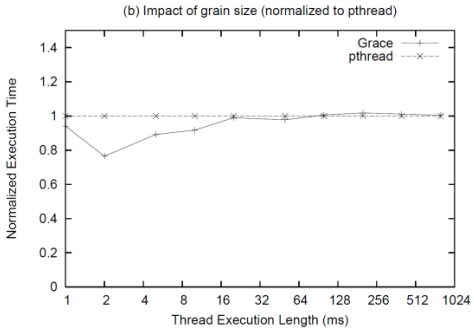
scalable heap

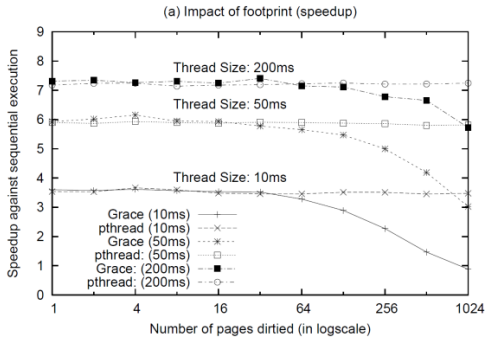
+ aligned globals

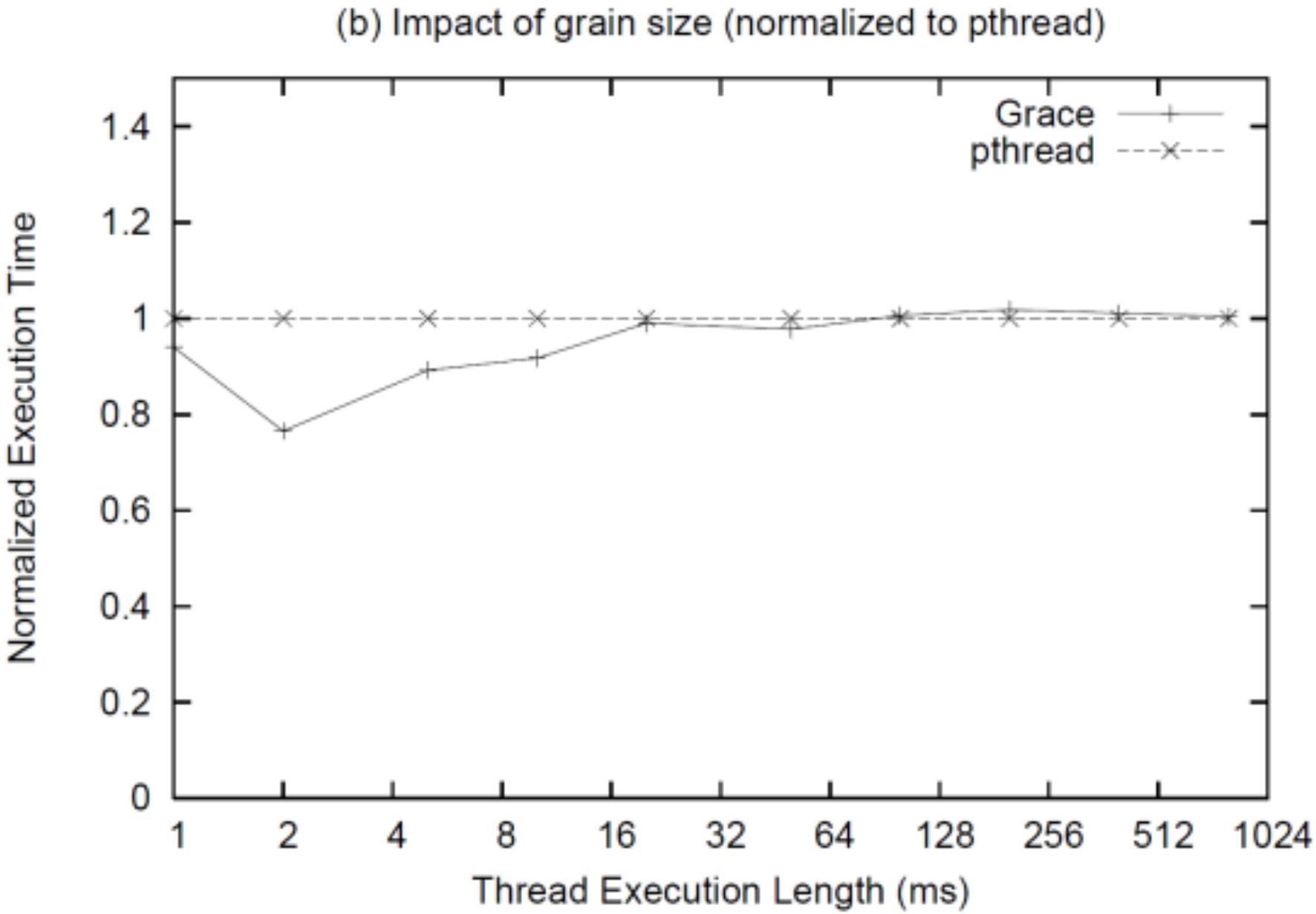
***I/O* (buffering + ordering)**

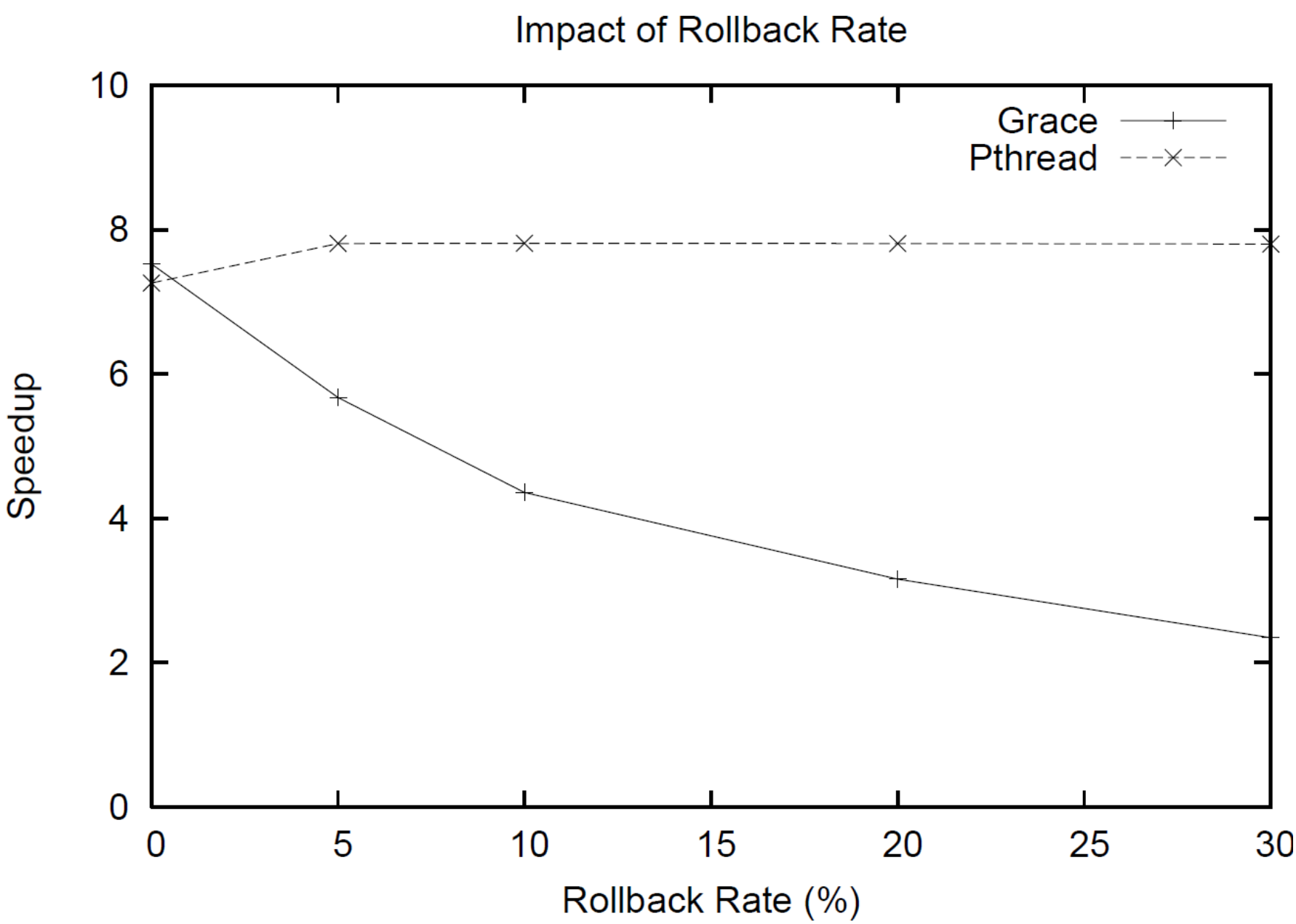
3.

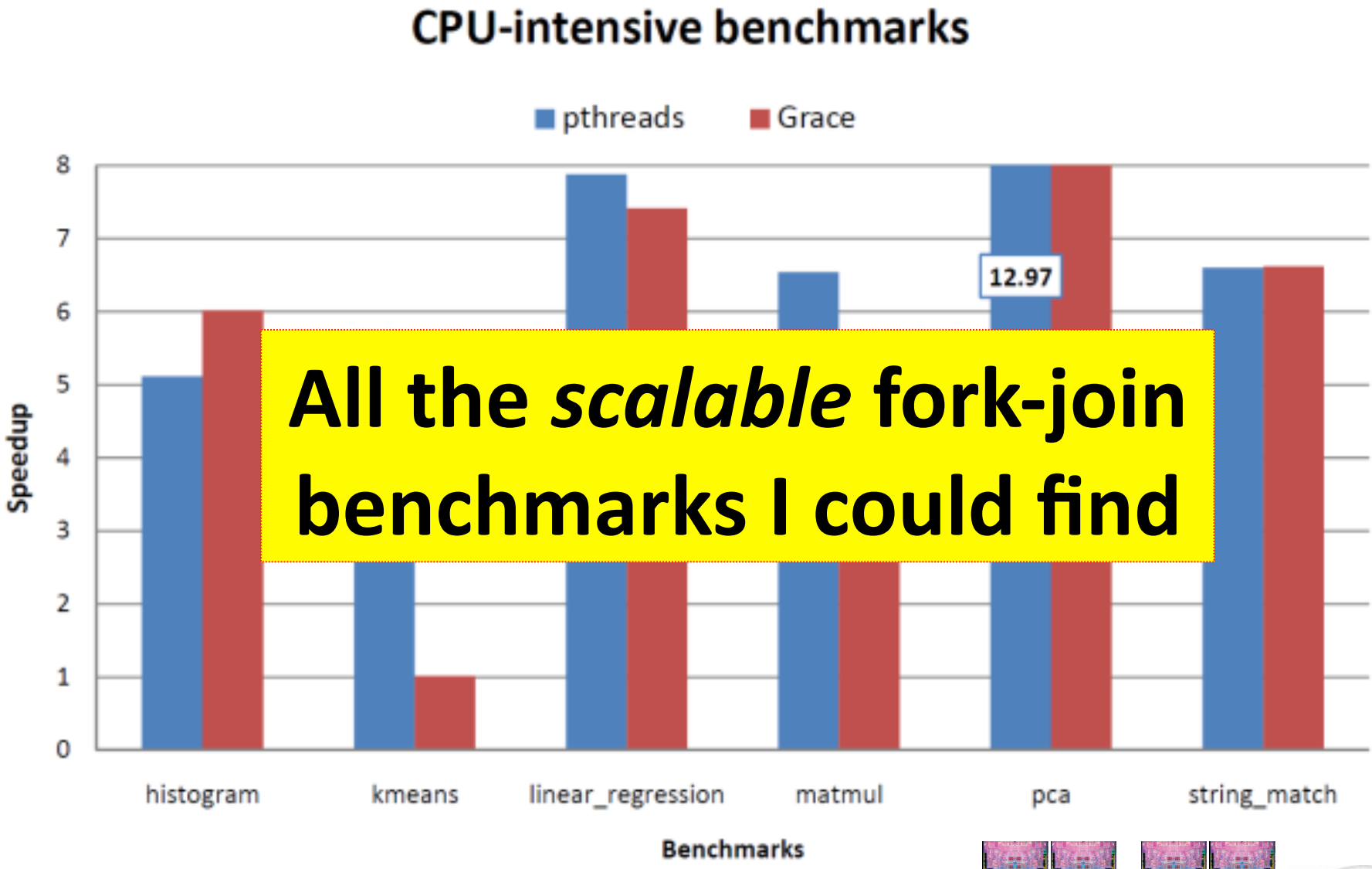

CPU-intensive benchmarks
pthreads
Grace
Speedup
0
1
2
3
4
5
6
7
8
12.97
10.80
histogram
kmeans
linear_regression
matmul
pca
string_match
Benchmarks


(b) Impact of grain size (normalized to pthread)
Normalized Execution Time
0
0.2
0.4
0.6
0.8
1
1.2
1.4
Grace
pthread
1
2
4
8
16
32
64
128
256
512
1024
Thread Execution Length (ms)

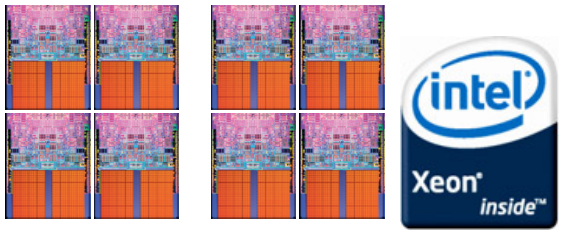

(a) Impact of footprint (speedup)
Speedup against sequential execution
0
1
2
3
4
5
6
7
8
9
Thread Size: 200ms
Thread Size: 50ms
Thread Size: 10ms
Grace (10ms)
pthread (10ms)
Grace (50ms)
pthread: (50ms)
Grace: (200ms)
pthread: (200ms)
1
4
16
64
256
1024
Number of pages dirtied (in logscale)


(b) Impact of grain size (normalized to pthread)
Grace
pthread
Normalized Execution Time
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
4
8
16
32
64
128
256
512
1024
Thread Execution Length (ms)

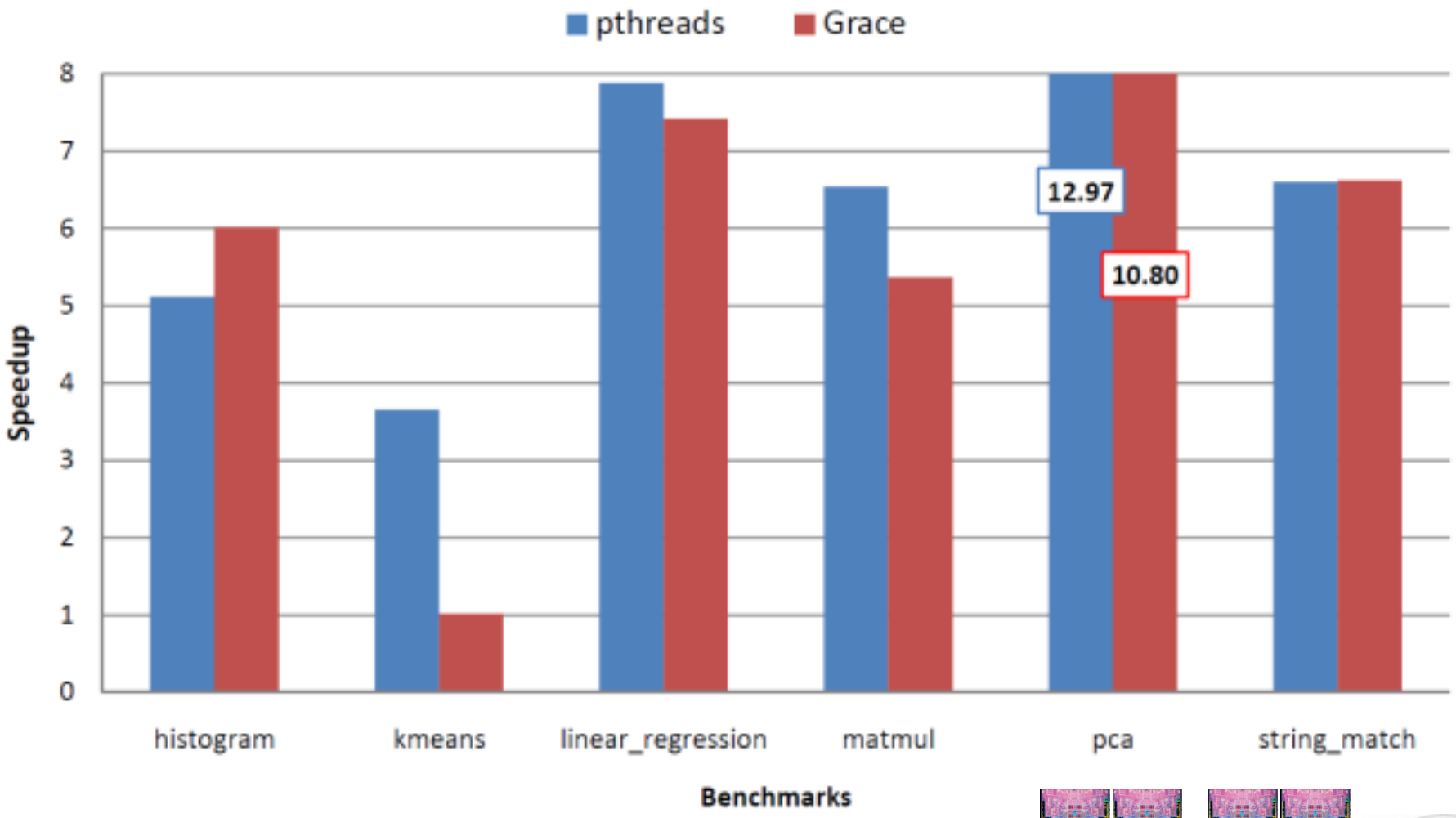

Impact of Rollback Rate
Grace
Pthread
Speedup
Rollback Rate (%)
0
2
4
6
8
10
0
5
10
15
20
25
30

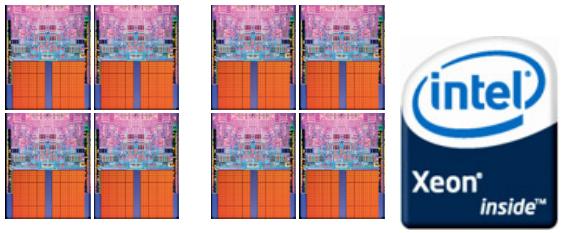


All the *scalable* fork-join benchmarks I could find



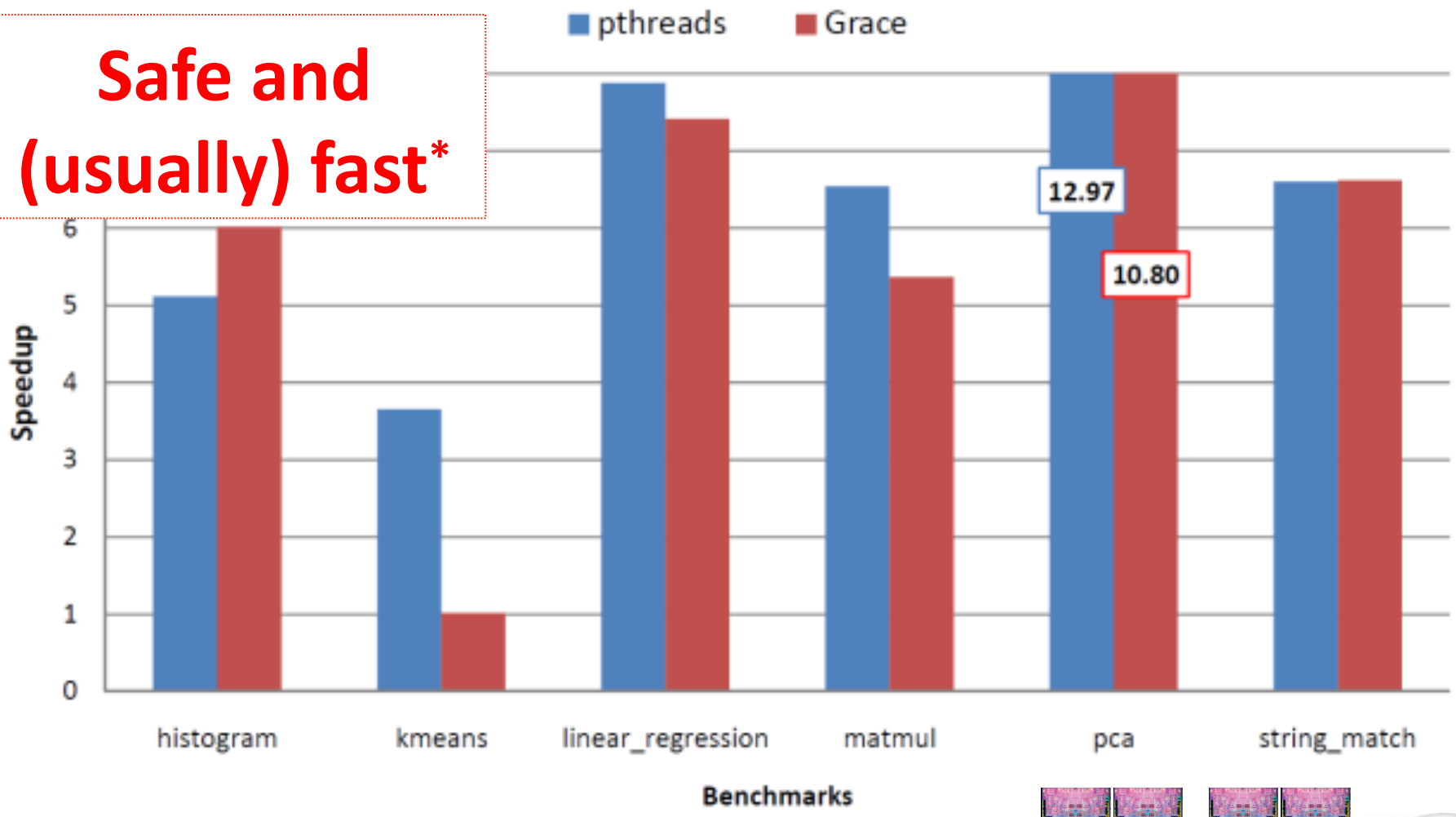
CPU-intensive benchmarks



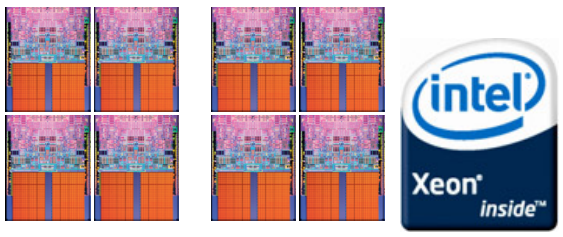


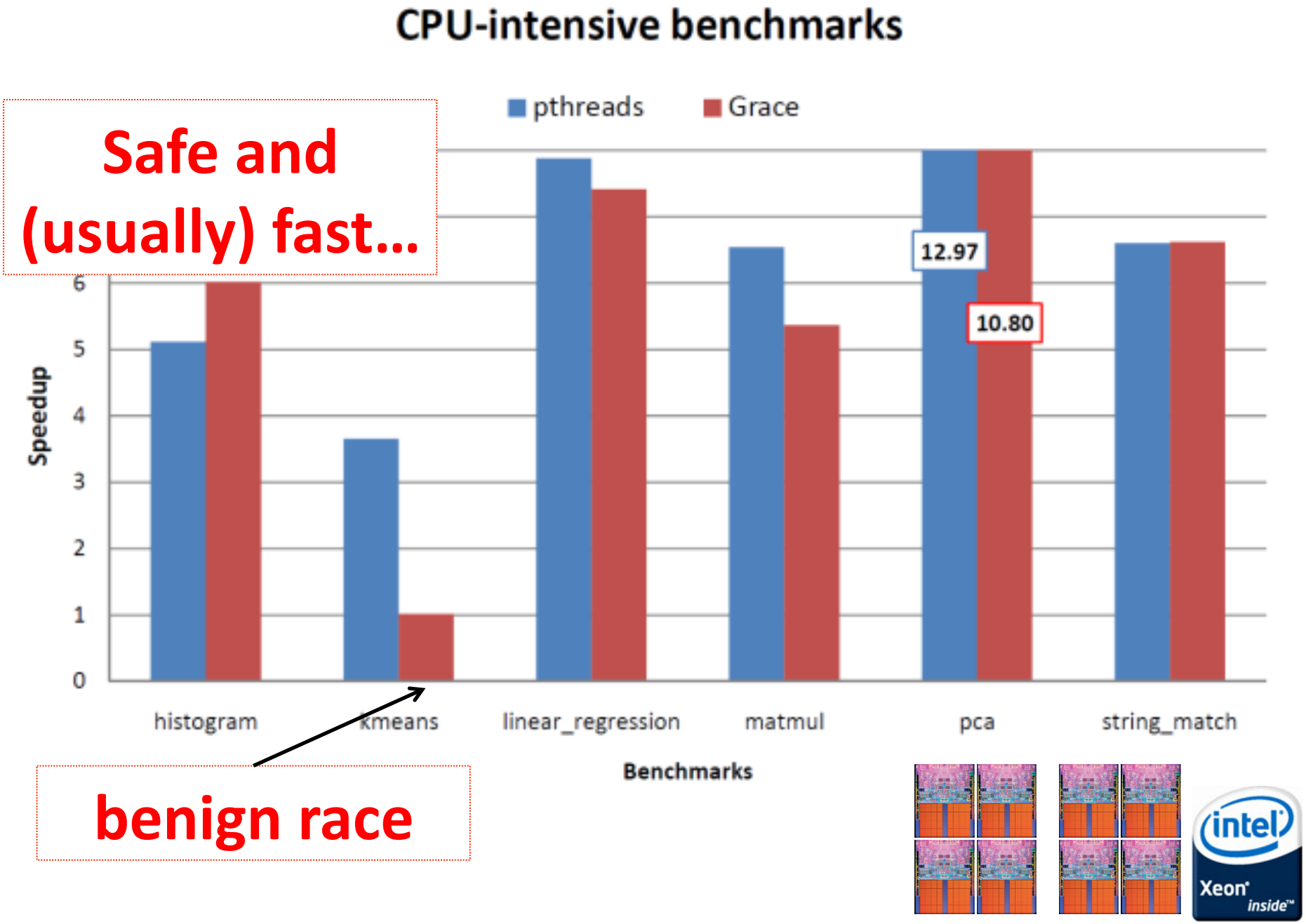
CPU-intensive benchmarks

Safe and (usually) fast*

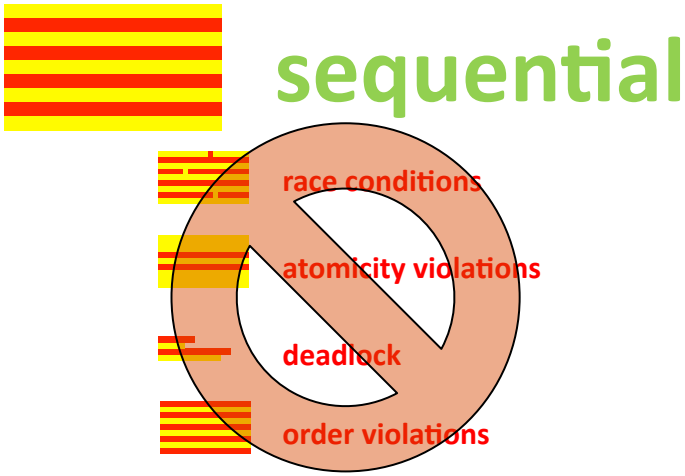


*1—16 lines changed

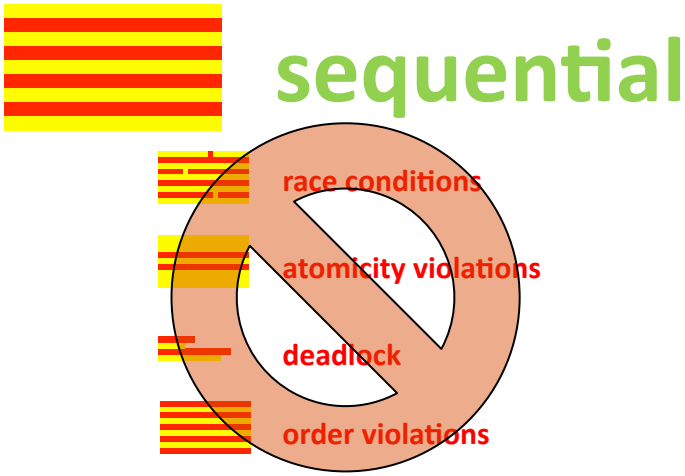


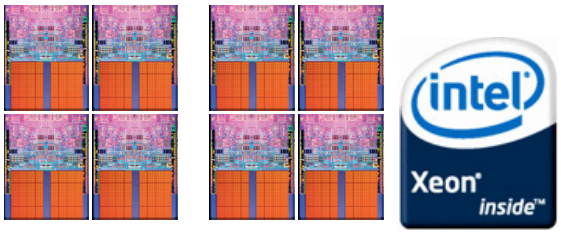

CPU-intensive benchmarks
Safe and (usually) fast…
pthreads
Grace
12.97
10.80
Speedup
0
1
2
3
4
5
6
histogram
kmeans
linear_regression
matmul
pca
string_match
Benchmarks
benign race
intel
Xeon
inside

Grace

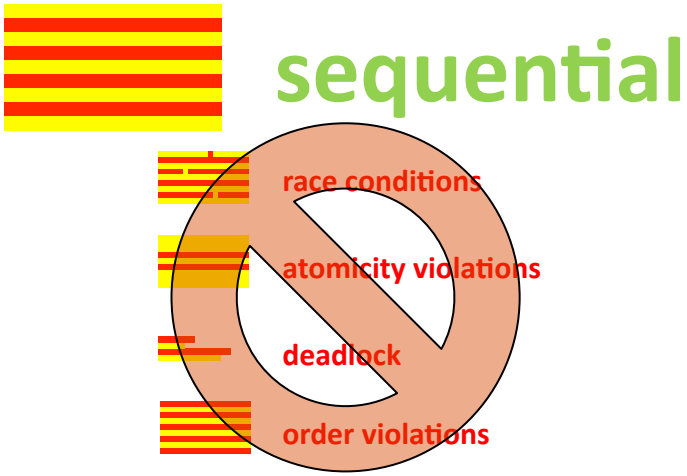

sequential
race conditions
atomicity violations
deadlock
order violations

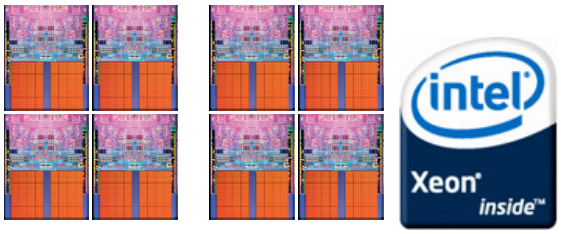
Grace

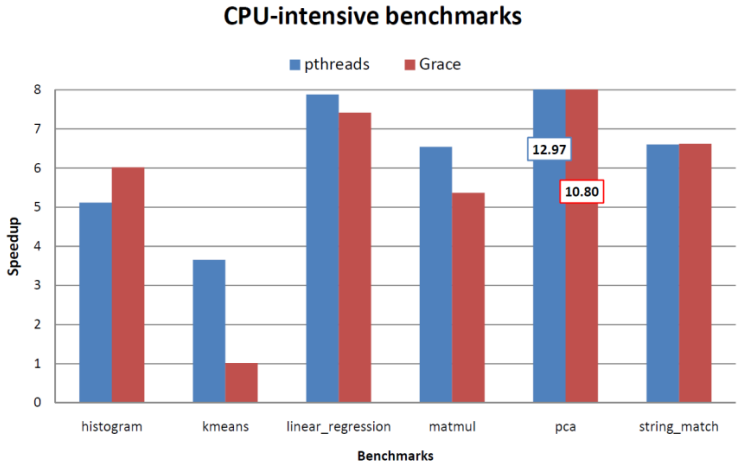

sequential
race conditions
atomicity violations
deadlock
order violations


intel
Xeon
inside™

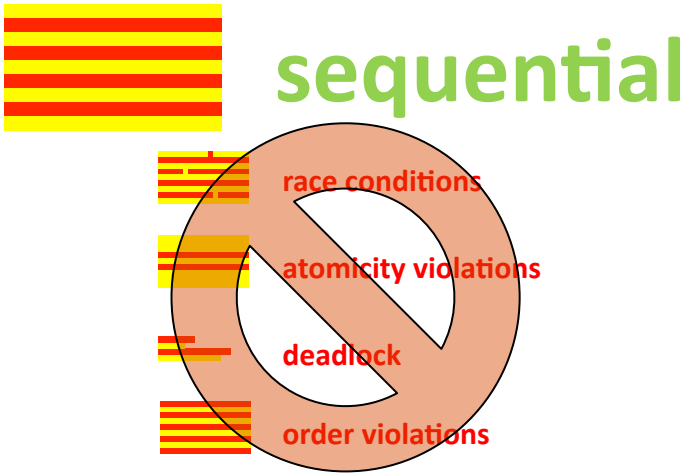
Grace

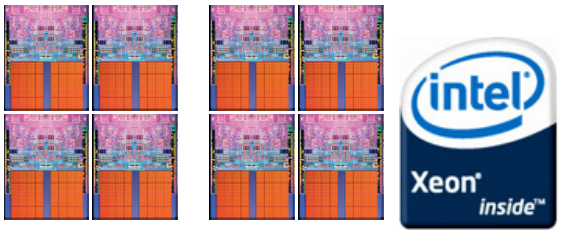

sequential
race conditions
atomicity violations
deadlock
order violations

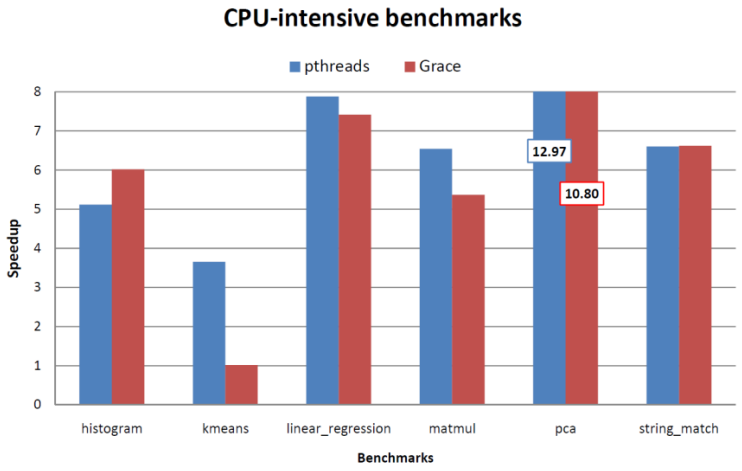

intel
Xeon
inside™


CPU-intensive benchmarks
pthreads
Grace
Speedup
0
1
2
3
4
5
6
7
8
12.97
10.80
histogram
kmeans
linear_regression
matmul
pca
string_match
Benchmarks

Grace








```
% g++ myprog.cpp -lgrace   -o myprog
```

“Moltes gràcies”

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Barcelona Supercomputing Center
Centro Nacional de Supercomputación