自己紹介みたいなもの

- ハンドルネーム ikwzm
- 現在隠居中(今年 6 月で 28 年間勤めた会社を辞めました)
- もうすぐ 51 才(けっこう年)
- 主に論理回路設計 (回路図～VHDL)
- たまにプログラム(SPARC/PowerPC アセンブラ～ C)

最近(隠居後)の活動

・Qiita に何件か投稿

（http://qiita.com/ikwzm/items）

フィボナッチを求める回路を

VHDL / Synthesijer / PolyPhony /

Synverll / Neon Light / Vivado-HLS

使って合成してみました

それでも
やっぱり
VHDL がいい

VHDL を使う唯一の理由

VHDL を使う唯一の理由

それは

並列プログラミング

並列プログラミング？

そんなの C でも Java でも出来るぞ

並列プログラミング？

そんなの C でも Java でも出来るぞ

最近のコンピュータじゃ当たり前だよ

並列プログラミング？

そんなの C でも Java でも出来るぞ

最近のコンピュータじゃ当たり前だよ

"並行コンピューティング技法"知ってる？

ちょっと待って

コンピューター屋さんと私とでは
ちょっと
並列のニュアンスが違うみたい

どこが違う？

どこが違う？

それは

粒度(granularity)

結論から先に言うと

・コンピューターの並列の粒度は

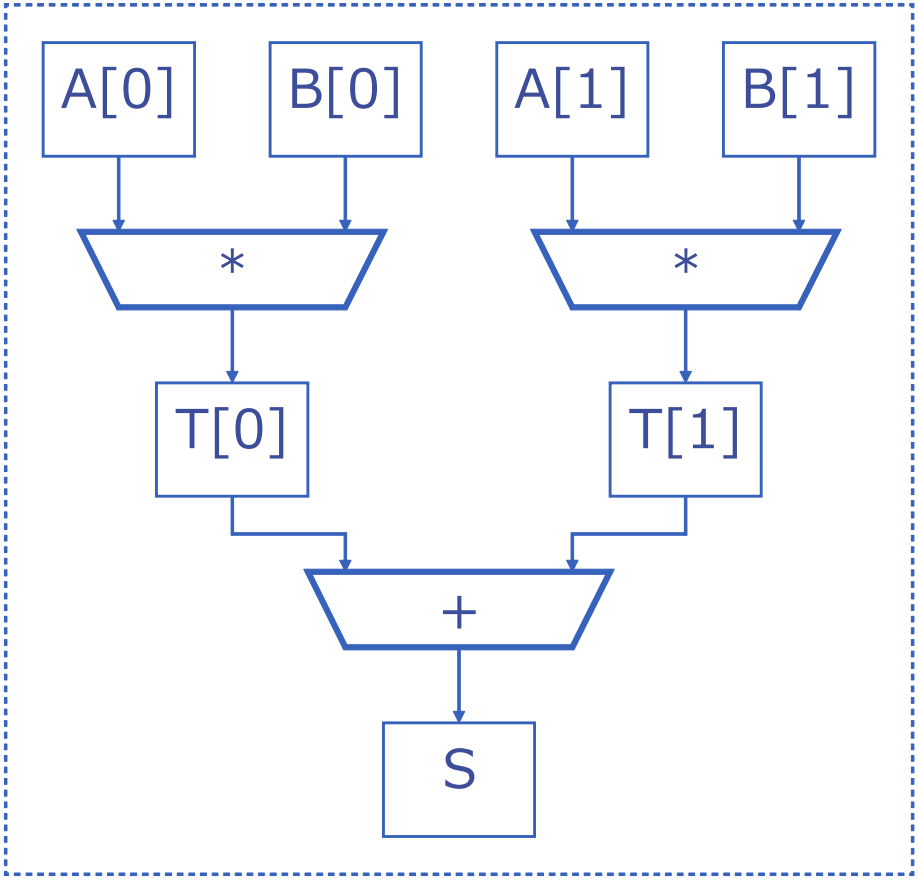
タスク(Task)/スレッド(Thread)

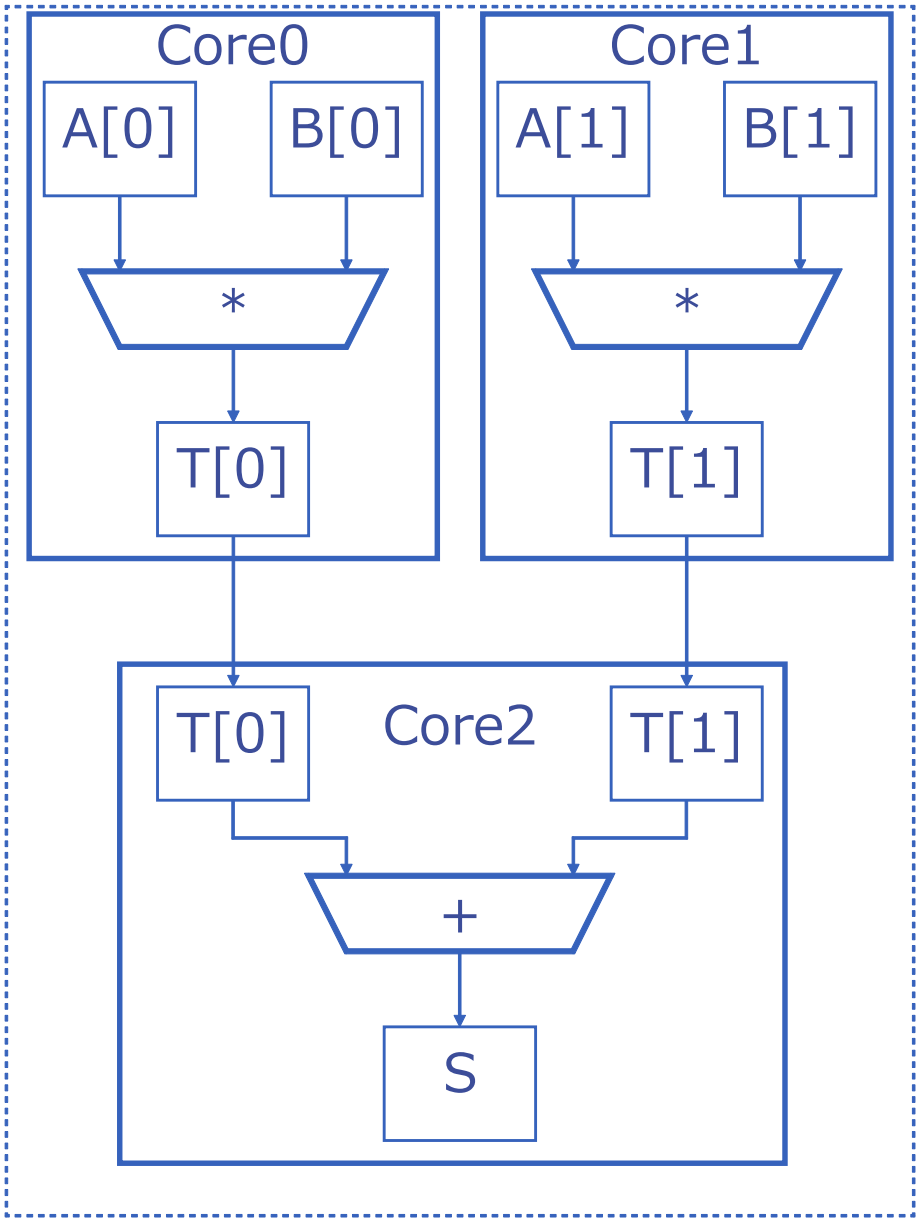
・論理回路の並列の粒度は

文(Statement)

例えば (かなり強引だけど)

```
S=0;
for(i=0;i<2;i++){
   T=A[i]*B[i];
   S=S+T;
}
```



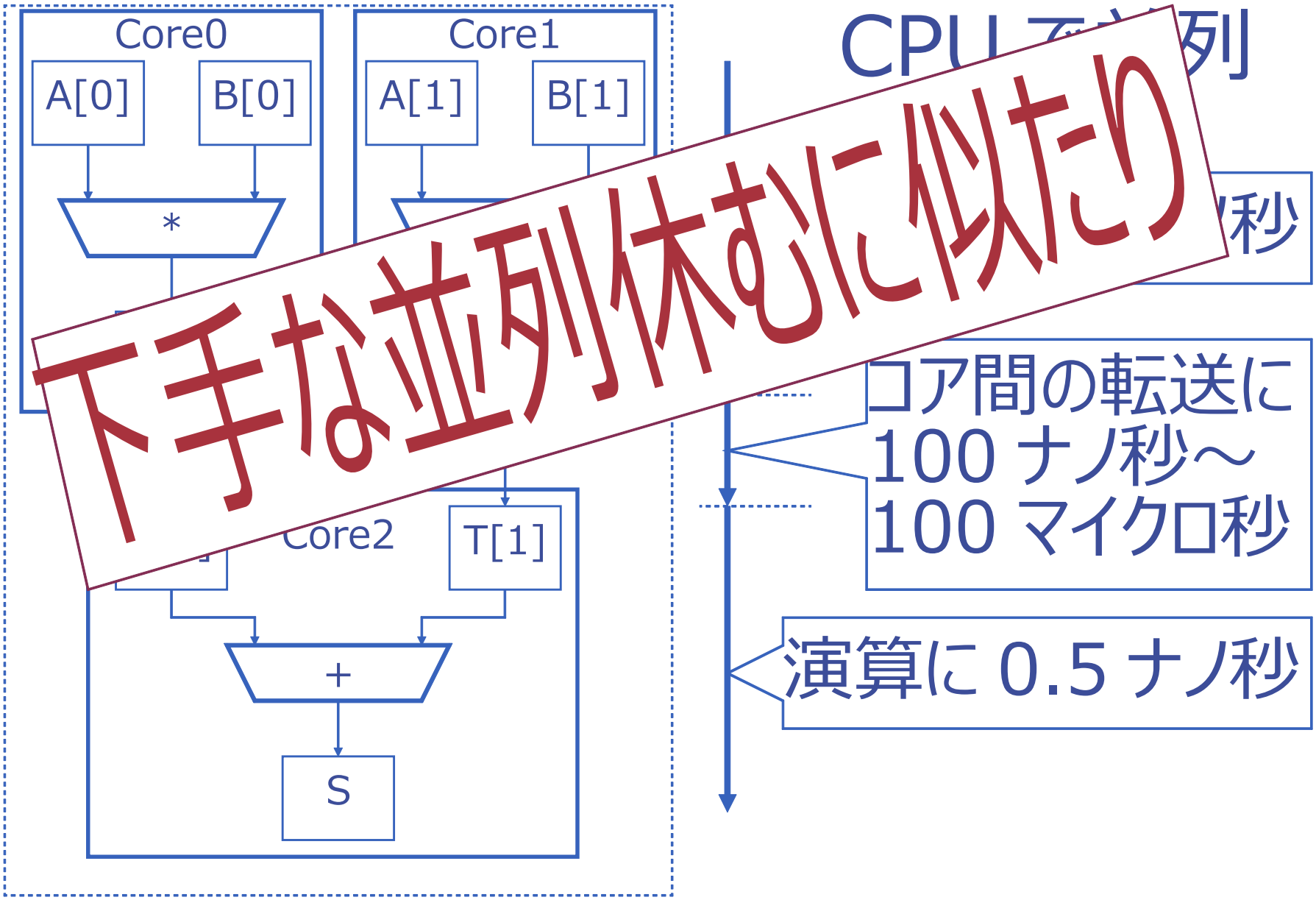


CPU で並列

演算に 0.5 ナノ秒

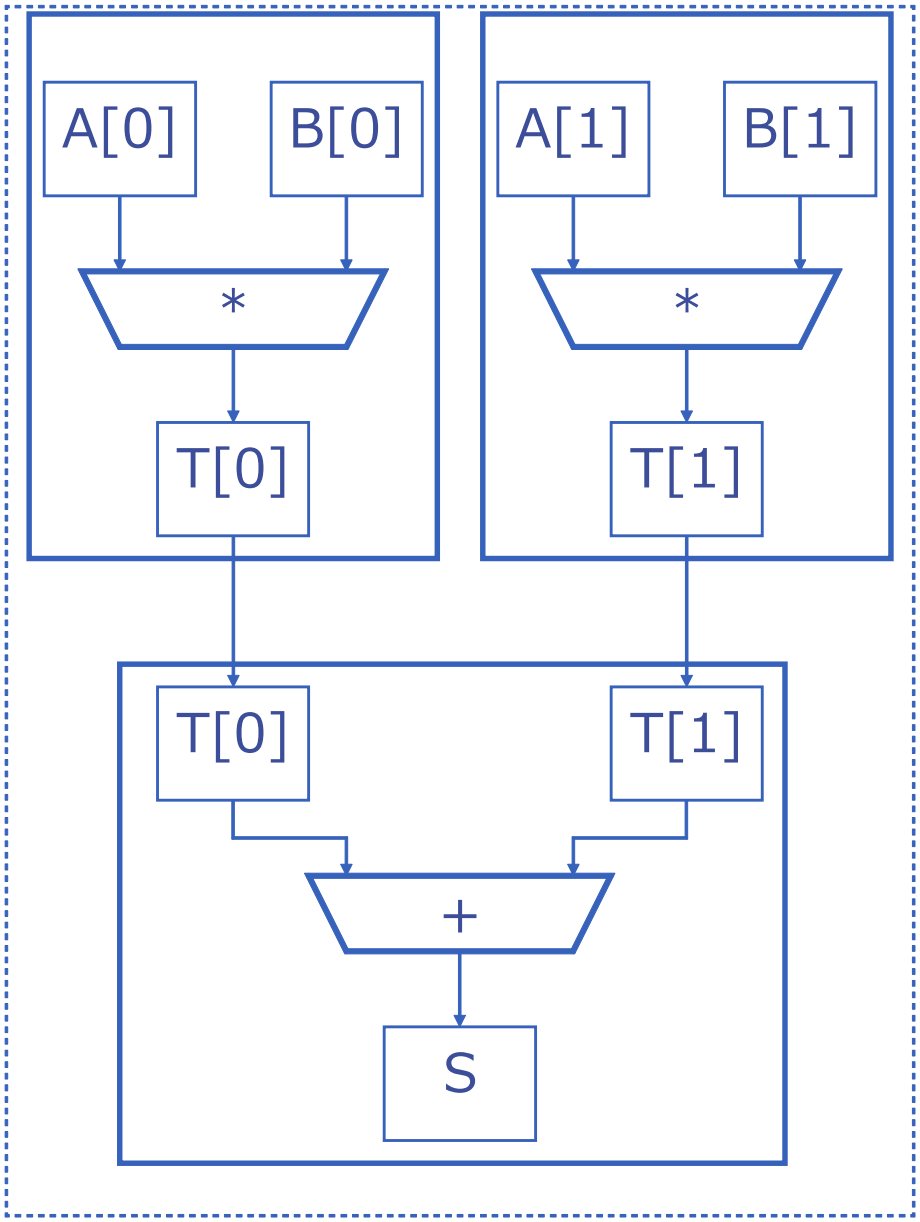
コア間の転送に
100 ナノ秒～
100 マイクロ秒

演算に 0.5 ナノ秒


Core0
A[0]
B[0]
*
Core1
A[1]
B[1]
Core2
T[1]
+
S
CPU で並列
秒
下手な並列休むに似たり
コア間の転送に
100 ナノ秒～
100 マイクロ秒
演算に 0.5 ナノ秒

コンピューターの場合

コア(CPU)間の転送(オーバーヘッド)が 1 命令の処理時間に比べて 3～5 桁ほど大きいので、 並列の粒度を大きく(タスク程度に)しないと効率が悪い。



論理回路だと

演算に数ナノ秒

回路間の転送に１ナノ秒～数ナノ秒

演算に数ナノ秒

論理回路の場合、

演算回路間の転送(オーバーヘッド)が演算時間に比べて小さいので、並列の粒度を 1 演算(言語にして 1statement 程度)にすることが出来る。

文(Statement)単位で並列を
いじくれるのは
HDL(Hardware Description Language)
くらいしか無かった

ただし、いじくれるのが

楽しい(たのしい)

からであって、決して

楽(らく)

ではないよ。念のため

文(Statement)単位での
並列処理を
一からプログラミングするのは
かなり面倒です

というわけで

高位合成
もっと頑張って

何をだよ

いろんな意味で