計算機構成　第1回
ガイダンス
VerilogHDLのシミュレーション環境

情報工学科
天野英晴

いまさらだが、、、

- コンピュータはIT社会の基盤部品
  - ノートブック、スマートフォーン、タブレット
  - サーバー、クラウド、スーパーコンピュータ
  - ビデオ機器、テレビ、ゲーム機器
  - ネットワーク機器
  - 冷暖房、冷蔵庫、電気釜、洗濯機、掃除機だって制御はコンピュータ

しかし、概観の話は「計算機基礎」でやっている

いまさらですが、コンピュータはIT社会の基盤となるパーツです。ノートブック、スマフォ、タブレットなど身近なコンピュータについては良くご存知だと思います。サーバー、クラウド、スーパコンピュータなど、大規模で普段は目に触れないコンピュータもありますが、皆さんがWebでAmazonやGoogleをクリックするとき、実はそれを使っているのです。ビデオやテレビ、ゲーム機器はコンピュータが主たる働きをします。ネットワーク機器は、もちろんネットワークの接続が主体ですが、制御するのはコンピュータです。これらは比較的コンピュータとして意識しやすいものですが、冷暖房装置、冷蔵庫、電気釜などの家電製品にだってコンピュータは使われています。このコンピュータの概観については既に計算機基礎でやっていると思います。

何をやるか？

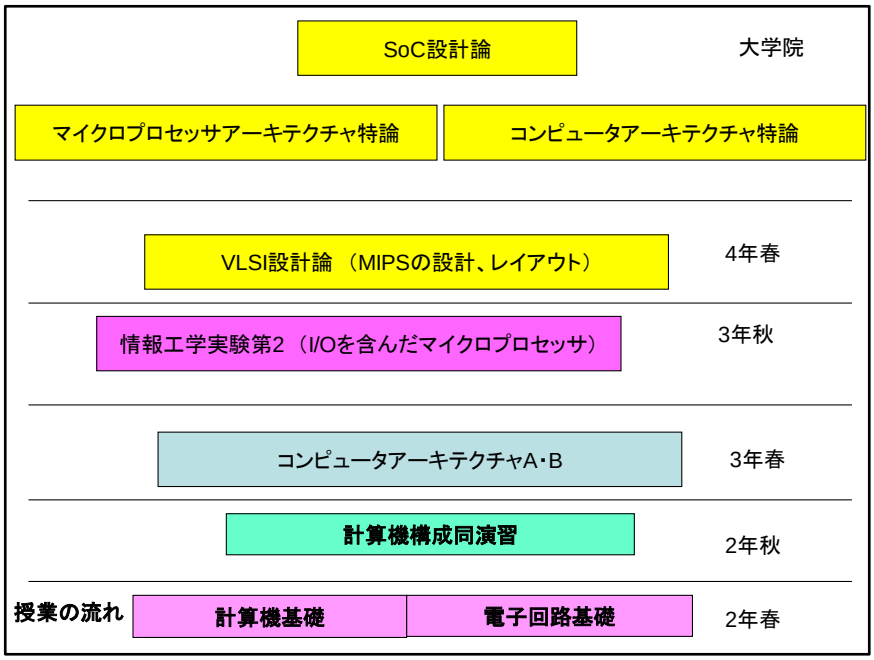
- CPU（中央処理装置）の設計をやり、シミュレーションをやりながら、内部構成を理解する。
- メモリの階層構造を理解する。
- I/Oの基本を理解する
- RISC (Reduced Instruction Set Computer)の命令セット、構成を中心に据える
- ハードウェア記述言語でのディジタル回路設計を学ぶ
  - Verilog-HDLの記述方式、シミュレーション方法
  - 演算回路
  - ALUと選択構文
  - CPUのデータパス、レジスタとメモリ
  - プログラム格納型計算機
  - RISCの命令セットアーキテクチャ
  - 分岐命令
  - サブルーチンコールとスタック
  - メモリの階層
  - キャッシュ
  - 仮想記憶
  - I/Oの基本

ではこの授業で何をやるかをざっくり説明しましょう。内田樹がどこかで書いていたようにシラバスほどバカバカしいものはありません。そこに書かれていたことを読んで、なんだかちゃんとわかるんだったら授業を受ける必要はないと言えます。とはいえ、雰囲気が分かること、自分の力で履修可能かどうか判断することは重要だと思います。この授業ではCPU（中央処理装置）の設計をやり、シミュレーションをやりながら、内部構成を理解します。CPUはRISC（Reduced Instruction Set Computer)の命令セットと構成を中心に据えます。現在のコンピュータはスマートフォンからスーパーコンピュータまでRISCでできており、これを勉強するのは当然といえます。この授業の特徴はハードウェア記述言語を使ってRISCを設計しながら、コンピュータを勉強していく点にあります。具体的な学習項目はスライドに列挙するとおりで、コンピュータの基本的な事項です。この授業ではコンピュータだけでなく、ハードウェア記述言語を使ったディジタル設計技術を勉強できる点に注意ください。次にメモリ階層について紹介し、I/Oの基本を習得します。

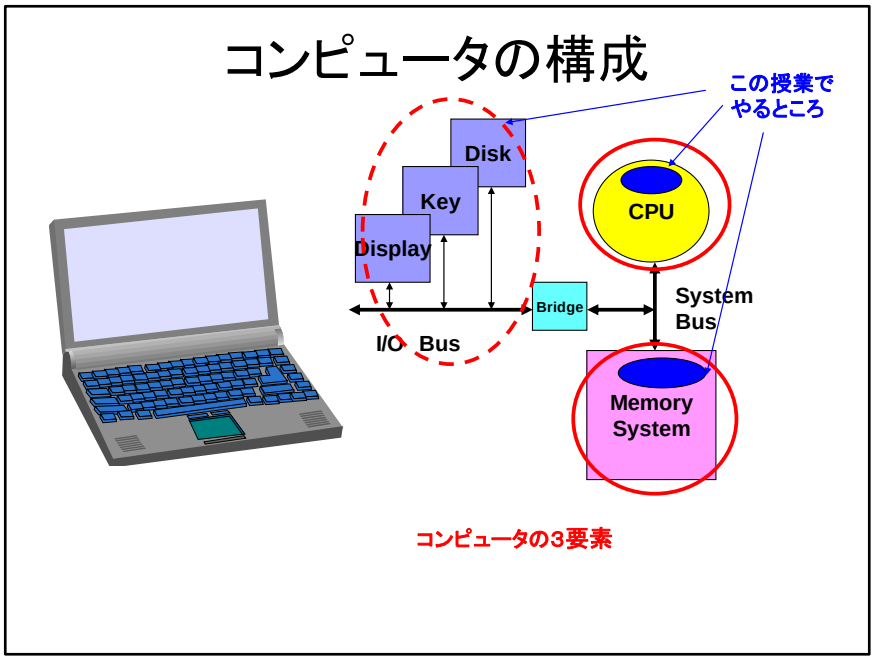
授業のやり方

- 授業資料はhttp://www.am.ics.keio.ac.jpに掲示
- 授業を90分、演習を90分→3単位はやや単位効率が悪いがその分調整する
- 演習はTAが面倒見てくれる。
- 6時15分には基本的に終わる。多くの人は6時前に帰る。
- 成績の付け方
  - 試験＋演習（各回5点、間違うと0点）により付ける
  - 休んだ場合、後から提出してもよい
  - この科目は楽に単位が取れるという意味で楽勝科目ではないが、Aの比率は高い→普通に演習を提出し、試験前にちょっと勉強すればAが取れる！

この授業ではWebに教材を掲示します。Webには用語集やVerilog－HDLの文法、設計事例も載っていますので、ぜひ見てください。この授業は90分間授業をやってから90分間演習がありますが、単位は3単位です。これは単位効率が悪いと思われるかもしれませんが、場合によっては授業がなかったり、演習があっさり終わったりもするので、大体リーゾナブルな線かと思います。演習中は二人のTAが親切に面倒を見てくれます。終わったら帰るシステムで、6時15分には絶対に終わります。多くの人は6時前に帰り、早い人は5時くらいに帰っちゃうのですが、5時半くらいからバイトを入れるのは無謀です。場合によっては難しくて苦戦することもあるからです。6時半以降にしてください。演習の提出口は学期末まで開けておきますので、休んだ場合は資料を読んで自分で演習をやってわからないことがあれば次回に僕かTAに聞いて提出してください。あまり課題を溜めると苦しいです。
成績の付け方は試験の点＋毎回の演習点です。Webに去年の基準が出てますのでご覧ください。概ね今年も同じです。僕の担当する多くの科目同様、この科目はAの割合が多い（75%くらい）という点では楽勝科目です。しかし、毎回それなりの演習があるので努力しないで単位が取れるわけではないです。教材、ヒントなどWebに豊富に載せています。ぜひ活用してください。わかんないことばが出てきたら、聞いてください。あるいは用語集をご覧ください。業界の言葉になれることも重要です。



それぞれの授業との関係です。この授業の次としてはコンピュータアーキテクチャA,Bと情報工学実験第２があります。コンピュータアーキテクチャAはマルチコアに集中し、並列プログラミングを紹介します。コンピュータアーキテクチャBは、この授業の直接の続きで、CPUの高速化、最適化のテクニックを紹介します。いずれも、1/4期で7回構成です。情報工学実験第２では、実際にFPGA上で、このMIPSプロセッサを実装し、実際のスイッチ、ブザー、LEDを使って簡単なゲームを作ります。さらに4年のVLSI設計論では、このMIPSをチップに実装する方法を学びます。大学院では、最近のマルチコアプロセッサや、高速なマイクロプロセッサの技術を学びます。

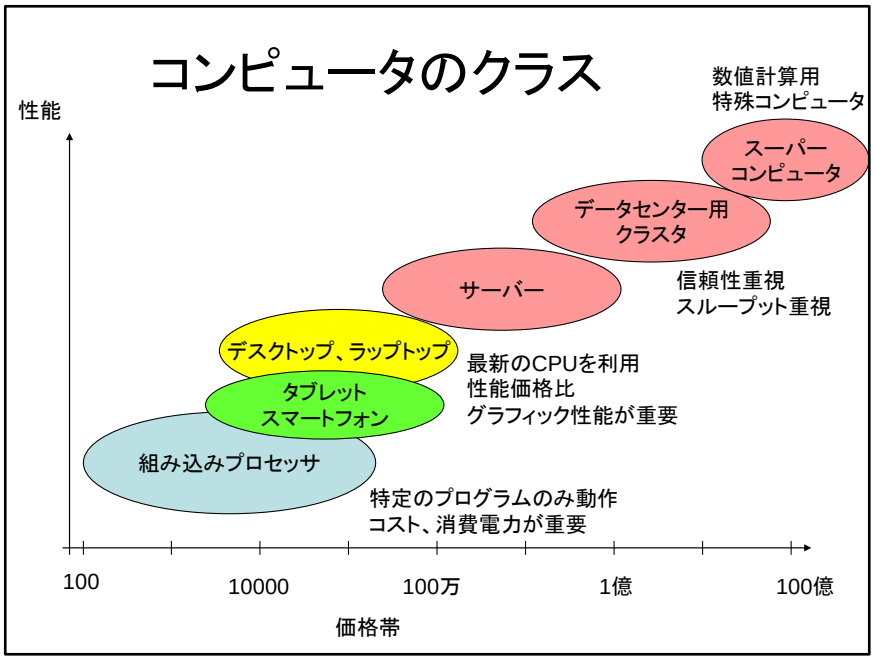


コンピュータは3つの部分に分けて考えます。中央処理装置CPU(Central Processing Unit)は、命令をメモリから取ってきて、それに従って演算処理を行う部分で、コンピュータの中心部です。メモリシステムは、命令、データを蓄えておく部分です。CPUに比べて地味な感じがしますが、実はコンピュータのコスト、性能に与える影響は大きいです。電子回路基礎で紹介したメモリ素子を組み合わせて利用します。最後に、外部との情報をやり取りを行うのが入出力装置(I/O:Input/Output)です。ディスプレイ、キーボード、マウスなどの人間とのやりとりを行う部分、Etherネットワーク、ディスクなどの補助記憶、USBなどを含みます。この3つの要素はどれも重要ですが、この授業では、CPUとメモリシステムの基本を中心に学び、入出力装置は、軽く一回やって、秋学期の実験で実際に入出力を行って実感を掴みます。

コンピュータの基本

- コンピュータのクラス
- コンピュータの仕組み
  - コンピュータとディジタル回路
  - 中央処理装置CPU
  - メモリ装置
  - 入出力装置
- コンピュータの歴史

今日は、最初なので、計算機基礎でも学んだコンピュータの基本をおさらいしましょう。まずコンピュータのクラスをやり、次に3要素を順に概観します。それからごくごく簡単なコンピュータの歴史をおさらいします。



コンピュータほど、値段と性能の幅の大きい製品はちょっと他にないのではないかと思います。もっとも安いものは数10円の組み込みプロセッサで、家電などの製品に組み込まれています。高いものは主に科学技術計算を高速に行うスーパーコンピュータで1000億円掛かるものもあります。（スーパーコンピュータ京は1100億円しました）性能の幅も10の9乗（10億倍）のオーダーで違います。（この辺、非常にアバウトな議論です。念のため）これらは、当然、使われ方、外見、ポイントとなる機能が違っています。大きくわけると図のようなクラスに分けられます。同じクラスのコンピュータは使われ方や重要視される性質が同じだと思って良いです。これらのクラスをざっと紹介しましょう。一つ、覚えていて良いことは、これだけ値段と性能が違うのにも関わらず、その基本的な命令の作り方は共通だと言う事です。もっとも簡単な組み込みプロセッサとスーパーコンピュータは実はほとんど同じ命令の作り方が使われています。これがRISC（Reduced Instruction Set Computer）と呼ぶ方法で、この授業で学びます。

コンピュータの仕組み

(1)コンピュータのいろいろ
パーソナルコンピュータ

- 個人が使うコンピュータ
- Personal Computer (PC)
- デスクトップ型とラップトップ型（ノートPC）
- 応答性能、グラフィック性能が重要

デスクトップ型http://www.dell.com/jp/より

ラップトップ型

もっともコンピュータらしいコンピュータがパーソナルコンピュータ（PC: Personal Computer)、あるいはパソコンです。机に置いて使うのをデスクトップ型と呼び、折りたたんで（最近は切り離すのも出てきている）運ぶタイプをラップトップ型と呼びます。日本ではこれをノートパソコン、ノートPCなどと呼ぶことが多いです。これらのコンピュータでは、人間が実際にキーボードやマウスを操作して使いますので、応答性能が重要です。また、画面に美しく精細なグラフィックが表示されるので、このための性能が重要です。また、コスト当たりの性能が重視されるので、最新のCPUを使う場合が多いです。デスクトップ型は、Intel社のCPUを使い、標準的なメモリ素子を装備して、WindowsがOSとして走る（MacOS,Linuxも時々使われる）標準的な製品が多く使われ、各社の製品がほとんど同じになってしまいました。このような状況では1台当たりの利益を得ることが難しく、コスト勝負のビジネスになっています。最近はデスクトップ型はオフィスでの利用に限定され、個人ではラップトップ型を使うことが多いです。皆さんもラップトップ型をお持ちだと思います。ラップトップ型では、重量、スタイル、電池の持続時間が、応答性能、グラフィック性能に並んで重視され、デスクトップ型と違って各社それぞれの特徴を出す余地があります。

コンピュータの仕組み

(1)コンピュータのいろいろ
サーバ

- 企業、官公庁、データセンターなどに置かれる
- 巨大なデータ、Webページなどを管理
  - チケット予約、ネット通販、ソーシャルネットワークなどもサーバが管理
- Amazon, Googleなどは強力な検索エンジンを装備
- 一定時間に処理するジョブ数、信頼性が重要

ブレード型サーバ　小型で効率の良いサーバ（pr.fujitsu.comより）

個人で使うのではなく、企業、官公庁、データセンターなどに置かれ、集中的に演算処理、データ記憶を行うコンピュータを、サーバーと呼びます。図は、ブレード型と呼ばれ、小型で効率の高いサーバーで、カードを多数挿すことで、多数のCPUを利用して、一度に多数のジョブを処理します。大規模なサーバーは、クラスタと呼ばれて、多数のラック（筐体）で実装され、やや高価で性能の高いネットワークで接続され、巨大なデータ記憶装置を持っています。皆さんも情報の検索、チケット予約、ネット販売、ソーシャルネットワークなどをWebで行うと思いますが、これらの処理を行うための巨大なサーバーは、データセンターと呼ばれる建物に置かれています。Amazon、Googleなどは巨大なデータセンターを数多く持っており、Web経由で様々な処理、データ管理などを行います。どこで行われているかわからないことから、このような処理をクラウドコンピューティングと呼びます。サーバーは、パーソナルコンピュータと違って応答時間というよりも、一定の時間に処理することのできるジョブの数（スループットと呼びます）が大事です。また、故障すると被害が大きいことから信頼性が重要です。このため、サーバーは最新のCPUではなく、やや古く実績のあるものを使うことが多いのです。信頼性の向上のためには、同じ処理を複数CPUで行い、データのコピーを複数持たせるなど冗長化の技術が使われます。

コンピュータの仕組み

(1)コンピュータのいろいろ
スーパーコンピュータ

- 科学技術計算を高速に行う巨大なコンピュータ
- 気象解析、物理学計算など
- 浮動小数演算能力が重要

スーパーコンピュータ 京は一度は世界一を奪取した

スーパーコンピュータは、データセンターに置かれた巨大サーバーと同様、たくさんの筐体にCPUのカードを積んだ構成で専用の建物に格納されていますが、気象計算、物理学の計算など科学技術計算の高速化が主な目的なので、浮動小数計算といって科学技術の数値計算能力が重要です。スーパーコンピュータのうちの性能の高いものは、非常に高価なことから国家プロジェクトで巨額の税金を用いて作られます。このため、マスコミの関心度が高く、新聞やニュースを賑わし、皆さんも記事を読むことがあるかと思います。写真は日本の国家プロジェクトで作られた「京」で、事業仕分けの対象になりかけたものの開発に成功し、一時は世界一を奪取しました。現在でも複雑な問題に対しては世界一の性能を発揮します。この辺は結構話題に上るので、機会を見つけて紹介しようと思います。

コンピュータの仕組み

(1)コンピュータのいろいろ
組み込みコンピュータ

- テレビ、DVDプレーヤー、ビデオカメラ、ゲーム、ネットワークコントローラ、車、エアコンなどに組み込まれるコンピュータ
- 特定の処理以外は行わない
- コスト、消費電力が重要
- 特定の処理に対しては高い性能が要求される場合もある

SONY PS3には強力なプロセッサCellが使われていた

組み込みコンピュータは、地デジ、DVDプレーヤー、ビデオカメラ、ゲーム、ネットワークコントローラ、車、エアコンなどの製品に組み込まれて、その部品として働きます。このため外からは見えないコンピュータです。このクラスのコンピュータは特定の処理以外は行わず、また必要な性能以上は必要ありません。組み込まれた製品に影響を与えることからコスト、消費電力が重要視されます。エアコンなどの制御には非常に小さく、低コストのマイクロコントローラと呼ばれるCPUが使われます。しかし、組み込みコンピュータが一概に性能が低いとは言えません。特定の処理に対しては高い性能が要求されるため、ゲームマシンやネットワークコントローラには強力なCPUが用いられます。この図はSONYのPS3ですが、当時としては破格の性能を持ったCellというプロセッサが使われました。今のPS4には強力なGPU(Graphic Processing Unit)と呼ばれる強力なグラフィック用の専用プロセッサが用いられています。

コンピュータの仕組み

(1)コンピュータのいろいろ
スマートフォン、タブレット

- パーソナルコンピュータと組み込みコンピュータの中間的な性格を持つ
  - アプリを入れて動作する→ パーソナルコンピュータ
  - 携帯電話、カメラ→ 組み込みコンピュータ
- パーソナルコンピュータの分野を急激に侵食しつつある

NTTDOCOMOのスマートフォン

SONYのタブレット

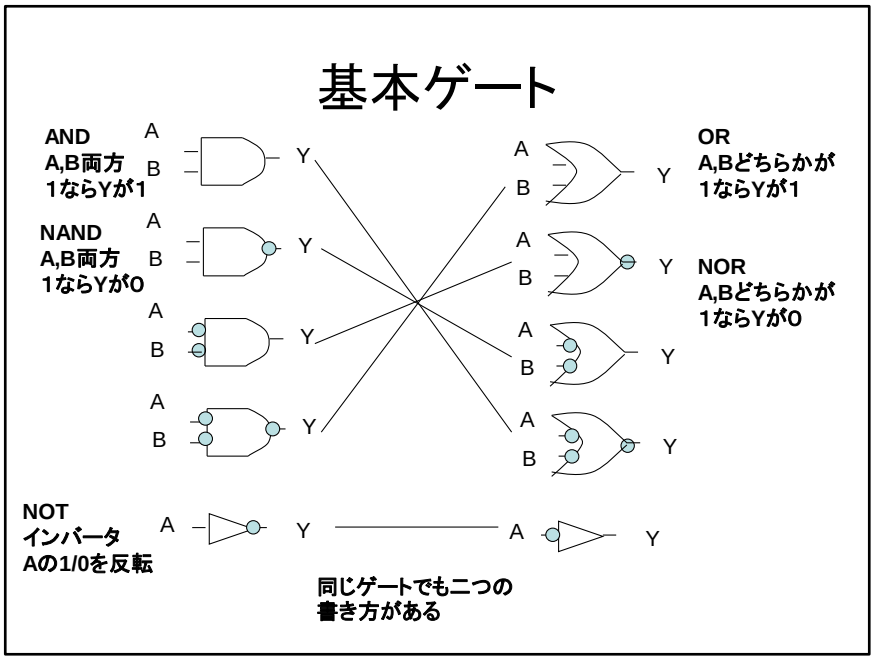
スマートフォン、タブレットは、PCと組み込み用コンピュータの中間的な性格を持ちます。電話機能、カメラ機能を中心に考えると組み込み用コンピュータですが、アプリを入れて様々なプログラムを動かすことができる点ではPCに近いです。応答性能は重要ですが、PCよりも消費電力、コストが重要視されます。要求事項のバランスが組み込みに近いと言えます。このクラスのコンピュータは、PCに代わってコンピュータ利用の中心になりつつあります。後に解説しますが、PCの領域ではIntel社のCPUが主に使われるのに対して、このクラスではARMというCPUが良く使われます。ARMもRISCの一種です。最近、RISC-VがARMに挑戦しています。この辺の話も紹介できると思います。

今まで紹介したコンピュータのクラスは重要です。これは、クラスが違うと、価値観が違ってくるからです。PCやサーバーの世界では、性能は高ければ高いほど有利です。プログラムの実行が早く終わり、一定の時間に処理できるジョブの数が増えるからです。しかし、組み込み用のコンピュータでは、必要とされる以上の性能があっても役に立ちません。その分消費電力やコストを小さくするのが良い設計です。同じコンピュータの設計者でも価値観の差が出て、会話が噛み合わないこともあります。

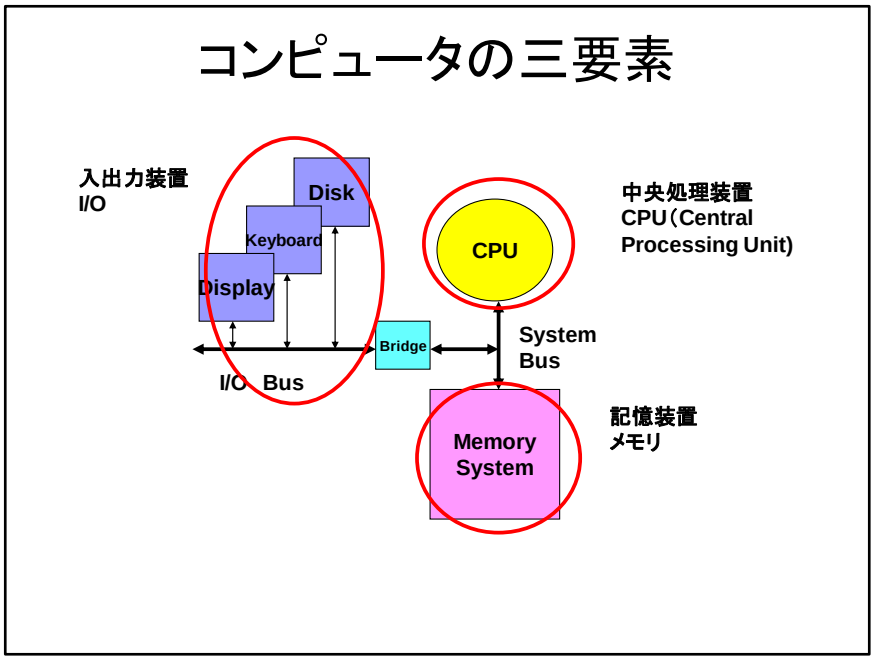
（2）コンピュータを作るディジタル回路

- コンピュータはディジタル論理回路からできている
  - 1･0だけを扱う簡単な論理素子（ゲート）を多数用いる
  - 論理ゲートは単純な機能しか持たないが高速、半導体上に多数搭載可能
    - 数ピコ秒の動作速度
    - 数億個を一つの半導体上に搭載可能
  - ブール代数を基にした設計技術が確立されている
  - ハードウェア記述言語HDL(Hardware Description Language)により記述、設計する
- 1980年以来CMOS（Complementary Metal Oxide Semiconductor）が発展し、マイクロコンピュータが急激に高性能、高集積化

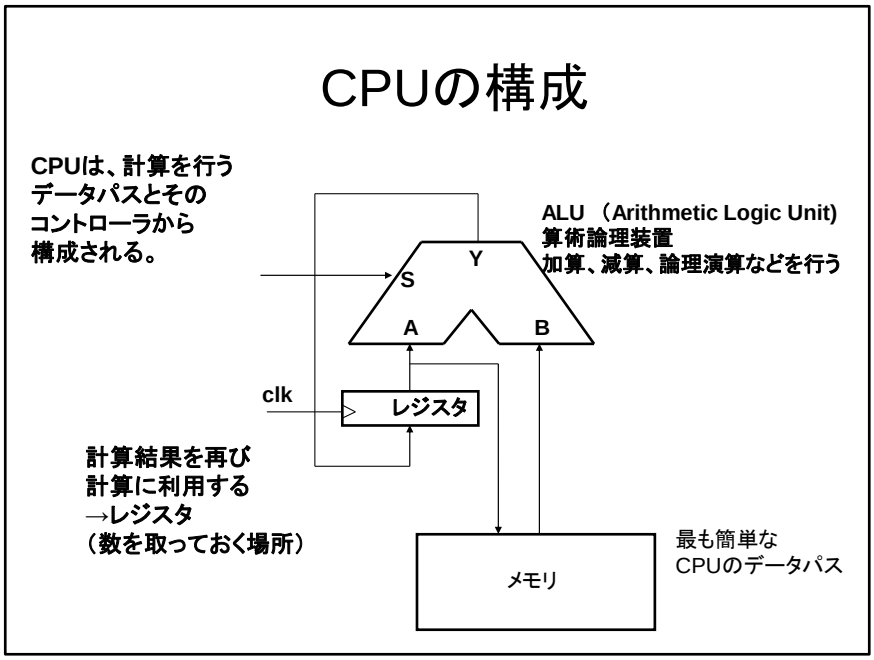
コンピュータは、典型的なディジタル回路です。1とOしか扱わないディジタル回路（論理回路）が2進数の演算を基本とするコンピュータと良く合っているためです。ディジタル回路の論理ゲートは、単純な機能しか持たない一方、高速に動作し、半導体上に多数搭載することができます。現在のコンピュータは、それぞれの論理ゲートは数ピコ秒の動作遅延で、数十億個を1個の半導体上に搭載することができます。このような大きな回路は、論理ゲート間の配線を記述する方法ではとても設計できないため、ハードウェア記述言語HDL（Hardware Description Language)で、記述するのが一般的です。ハードウェアの設計手法としてはC言語を使う方法が最近は一般化しているのですが、コンピュータの設計ではHDLが引き続き使われています。この理由は実際に記述してみると理解できると思います。電子回路基礎で学びましたが、1980年以降CMOS（Complementary Metal Oxide Semiconductor)が急速に発展し、現在、コンピュータはこのCMOS半導体の上に実装されるようになりました。



今までの論理設計の授業ではこの基本ゲートを使った設計法を勉強したと思います。この授業ではこれをベースとしてコンピュータのハードウェアを設計する方法を学びます。

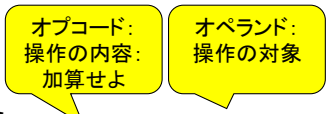


コンピュータは3つの部分に分けて考えます。中央処理装置CPU(Central Processing Unit)は、命令をメモリから取ってきて、それに従って演算処理を行う部分で、コンピュータの中心部です。メモリシステムは、命令、データを蓄えておく部分です。CPUに比べて地味な感じがしますが、実はコンピュータのコスト、性能に与える影響は大きいです。電子回路基礎の時間などで紹介したメモリ素子を組み合わせて利用します。最後に、外部との情報をやり取りを行うのが入出力装置（I/O:Input/Output)です。ディスプレイ、キーボード、マウスなどの人間とのやりとりを行う部分、Etherネットワーク、ディスクなどの補助記憶、USBなどを含みます。この3つの要素はどれも重要ですが、この授業では、CPUとメモリシステムの基本を中心に学びます。入出力装置は、3年のコンピュータアーキテクチャで基本を学び、秋学期の実験で実際に入出力を行って実感を掴みます。実際にコンピュータを使う上では、入出力装置は非常に重要なのですが、これは実際に使ってみるのが一番です。この話は後の方でもう一度紹介します。



CPUは、計算を行うデータパスとこれを制御するコントローラからできています。データパスの中心はALU（Arithmetic Logic Unit)という加算、減算、論理演算などを行う組み合わせ回路と、レジスタといって数を記憶しておく装置の組み合わせでできています。ALUの出力をレジスタの入力に戻すことにより、一度計算した値を再び計算に利用できるようにします。データはメモリに入れておき、ここから読み出してきて、結果を記憶させます。これがCPUの基本的な構造です。この辺は、この授業で詳しく勉強します。

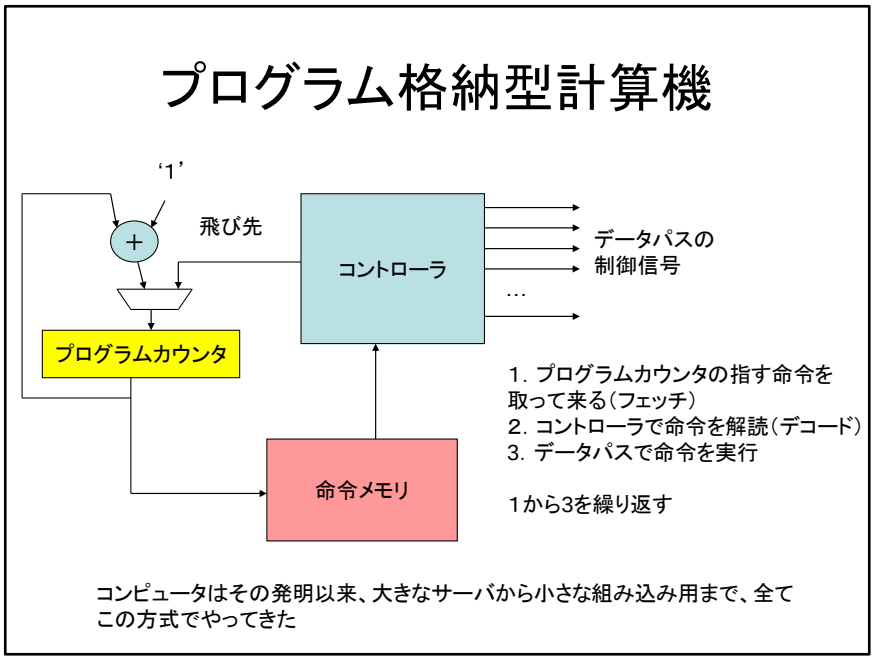
コンピュータの命令



演算命令: ADD R1,R2,R3
レジスタ2とレジスタ3を足して答をレジスタ1に入れろ

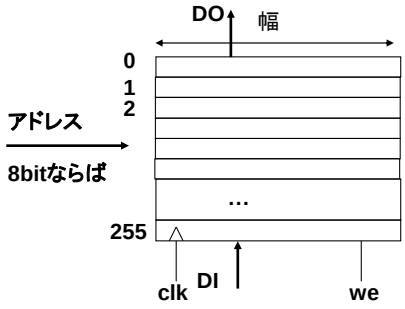
分岐命令: BEQZ R1,飛び先を示すコード
レジスタ1が0ならば、飛び先を示すコードから次の命令を実行せよ

コンピュータは、命令に従って処理を行います。この命令は、操作の内容を示すオプコードと操作の対象を示すオペランドの組み合わせからできています。例えばレジスタ2とレジスタ3の中身を足して答えをレジスタ1に入れろ。という命令はADD R1,R2,R3と書きます。ADDの部分がオプコード、R1,R2,R3の部分がオペランドに当たります。命令は基本的に書いてある順番に実行しますが、命令実行の順番を変える命令もあります。これを分岐命令と呼びます。例えば、レジスタ1が0ならば、順番に命令を実行するのを止めて飛び先と書いてあるところに相当する命令を実行します。コンピュータの命令をどのように作るかはこの授業で理解していただきたい重要な点です。



コンピュータは誕生以来70年くらい経っているのですが、プログラム格納型という方式が一貫して使われています。この方式はプログラムカウンタというレジスタを設けておき、このレジスタが示す命令を取ってきて、これを実行します。次にプログラムカウンタを先に進めて次の命令を実行しますが、分岐命令があれば、プログラムカウンタの中身を書き換えて別の命令を実行します。プログラム格納型の原理は大変簡単ですが強力です。現在でも組み込みコンピュータからスーパーコンピュータまで全てのコンピュータがプログラム格納型を使っています。

2）記憶装置（メモリ）



メモリは幅wbit, 深さ2^n
この例はw=16, n=8
深さ2^8 =256
（本当はもっとずっと多数のデータを格納する）

メモリのモデル＝基本的に表である。アドレスで示す番地にデータを保存する。

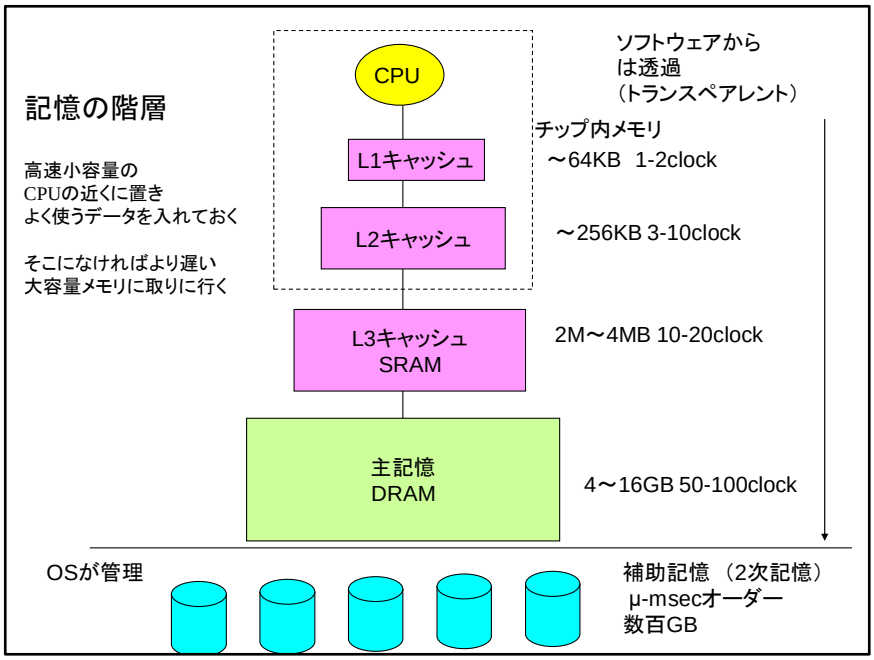
コンピュータの3つの要素のうちの2番目は、記憶システム（メモリシステム）です。メモリのモデルは単純な表で、アドレスという番地を与えて、対応する値を読んだり書いたりします。メモリのデータ幅wは、コンピュータで扱うビット幅に合わせて8ビットの倍数になっていることが多いです。アドレスの0番地から最大番地までを深さと呼びます。アドレスのビット数をnとするとメモリの深さは2のn乗になります。メモリの容量は2のn乗×w bitとなります。

メモリの容量

- 深さ×幅
- 右の表に幅を掛ければ全体の容量が出る
- 省略した言い方でも十分（端数を覚えている人は少ない）

アドレス本数	容量	省略した言い方
8	256	256
10	1024	1K
12	4096	4K
16	65536	64K
18	262144	256K
20	1048576	1M
24	16777216	16M
28	2683435456	256M
30	1073741824	1G
32	4294967296	4G

メモリの容量に限らず、コンピュータでは2のn乗の数を扱います。2の10乗は1024ですが、端数は省略して1Kと呼びます。同じく2の20乗は1M、2の30乗は1Gと呼びます。この表ではコンピュータでよく使う2のn乗の数が示してあります。このうち1K、1M、1Gだけは絶対に覚えておくのが便利です。



メモリシステムはCPUが読み出そうと思ったらすぐに読めるくらい高速で、無限に近いほど容量が大きいのが理想です。しかし、現実には高速のメモリは容量が小さく、容量の大きいメモリは遅いというジレンマがあります。そこで、これらのメモリをうまく組み合わせ、CPUに近いところに高速小容量のメモリを置き、良く使う命令やデータを入れておきます。ここになければ、これより遅いけれど容量が大きなメモリから取って来ます。この機構を記憶の階層と呼び、CPUに近い高速小容量のメモリをキャッシュと呼びます。キャッシュと記憶階層は、コンピュータの局所性と呼ぶ重要な性質に支えられています。この記憶の階層とキャッシュはこの授業の後半の目玉です。

メモリの分類

- 半導体メモリ
  - RAM （RWM)： 揮発性メモリ
    - 電源を切ると内容が消滅
    - SRAM(Static RAM)
    - DRAM(Dynamic RAM)
  - ROM(Read Only Memory)：不揮発性メモリ
    - 電源を切っても内容が保持
    - Mask ROM 書き換え不能
    - PROM(Programmable ROM) プログラム可
      - One Time PROM 一回のみ書き込める
      - Erasable PROM 消去、再書き込み可能
        - » UV EPROM （紫外線消去型）
        - » EEPROM（電気的消去可能型） フラッシュメモリ
- 磁性体メモリ
  - 磁気ディスク
  - 磁気テープ

メモリは大きく半導体メモリと磁性体メモリに分類されます。半導体メモリは、RAMとROMに分類されます。RAMとROMは本来の意味とはかなり違った使い方をされています。RAMはRandom Access Memoryの略で、アドレスに関わらずアクセスの方法と時間が同じものを指します。ROMはRead Only Memoryの略で読み出し専用メモリの意味です。しかし、最近ではRAMは揮発性メモリ、つまり電源を切るとデータが消えてしまうメモリ、ROMは不揮発性メモリ、すなわち電源を切ってもデータが消えないメモリの意味に使われます。

磁性体メモリは、磁気素子の上に記憶するので、不揮発性です。読み書きには大変時間が掛かりますが、膨大な情報を記憶することができます。

半導体メモリの例

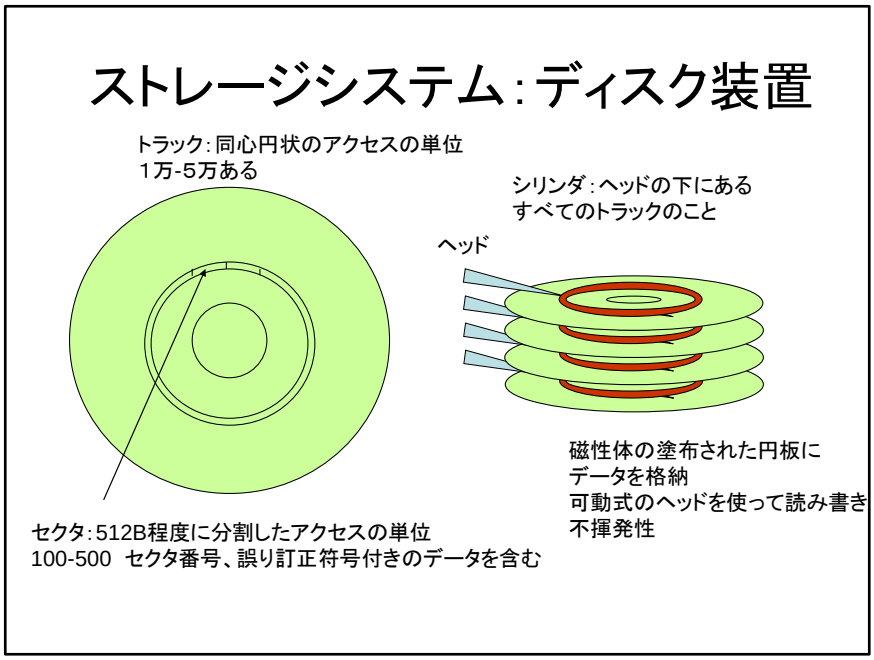
昔のSRAM、EPROM

DDR-3 SDRAM

フラッシュメモリを
使ったSDカード

フラッシュメモリを
使ったUSB
メモリ

おなじみのメモリもあると思います。このスライドは昔のRAM、今のパソコンで使われるSDRAM、写真データなどを入れるSDカード、USBメモリを示します。皆さんが良くご存知なのはフラッシュメモリでしょう。



ディスクは、磁気式メモリの代表です。最近はフラッシュメモリに押されてきているとはいえ、まだまだビッグデータ記憶の主力として活躍しています。ディスクは、磁気を塗布した円板で、表面上に磁気の形でデータを蓄えます。すなわち、ディスクは、電気的記憶媒体ではなく、磁気的記憶媒体です。データは一万から5万ある円周上に蓄えられ、この一周をトラックと呼びます。（陸上競技と同じです。）トラックは、100～500個程度の、512Bに分割したアクセスの単位に分割されます。これをセクタと呼びます。各セクタにはセクタ番号と誤り訂正符号付きのデータが含まれて居ます。ディスクは磁気的な記憶なので、読み書きするためには、磁気、電気変換を行う必要があります。これを行うのがヘッドです。ヘッドを伸び縮みさせ、円板を回転させることにより、ヘッドを読み書きしようとするデータの上に持ってきて、データを読んだり書いたりします。容量を大きくし、性能を上げるため、複数の円板を同軸上に回して複数のヘッダでアクセスする方法が一般的です。一時期にヘッドの下にある全てのトラックをまとめてシリンダと呼びます。100GB-1TBという大量のデータを記憶できますが、読み書きの時間は遅く、部分的に壊れ易い問題点があります。

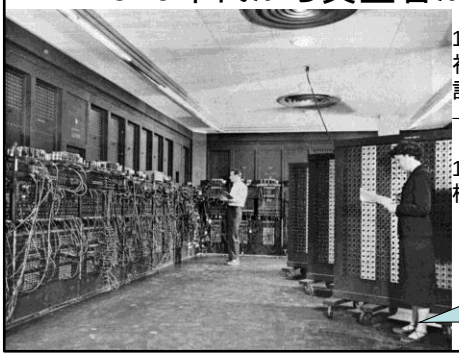
入出力装置　I/O

- コンピュータが外部と情報をやりとりするための装置
  - キーボード、ディスプレイ、マウス
  - ネットワーク
  - ディスクなどの補助記憶もI/Oとして接続される
- 標準的なインタフェースを使う
  - USB
  - PCIバス
  - PCI express

入出力装置は、コンピュータが外部と情報をやりとりするための装置で、実際上は非常に重要ですが、この授業では最後に一般的な事項だけやろうと思います。これは、I/O装置は相手によりけりな点が大きいためです。

コンピュータ略史
1. コンピュータ誕生

- 機械式の計算機
  - バベジの階差機関、解析機関が有名
- 1940年代から真空管が利用可能に

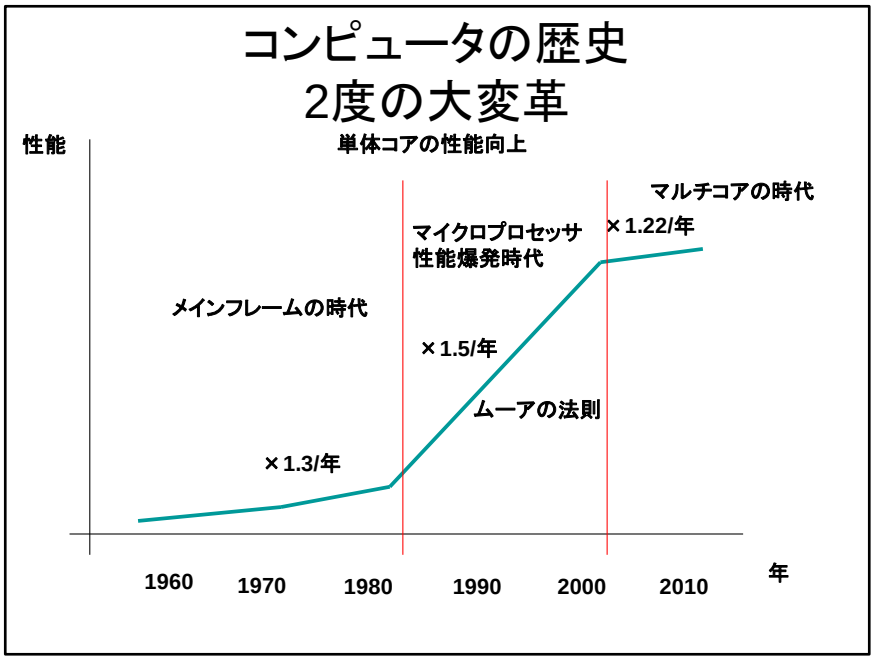


1946年に開発されたENIAC
初めて実用的に利用された電子式
計算機
→プログラム格納型の考え方がまとめられる

1949年のEDSAC：世界初の電子式プログラム
格納型計算機

ENIACは一部のプログラムを配線変更により行った。

ではここで、コンピュータの歴史を簡単に押さえておきましょう。自動的に計算を行う機械を作ろうとする試みは古くから行われており、中でもバベジの階差機関、解析機関はコンピュータの原理に通じるものがありました。1940年代に真空管が使えるようになると、電子式な計算機を作る試みが各地で行われました。このうちどれを世界最初の電子式コンピュータとするかは、諸説あるのですが、1946年に米国で稼働したENIACは、始めて実際に業務で用いられたコンピュータとして有名です。ENIACは、一部のプログラムを配線変更により行ったため、現在のコンピュータとは動作原理が違っていました。1949年、英国で稼働したEDSACは世界初の電子式プログラム格納型コンピュータであり、この方式が今でも使われています。



コンピュータの性能向上の概略を示します。縦軸は対数表示です。ここで、1980年代のメインフレームからPCへの変革、2003年に起きたマルチコア革命の二つの変化が重要です。

コンピュータ略史

(2)メインフレームの時代

1950年→1980年初頭

- 素子の発達により小型化、複雑化が進む
  - 真空管(第1世代)→トランジスタ(第2世代)→集積回路(第3世代)
- 事務計算、科学技術計算用に普及
  - 企業、大学単位で設置
  - パンチカードによるバッチ処理→ 端末を使った処理TSS(Time Sharing System)に
  - OS、プログラミング環境の発展
  - ミニコンピュータ、スーパーコンピュータ等目的別に分化が進む

コンピュータの基本的な動作原理はプログラム格納型でEDSACより変わっていません。しかし、性能は、基準とするコンピュータによりますが、現在の普通のPCは、EDSACの1億倍以上の計算能力を持っています。(EDSACは水銀遅延線をメモリに使っていたため実行できる命令は1秒間に650回でした。普通のPCでは100億以上の命令を実行することができます)全く基本方式が同じのままで性能の増加とコストの低下がすざまじい点でコンピュータは工業製品としてはかなり特徴的であると言えます。このコンピュータの進歩は、登場直後から始まり、第一世代の真空管から第二世代のトランジスタ、さらに第三世代の集積回路と次々に新しいデバイスを使い、世代の交代を進めていきました。

コンピュータの登場からしばらくは、高価であったため、メインフレームと呼ばれる大型コンピュータを皆が共有で使いました。このメインフレームは、企業や大学単位で設置され、事務計算、科学技術計算に用いられました。当初は、パンチカードを使ってプログラムを打ち込み、しばらく待つと結果がプリンタから出力されるバッチ処理と呼ばれる方法で使われました。この環境は、端末を使ったTSS(Time Sharing System)に置き換わりました。メインフレームは、専用で使うことができなかったため、研究者にとっては使い難い存在でした。このため、占有して使えるミニコンピュータが登場しました。一方、高速な科学計算用にスーパーコンピュータも登場して、コンピュータの目的別分化が進みました。

コンピュータ略史

(3)コンピュータ大発展時代

1980年中ごろ→2003年

- 個人が使うパーソナルコンピュータ、ワークステーションへ急激に移行
  - インターネットの発達
  - Ethernetの普及
  - 標準OS（Windows,Linux）の普及
  - 半導体技術の発展
  - RISC（Reduced Instruction Set Computer)とこれに伴う高速化、パイプライン処理、スーパースカラ型
  - マイクロプロセッサの猛烈な性能向上

  → 年間1.5倍に性能が向上（ムーアの法則）

  動作周波数は3GHzに達する
- コンピュータはあらゆる場所で使われ、なくてはならないものとなる

1980年代になると、コンピュータのCPUが1つの半導体素子に収まるようになり、マイクロプロセッサが発展しました。半導体素子自体の発達に加えて、RISCの登場とパイプライン処理、命令レベル並列処理など、この授業で勉強する高速化手法が発達し、全体として年間1.5倍（18ヶ月で倍）に近いくらいの勢いで性能が上がるようになりました。これをムーアの法則と呼びます。これに、インターネットの発達、標準OSの普及、Ethernetの普及などが加わり、コンピュータの使われ方に大変革が起きました。今までの大型計算機を多数で共同で使う方法から、個人がパーソナルコンピュータを使い、それらがローカルコンピュータネットワークで接続され、インターネットを経由して世界中のコンピュータが接続され、情報を交換し、Webを閲覧するという現在の使われ方に以降したのです。コンピュータは家庭にも進出し、あらゆる電化製品に組み込まれて、世の中のあらゆるところで使われるようになりました。

コンピュータ略史

（4）マルチコア時代

2003年→現在

- 単体CPUの性能が限界に達する
  - 発熱、消費電力の問題
  - 同時に実行できる命令数が限界に
  - CPUのスピードにメモリが付いていけない
  - 半導体の性能向上の限界
- 周波数を上げるのではなくCPU（コア）の数を増やす→マルチコア
  - GPU（Graphic Processing Unit)などのメニーコアによる高速化も普及
- パーソナルコンピュータからタブレット、スマートフォンにコンピュータの主な使われ方が移りつつある

このコンピュータの性能爆発時代は、2003年まで続きました。この間コンピュータの動作クロック周波数（授業中に説明します）は数MHzから3GHzまで上昇しました。しかしここで再び転機が訪れます。あまりにクロック周波数が上昇したため、コンピュータで消費する電力が増加し、熱を発散させるのが困難になってきたのです。さらに、CPUの中で行う高速化手法が限界に達し、CPUに比べてメモリの速度が上がらないなどの問題点のため、単体のCPUの性能を向上させるのが困難になってきました。2003年にマイクロプロセッサを主導してきたIntel社はその方針を変更し、単体のCPUの性能を向上させるのはもう止めて、一つの半導体の中のCPU（コア）の数を増やすことによって性能の向上を目指すことを宣言しました。マルチコア時代の到来です。現在、みなさんのお使いのノートPCには4－6個のコアが内蔵されています。また、コンピュータの利用はスマートフォン、タブレットと、大規模なデータセンターによるクラウドコンピューティングが主体となり、80年代以来、コンピュータの中心であったPCが衰退しています。またGPUなど新しいタイプのコンピュータも現れ、コンピュータはますます分化し、拡散して居ます。皆さんは、1980年代のメインフレームからPCへの変革、2003年に起きたマルチコア革命の二つの変化を頭に入れておいてください。

ハードウェア記述言語

- HDL (Hardware Description Language)
- 「計算機基礎」で習ったゲート接続図を使ったハードウェア設計は今は使われない
  - スケマティック設計と呼ばれる
- Verilog-HDLとVHDLの二つが標準
  - 最近は多くのCAD (Computer Aided Design)が両方を受け付ける
  - CADによる論理合成、圧縮によってゲート接続図（ネットリスト）に変換される
- 最近はCレベル設計も一般化
  - HLS(High Level Synthesis:高位合成）と呼ばれる方法でハードウェアを合成する
  - 用途によって使い分けられている

この授業では、CPUをハードウェア記述言語で設計し、シミュレーションしながら、その動作を理解します。以降、今日はハードウェア記述言語を概観し、シミュレーションの方法、波形の観測の仕方を勉強します。ハードウェア記述言語は二つの勢力が拮抗しています。Verilog HDLとVHDLです。Verilog HDLの後継言語のSystem Verilogも、多少は使われ始めていますが、まだ従来のVerilogの方が良く使われます。この二つの言語は、データの記憶と流れ、その間に行われる処理に着目して、プログラミング言語に似た言語を使ってハードウェアを記述します。このような記述をRTL（Register Transfer Level)と呼びます。RTL設計されたハードウェアは、シミュレーションにより動作を確認した後、CAD(Computer Aided Design)を用いてゲート接続図（ネットリストと呼びます）の形に変換し、最適化を行います。最近はC言語とほとんど同じ（制約つき）言語を使って、ハードウェアで行う処理自体を記述すると、RTL記述に自動的に変換してくれるHLS（High Level Synthesis)技術が発展してきたため、ハードウェアをC言語（Javaで設計する方法もある）で記述する方法が一般的に普及してきました。この方法は複雑なアルゴリズムをハードウェア化するのには便利ですが、CPUやネットワークコントローラなどタイミング制御が重要なハードウェアの記述には向いていません。用途によって使い分けられています。

VerilogとVHDL

	Verilog-HDL	VHDL
出自	論理シミュレーション記述	仕様書
標準化	デファクトスタンダード	国際標準
記述	Pacal風（嘘）	PL/I→ADA
特徴	広い範囲でシミュレーションは可能	記述が厳格

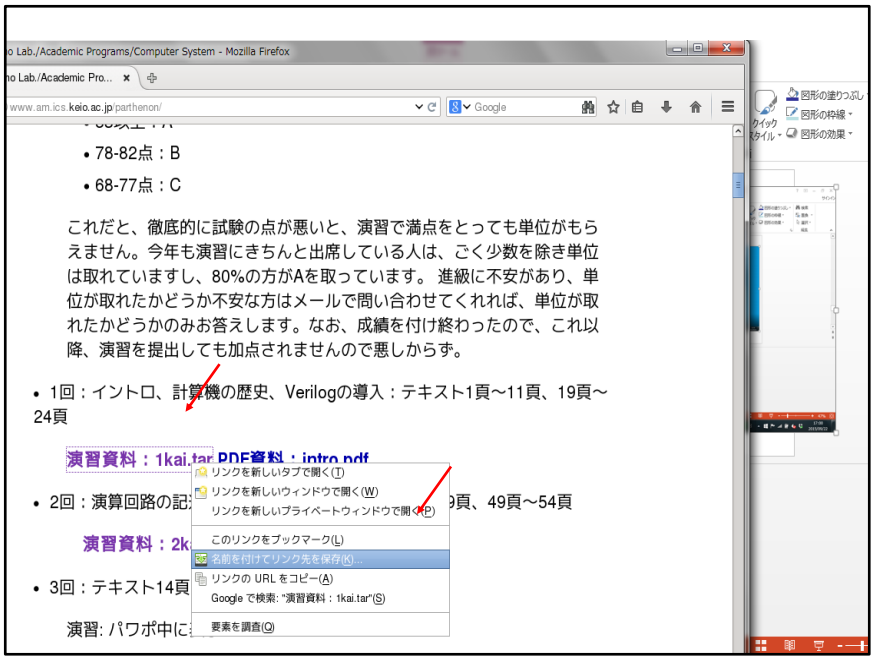
情報工学科ではVerilog HDLを採用

Verilog HDLとVHDLは共に良く使われますが、その性格は全く違っています。

Verilog-HDLは論理シミュレーションを記述する言語として誕生し、使われているうちに標準化されたデファクトスタンダードです。構文（シンタックス）はPascal風と称していますが、ウソで、Pascalとは結構違った独自の構文です。シミュレーションが動作すれば、そのハードウェアの様子が分かるので、とにかく動作することが重要です。このため、Verilog HDLは、宣言などをいい加減に書いてもシミュレーション上は動作してしまいます。これはありがたいことなんですが、合成の段階までに問題点に気づかないと、とんでもないバグを含んだハードウェアを生成することになります。Verilog HDLは、文法上不合理な点を色々持っているため、これを改良したSystem Verilogが登場しています。しかし、ツールが対応しないので、ここでは使いません。ここで使う文法の範囲ならばVerilogもSystem Verilogもさほど変わりません。

一方、VHDLはハードウェアの仕様書記述用の言語が発達したもので、厳格な構文を持ちエラーチェックをきっちり行います。米国国防省が制定したPL/I、ADAの流れを汲む構文で、国際標準としてトップダウンに制定されました（PL/IとADAは、米国国防省がこれ以外の言語で書いたソフトウェアの納入を認めなかったため、プログラミング言語として一時期相当使われました。しかし、あまりの融通の利かなさに腹を立てたプログラマたちは、長年の苦闘の末にこれを絶滅に追い込みました）。記述が厳格なので、シンタックスエラーを修正する段階でバグをかなり減らすことができます。一方、簡単なハードウェアの記述にも多数の行数を要するので不便です。

本大学の情報工学科ではVerilog　HDLを使います。僕は最初はVHDLを使っていた（使っていたCADがこれしか受け付けなかった）のですが、この言語があまり好きになれず、Verilog HDLに切り替えました。VHDLでハードウェアを設計していると、新しいシステムを設計するんだ、というクリエイティブなことをやっているのではなくて、わかりきったシステムの仕様書を書いているような気分になってくるんです。もちろん、なんで書こうとクリエイティブな設計はできるのですが、ま、気分の問題です。幸いにして最近のCADは両方を受け付けてくれますので、どちらかで設計できれば問題ありません。

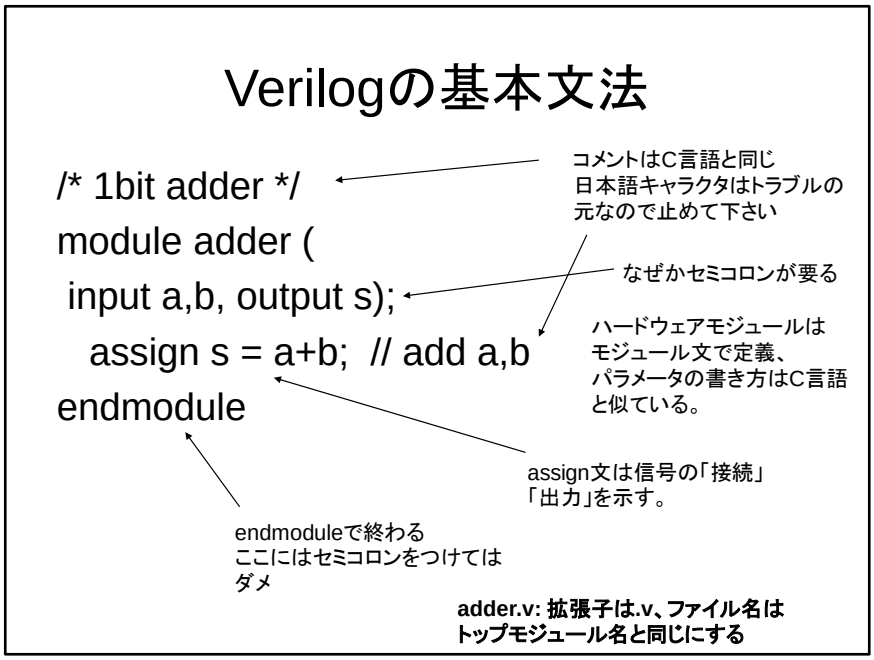


このページは、例によって用語解説やVerilogの文法の解説が載っていて便利です。この授業のテキストのページと合わせて活用してください。今日はこのうちの授業資料の1kai.tarを右クリックして、リンク先を指定してダウンロードします。作業用にこの授業専門のディレクトリを作り（mkdir systemとかやる）、そこにダウンロードすることをお勧めします。

tar fileの解凍

- cd ディレクトリ名
- tar xvf 1kai.tar
- cd 1kai
- ls

次にこのダウンロードされたファイルを解凍します。このファイルはtarという由緒正しいLinuxのアーカイブ（書庫、ファイルをまとめて一つにするやり方）形式になっています。（なんてったってtarのtはtapeでテープ時代から使われてたものです）ダウンロード先のディレクトリに行き、tar xvf 1kai.tarというコマンドで解凍します。そうすると1kaiというディレクトリが出来てくるので、このディレクトリに入ります。以降、解説する演習用ファイルもこのディレクトリに入っています。



では、演習用ディレクトリ中のadder.vを見てみましょう。これがVerilog HDLで書いた1ビットの加算器です。C言語のプログラムの拡張子を.cにしたのと同様に、Verilog HDLのファイルの拡張子は.vにします。また、ファイル名はトップモジュール名（最上位階層のモジュール名、ここではadder）にします。エディタは皆さんの好きなのを使ってください。僕はviを使いますが、皆さんはemacsがお好きな方が多いかと思います。

さて、まず最初の行はコメントです。コメントの付け方はCと同じで/* */で囲むか、//の後に書くかどちらかです。ここで日本語を使いたくなる人が居ると思いますが、日本語のフォントがトラブルの元となるので止めてください。英語でコメントを付けましょう。さて、Verilog HDLは（VHDLも）、ハードウェアをモジュールという単位で階層的に記述していきます。この場合1ビットの加算器が1つのモジュールになります。Verilogの記述はモジュールの定義から始まります。

module文の後にモジュール名を書き、これに続く（ ）内に入出力端子を定義してやることでモジュールが定義されます。ここではadderがモジュール名で、inputの後のa,bが入力端子名、outputの後のsが出力端子名です。後に紹介する方法で指定しない場合は1ビットの端子として宣言されます。

これらはカンマで区切っていくつでも並べて書くことができます。input文、output文自体が複数出てきても問題ありません。ここでC言語との違いは、なぜか最後の ）の後にセミコロンが必要な点です。

モジュールを定義したら、次からの文はそのモジュールの構造あるいは動作を書きます。ここではassign s=a+b;でa入力とb入力の加算結果をsに出力する、あるいはaとbを加算器に入れた出力をsに接続する、という意味があります。Verilogでは単純に＝を使うことはできず、必ずassignを先に付けます。（これについては後に詳しく解説します。）文の終わりはC言語と同様にセミコロンを付けます。
このモジュールはこれで終わりなので、endmodule文でモジュールの終わりを宣言します。この文の終わりにはセミコロンを付けてはいけません。

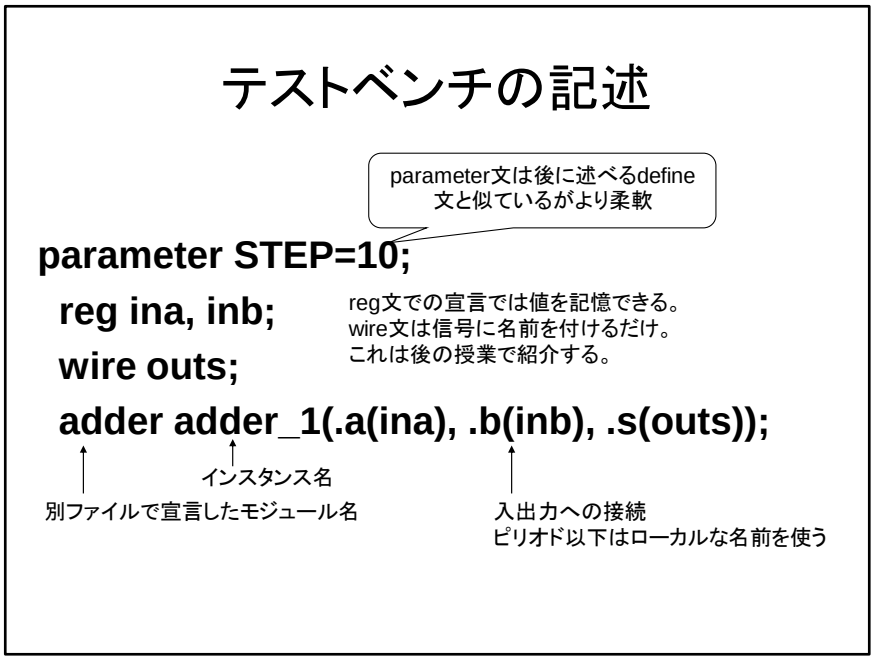
テストベンチ(test.v)

- シミュレーション制御のための記述

```
module test;
  parameter STEP=10;
  reg ina, inb;
  wire outs;
  adder adder_1(.a(ina), .b(inb), .s(outs));
  initial begin
    $dumpfile("adder.vcd");
    $dumpvars(0,adder_1);
    ina <= 1'b0;
    inb <= 1'b0;
  #STEP
    $display("a:%b b:%b s:%b", ina,inb,outs);
    ina <= 1'b0;
    inb <= 1'b1;
 #STEP
...
```

adder.vを記述したらこれをシミュレーションしてテストする必要があり、このために別のモジュールが必要になります。このモジュールは、実際のハードウェアではなく、特定のモジュールをテストするだけの目的の記述で、テストベンチと呼ばれます。ここではmodule testをtest.vというファイル中に記述してあります。これをエディタで開いて見て下さい。テストベンチはそれ自体の入力はないので、モジュール名の後ろの（ ）はありません。

Verilogは、論理シミュレーションの制御用の言語から発達したので、シミュレーション用の記述が充実しています。これは裏を返せばやたらにある機能を使わないと簡単な回路のシミュレーションができないことで、このテストベンチを書くことが入門者の壁になっています。ただし、テストベンチは基本的に標準パターンしか使わないので、あまり深く意味を考えず、シミュレーションをしてみましょう。ここで簡単に説明しますが、あまり深く突っ込まない方がいいです。



まず、最初にparameter文でシミュレーションの実行の1ステップの時間を定義しています。ここでは10nsecがシミュレーションの単位時間である旨を宣言しています。このSTEPという記号を使ってシミュレーションの時間を進めて行きます。次にina, inb, outsというテストベンチ中の信号名を定義しています。ここでregはregisterの略で、データを記憶します。これに対してwireは単に信号に名前をつけているだけです。この辺は後の授業でよく説明しますので、ここではあまり触れないでおきます。さて、次にadderから始まる行で、adder.vで宣言したadderの実体を生成します。ここで、モジュール名は先ほど定義したadderを使う必要があります。これに対して、実体名は何でもいいのですが、ここではadder_1という名前にします。実体名の後の（ ）の中で、モジュールの入力にテストベンチの信号を割り当てます。ここではaにinaをbにinbを、sにoutsを繋いでいます。モジュールの入出力名にはピリオドを付けてその後の（ ）内に上の階層の信号名を書いてやります。信号名が直接指定されているので、書く順番はどうでも良くなっています。

シミュレーションの制御

initial文はシミュレーションを一回実行

```
initial begin
   $dumpfile("adder.vcd");   波形ファイルを指定
   $dumpvars(0,adder_1);     記録する範囲を指定
   ina <= 1'b0;
   inb <= 1'b0;              reg文にはブロッキング代入<=
                             （これも後に紹介する）
                             ここでは入力を制御
  #STEP  時間消費
   $display("a:%b b:%b s:%b", ina,inb,outs);
   ina <= 1'b0;              値の表示、プリント文と似ている
   inb <= 1'b1;              %bで2進数表示
                             リターンは自動的に入る
 #STEP
```

initial文以下はシミュレーションの制御を行う部分です。この文はシミュレーションを順に一回実行します。beginからはじまってendで終わります。最初に＄dumpfileと＄dumpvarsで、シミュレーション結果を記録する波形ファイルと、そこに記録する信号の範囲を指定します。Verilogでは＄から始まるのは標準関数です。この授業では波形ファイルはvcd形式を使います。この形式はファイルサイズが大きくなる欠点があるのですが、簡単で昔から使われているのでほとんど全てのCADで扱ってくれます。拡張子にはvcdを付けてください。ファイル名の部分は何でもいいのですが、adderをテストするのでadderという名前にしています。次のdumpvarsは記録する信号の範囲を示し、ここでは実体名がなければなりません。今回adder_1を指定してこのモジュールの信号を全て記録します。最初の0は全て記録することを示します。次に入力信号に値を設定します。ina, inbはregで宣言したので、値を覚えておいてくれます。これに＜＝で0、0を設定します。この＜＝はブロッキング代入文といい、レジスタに値を設定するときに使います。これも後ほど詳しく紹介します。それから＃STEPで10nsec時間を進めます。それから$display文で結果を表示します。このdisplay文はC言語のprintfとほとんど同じですが、%bというフォーマットで、2進数の表示ができます。また、改行は自動的に入ります。これで入力が0，0の時の入出力信号が表示されます。また、＃STEPでシミュレーション時間を進め、全ての組み合わせの入力を入れて結果を表示し、最後に＄finihshでシミュレーションを終了します。その後にinitial文のbeginに対応するendを書きます。

テストベンチもモジュールの一つなので最後はendmodule文が必要です。

Verilog-HDLのシミュレーション

- Ikarus Verilogを利用
  - ○コンパイル型のフリーソフトウェア
    - Linux, Windowsマシンにインストール可能
    - iverilog XX.vでコンパイル、必要なファイルを全部書く
    - vvp a.out（あるいは単に./a.out)で実行、かなり高速
  - ○Verilog2000に対応
  - ×遅延付シミュレーションができない
- 波形Viewerはgtkwave
  - ○フリーソフトウェア
    - Linux,Windowsマシンにインストール可能
    - gtkwave XX.vcdで起動
    - 基本的なViewerの機能は全て持つ
  - ×他のViewerに比べて少し使いにくいかも、、

さて、では今まで紹介したテストベンチtest.vを用いてadder.vをシミュレーションしてみましょう。ここで使うのはIkarus Verilogというフリーソフトウェアを利用します。Ikarus Verilogはコンパイル型のシミュレータで、まずiverilog test.v adder.vと打ち込んでコンパイルします。C言語同様、必要とするファイルを全て並べます。シンタックスエラーがなければa.outという実行形ができます。ここで、vvp a.outというコマンドを打ち込むことによりシミュレーションができます。ありおはC言語同様、./a.outでも実行できます。フリーソフトウェアにしては高速で、元々のVerilogだけでなく改訂版のVerilog2000にも対応してくれます。Linux, Windows両方に対応するので、皆さんのPCにインストールすることも可能です。「作りながら学ぶコンピュータアーキテクチャのページ」にサイトが紹介されています。

ではやってみよう

iverilog test.v adder.v

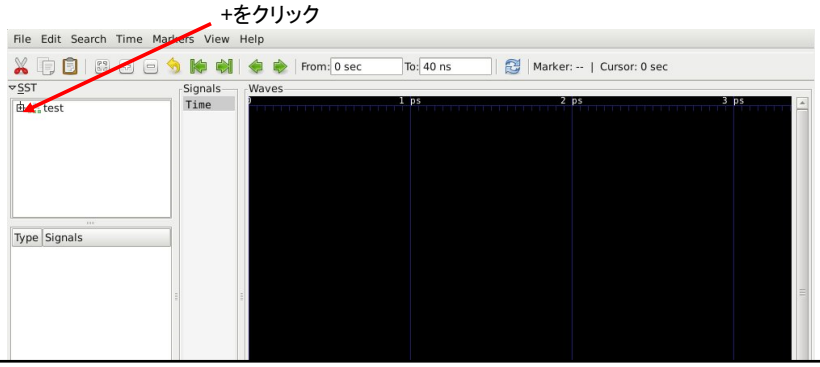
./a.out

結果が表示される

gtkwaveを使って見よう

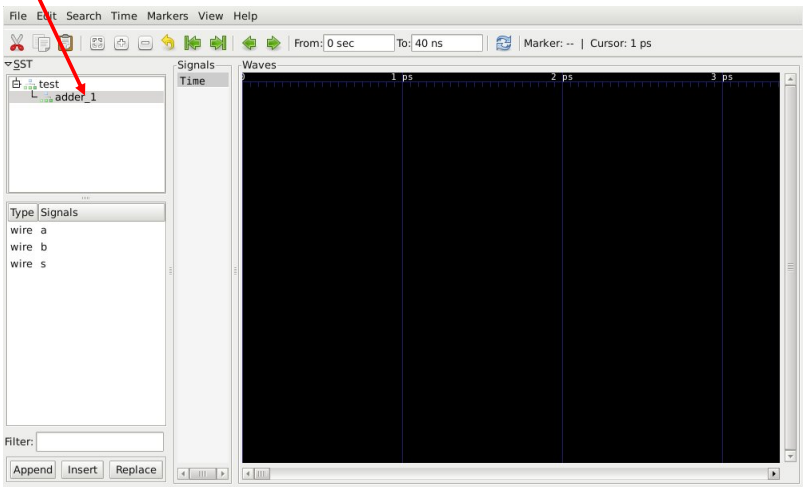
- gtkwaveは強力なデバッグ用ツール
- これなしではとても演習は乗り切れない！

gtkwave adder.vcd

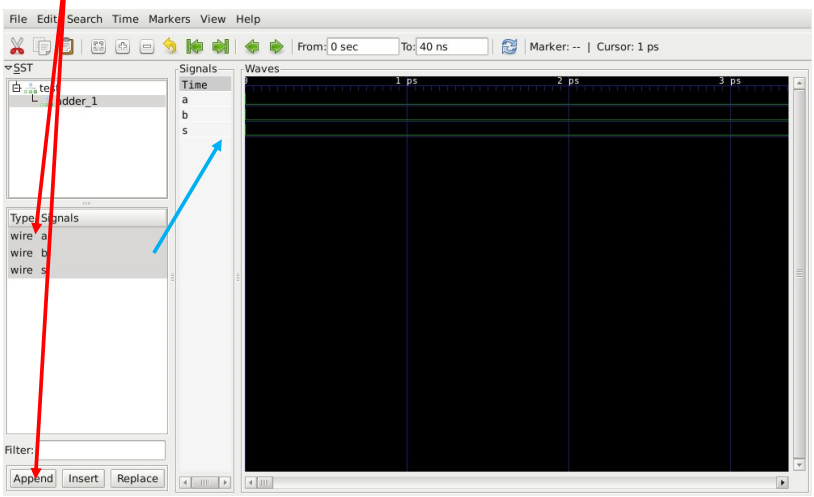


gtkwaveは波形ビュアー、すなわちシミュレーション結果を波形として観測するツールです。大変強力なツールなので、ぜひ使いこなせるようにしてください。

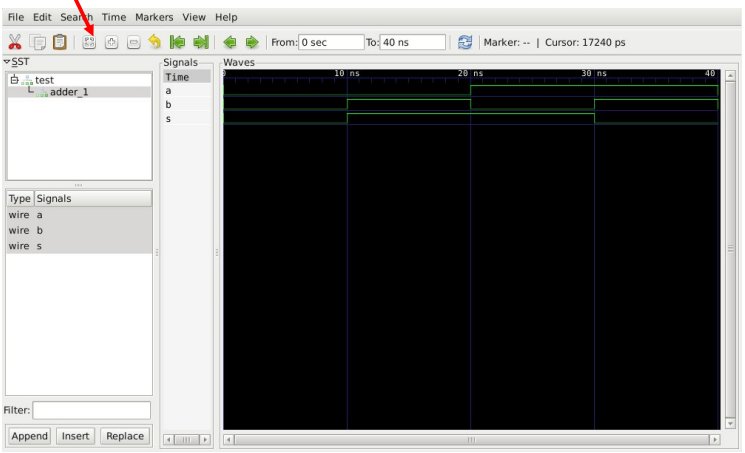
モジュール名をクリックすると信号名が下の窓に表れる



信号を選択してAppendを押すと右の窓に波形が表れる
信号名をDrag-and-Dropしてもいい



Zoom Fitを押すといい感じに表示される
ー印（Zoom Out)を連打してもいい



演習

加算を論理AND（＆）に置き換えてシミュレーションを実行しよう

andは予約語なので注意　module anという名前にし、an.vというファイルに入れる

test.vを入れ替える。adderではなくanに変更

Webからtar fileを取って来る

tar xvf 1kai.tarで解凍

提出は　keio.jpに提出

では演習をやってみましょう。今日は非常にちょろい演習で、加算の＋を論理積の＆に入れ替えてシミュレーションを実行するという課題です。演習の提出はkeio.jpに行いますのでご注意ください。

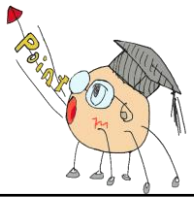
コンピュータの基礎　まとめ

- コンピュータはCPU,メモリ、I/Oの3要素からできている
- コンピュータは色々なクラスがあって、性能、価格が10億倍（いい加減）も違うけれど、どれもほとんど同じ命令体系で動く
- コンピュータはできてから75年くらいの歴史を持つが基本方式はプログラム格納型で変わっていない。しかし性能は1億倍（いい加減）くらい良くなっている
- コンピュータの歴史で重要なのは1980年代の使い方の革命と、2003年のマルチコア革命。今はマルチコア時代が続いている。

ではインフォ丸によるまとめです。コンピュータの基礎はこの４点を覚えておいていいでしょう。コンピュータは非常に特殊な工業製品といえます。

HDLのまとめ

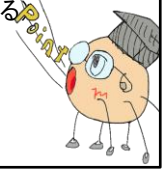
- HDLとは、レジスタに対するデータの記憶と、レジスタ間のデータの流れをプログラム言語風に記述する（RTL設計）ハードウェア設計用言語
- Verilog HDLはmodule単位でハードウェアを記述する
  - module/endmodule
  - input/output
  - assign
- テストベンチはシミュレーションのやり方を記述する
  - initial
  - $display
  - $dumpfile
  - $dumpvars
  - $finish



次はHDLのまとめです。

Verilogシミュレーションの実行

- シミュレーションのコンパイル
  - iverilog *.v
  - ディレクトリ内に同一モジュールがある時は、ファイルを全て指定する
  - iverilog test_poco.v poco1.v rfile.v alu.v など
  - エラーが出た場合、メッセージを良く読んで！
- シミュレーションの実行
  - vvp a.out　あるいは./a.out
  - iverilog 実行時に-oで実行ファイル名を指定することができる。
- 波形の表示
  - gtkwave XX.vcd
  - モジュールを選択すると信号名が表示される
  - これをクリックして選択→Appendをクリックすると波形が表示される
  - スケールがpsecなのでマイナス（－）をクリックしまくってスケールを調整（一番左のZoomFitをクリックすると自動的に調整してくれる）



最後は演習実行の方法のまとめです。これは、この授業で何度も使います。