■談話室

なにか面白いデバイスないですか？ —退職教員回顧録—

元　東京大学素粒子物理国際研究センター
坂 本　宏
sakamoto@icepp.s.u-tokyo.ac.jp

2025 年（令和 7 年）8 月 1 日

1 はじめに

早いもので，定年退職から 7 年が過ぎました。いろいろなことを忘れかけてしまっていると思っていたところ, 高エネルギーニュースに回顧録を書いてくれないかとのお誘いをいただきました。これを機会に自分なりの回顧録をしたためることにしました。思えばいろいろな電子回路技術や計算機技術と格闘してきたものだと思います。実際, 自分の研究生活はディジタルエレクトロニクスやコンピュータの発展と共にあったように思うのです。

2 SN74 シリーズとの出会い

私が最初に出会ったディジタルエレクトロニクスデバイスは SN74 シリーズ[1]と呼ばれる論理素子, TTL (Transistor-Transistor Logic)です。SSI (Small Scale Integration)とか MSI (Medium Scale Integration)などの集積回路のカテゴリーで, 例えば論理演算回路 NAND が数個のバイポーラトランジスタから構成され, 1 つのチップに 4 回路入った SN7400 などです。修士課程に進学した 70 年代後半には標準的なデバイスでした。実験装置の制御をするのに, 論理回路をまず設計し，対応する論理素子をシリーズから選択し，100mil (2.54mm)ピッチで縦横にホールが開いたユニバーサル基板に半田付けします。配線は錫引き線を繋いでいきます。どんな大規模な装置もこういった論理素子から作り上げることが出来るというディジタルエレクトロニクスの神髄を体験しました。銅箔の張られたベーク基板に配線パターンをテープなどで貼って, エッチングで不要な銅箔を溶かし, 出来たパターンにドリルで部品用の穴をあけて回路基板を作ったりしました。この頃は手作りで何でも出来た時代でした。

3 Z80 との出会い

同じ時期に初期の 8 ビット汎用マイクロプロセッサ Z80 にも出会います。いわゆるパソコンが初めて登場した時期で, 研究室で米国タンディラジオシャックの TRS80[2]を買ってもらいました。当時で 40 万円くらいしたと思います。プログラミング言語は BASIC が搭載されており，8 ビットの汎用入出力ポートに上述した SN74 シリーズで作成した回路を繋いで遊んでいました。外部記憶はカセットテープで, 作成したプログラムを書いたり読んだりしていました。作成した電子回路がプログラミングによりどのようにでも働かせることが出来るソフトウエアの威力を知りました。

4 CAMAC との出会い

修士課程では小さな加速器で実験していてデータは MCA (Multi Channel Analyzer)とか PHA (Pulse Height Analyzer)とか出来合いの装置で取得していました。博士後期からは実験場所を変え，大型のサイクロトロンを使うようになりました。複数の検出器を使うことが普通で, それらの信号を NIM モジュールで処理した後, CAMAC[3]モジュールを経由して次に述べるミニコンピュータ PDP11[4]でデータを収集するシステムになっていました。モジュールの中に汎用の入出力ポートを持つものが有り, 実験情報の一部を, それを通して読み込ませたりしました。標準化された計算機インターフェースがあることで実験データの収集が, 限られた最小の労力で実現することを知りました。CAMAC は TTL ロジックの初期の仕様にあわせてあり, 非常にシンプルなデザインで初心者でもモジュールが製作可能でした。

5 PDP11 との出会い

CAMAC インターフェースのホストコンピュータは当初 PDP11/40 でした。16 ビットのミニコンピュータで 28k ワード(1 ワードは 16 ビット)のメモリーを搭載していました。OS は RT11[5]というシンプルなリアルタイム OS で，プログラミング言語はアセンブラーでした。計算機本体の表面パネルには赤と紫のスイッチが並んでいて, ブートストラップコード(最初に実行されるプログラム)を手で入力したこともありました。PDP11 のアーキテクチャと命令セットはきわめて体系的で覚えやすかった。8 本の汎用レジスタ（プログラムカウンターを含む），8 つのアドレッシングモードがそれぞれ 3 ビットで表現され，(3+3)*2 ビットで 2 つの引数, 残り 4 ビットで命令の種類を表していました。アセンブラー

でのプログラミングを通して, コンピュータの構造と動作をより深く理解することが出来るようになりました。

6 Fastbus との出会い

大学院を終えてしばらくし, つくばの高エネルギー物理学研究所に就職し TRISTAN 加速器 VENUS 実験に参加することになります。TRISTAN 加速器はアジアで最初の電子陽電子衝突型加速器です。非常に多くの検出器が用意され, 大量のデータを取得・処理する必要があります。それらをどうやって計算機に取り込むかを考えたとき, それまでにあった CAMAC では 1 ワード読込に 1 マイクロ秒必要など能力が不足します。そのためより高速にデータを読み込める規格が必要でした。そこで採用されたのが Fastbus[6]でした。当時最も高速な論理素子であった ECL (Emitter Coupled Logic)[7]を基本技術に採用していたため，なじみのある TTL とのレベル変換に基板上の広い面積が喰われる問題がありました。また, 非常に大規模なマルチプロセッサシステムを構築することを念頭に複雑なプロトコルが規定されており, その理念を実現したシステムは最後まで作られることはなかったのではないでしょうか。

7 TKO との出会い

Fastbus ですべての検出器データを読み出すことは現実的ではないとして，VENUS では TKO(Tristan KEK Online)というローカルな（世界基準ではない）規格[8]が採用されました。CAMAC に比べて十分高速大容量で, モジュールの面積も広く, プロトコルもデータ収集に必要最小限の簡素なものでした。データ転送だけでなく, タイミング信号や閾値電圧もバックプレーンで供給されます。検出器信号などかさばる入力は背面から供給されているので, モジュールの脱着に信号ケーブルを外す必要がない，より信頼性が高く保守性の良い設計でした。衝突型加速器実験で必要とされる機能を網羅した良い規格でした。特に勉強になったことは, 規格策定の過程を詳細に聞くことが出来たことです。それまでの加速器実験の経験を元に議論を重ね，必要十分なものに規格を収束させる過程は非常に教育的であったと感じました。

8 VAX11 との出会い

VENUS のオンライン計算機は VAX11/780[9]という, 当時最先端を行く 32 ビットミニコンピュータでした。OS は VAX-VMS で，仮想メモリーが使えました。プログラム中で使われるメモリーのアドレス（論理アドレス）を，メモリー管理機能が自動的に実在するメモリーのアドレス(物理メモリー)に変換してくれます。空いている物理メモリーを次々に論理アドレスにマップして行きます。プログラマーは実装された物理メモリーのサイズを気にせず, 巨大なアドレス空間を使用できます。実メモリーにプログラムを押し込む努力から解放されたことは素晴らしかった。また，VAX FORTRAN[10]という，従来の FORTRAN とはかけ離れて自由なプログラミングが出来る環境で作業が出来ました。ソフトウエアの生産性を高めることの意味を理解しました。

9 PC98 との出会い

いわゆるパソコンも普及してきました。Intel 社の 8086 系の CPU を搭載した PC9800 シリーズ[11]のパソコンは個人にも普及し，自宅から RS232C モデムを経由して研究所の計算機にログインできるようにもなりました。MS-DOS[12]を搭載しており，別売りでしたが C 言語でのプログラミングが出来ました。C 言語はやや高級言語という位置づけで, 文法が機械語を強く意識したものでした。ポインターはアセンブラーでのレジスタ間接参照そのものでした。入出力がすべて関数で表現され，特別な入出力命令を持たないことなどもすっきりしていました。アセンブラープログラミングの経験を引き継いだ形でオンラインプログラムが書ける, 強力な言語を手に入れた感動がありました。PC98 上で動く，VT100 というシリアル端末のエミュレータプログラムを作りました。

10 Transputer との出会い

この時期に見つけた面白い技術が Transputer[13]でした。3cm×10cm ほどの小さな基板にプロセッサとメモリー, 4 本の 5Mbps 高速シリアルリンクが搭載されており, Occam2[14]という言語でパラレルプログラミングが出来ました。これを使うと簡単にパラレルプロセッシングシステムが構築できました。格子状に並べたチップが上下左右にリンクで接続されており, プログラム自体もそのリンクを経由してそれぞれのモジュールにダウンロードされます。Occam2 ではリンクからの入力（linkin?var1 のように表現）とリンクへの出力（linkout!var1）が変数への代入（var1=123）と並んでプログラムの基本要素となっており，並列処理が明示的に記述できます。それまでのシーケンシャルなプログラミングと違い, 複数のプロセスが通信しながら同時に処理を行い, 全体として一つのタスクを遂行する, パラレルプログラミングのダイナミックさを経験しました。VENUS 検出器の中央ドリフトチェンバーのリアルタイムイベントディスプレイなど作って楽しんでいました。

11 VMEbus との出会い

VENUS 実験に従事する傍ら, 次期計画のプロジェクトにも参加するようになりました。米国に建設が始まった SSC 加速器の SDC 実験です。ミューオンのレベル 1 トリガーを開発するグループに所属し, トリガー回路の設計にかかります。そこで使うことになったのが VME バス[15]のシステムです。

VMEは元々計算機本体を複数の基板で構成するために用意されたものでしたが, プロセッサの小型化が進み, シングルボードコンピュータ(SBS)が VME のマスターになれるようになって，その役割が大きく変わりました。SBC やホストインターフェースがマスターになり, VME スレーブを検出器インターフェースなどに使用することで, CAMACに比べ 1 桁速いデータ転送が可能な読み出しが出来るようになりました。スレーブの設計は意外とシンプルで, 限られた数の論理素子で実現できます。これ以降, 私の現役の間はVMEが実験用インターフェースの主流でした。

12 Xilinx FPGA との出会い

トリガーロジックのデザインに取りかかった頃, 友人から非常に面白いデバイスがあると教えてもらいました。当時 HERA 加速器の ZEUS 実験にいた徳宿さんです。そのデバイスが Xilinx の FPGA でした。私たちは XC3000 シリーズ[16]を採用しました。それまで SN74 シリーズを使って印刷配線板を作り, プログラミング可能な素子としては GAL などの PLD 程度でした。回路の機能は基本的に印刷配線板のパターンによって実現していた。それが, FPGAになるとチップ内の配線や論理演算を, 内蔵されたコンフィギュレーションメモリーのパターンで設定できる。その中身を書き換えるだけでいくらでも設計変更が出来る。FPGAに供給される入力信号線と演算結果を取り出す出力信号線さえちゃんと基板上で接続されていれば, それをどう処理するかは, 極端な話，実験が始まってしまった後でも書き換えることが出来る。回路開発の革命的な変化でした。SSC 加速器 SDC 実験のミューオン検出器のプロトタイプトリガーモジュールを製作しました。PT1 と名付けました。SSC は中止となりましたが FPGA によるモジュール開発はその後も続き, PT7 まで行きました。

13 UNIX ワークステーションとの出会い

90 年代に入ってから UNIX ワークステーションが普及します。DECstation や SUN Workstation を使っていました。32 ビットのプロセッサを搭載し, 多様なインターフェースをそなえ, ネットワークのホストとしても強力です。オンライン計算機として KEK-PS 実験ではデータ収集の主役にも使われていました。UNIX[17]という OS はオープンソフトウエアとして発展してきた経緯から, 利用者が OS の振る舞いを理解するのが易しい。OS が元々C 言語で記述されていたこともあり, C言語によるソフトウエア開発が気持ちよく出来ました。いわゆるリアルタイムOSではないにもかかわらず, タイミングクリティカルなDAQソフトにもソフトウエアのチューニングにより十分働いてくれました。同じ実験グループの若手にもっと使って欲しいと考え,「高エネルギー屋のためのユニックス入門」[18]を書きました。

14 C++言語との出会い

オブジェクト指向言語であるC++[19]もワークステーション上で使えるようになりました。クラスの導入，実装の隠蔽, 仮想継承, 抽象クラスなどの機能により, より対象物に即したソフトウエアデザインが可能になりました。それまで, アセンブラー, BASIC, FORTRAN, C などの言語を使ってきましたが, コードのサイズが大きくなるにつれ, 大量の関数や広域変数が現れ, それらの管理にエネルギーを費やしていました。オブジェクト指向技術の導入で, そういった苦労から一気に解放された時は感動しました。オブジェクト指向解析・設計という開発概念に基づき, 問題領域を決め, そこに登場する対象物の振る舞いをクラスとして定義していく。クラスの継承から, より抽象化されたものの存在が浮かび上がってくる。クラスのデザインは非常に楽しい物作りプロセスでした。大規模ソフトウエアには必須の技術で, 今でもそのありがたみを実感しています。

15 Intel PC・Linux との出会い

PC98 は日本独自のシステムでした。それ故プロセッサの発展に追随することは出来ず消えていきました。それに対し, IBM-PC およびその互換機[20]は Intel チップの発展をすべて吸収して行きます。プロセッサが 32 ビットから 64 ビットに拡張し, OS も Windows95 以降それまでのワークステーションと遜色ない使い勝手と性能を実現していきます。ただ，Windows はオープンソフトウエアではなかったため, DAQ のホストに使おうなどとはとても思えませんでした。そこに Linux[21]が登場します。基本的に UNIX と互換であり, オープンソフトウエアです。あわせて gcc など開発環境がそろっており, ワークステーションに取って代わるに十分なものでした。OS のパッケージとして無料のディストリビューションも用意されていました。国内では理化学研究所がレポジトリーサービス[22]を提供してくれています。

16 Grid Computing との出会い

私は 2001 年に東大に移り，LHC 加速器 ATLAS 実験の日本のコンピューティングを担当するようになります。ご存じのようにLHCのコライダー実験は膨大な量のデータを生み出します。それらを処理するための計算資源は 1 つのサイトでまかなえるものではありません。そのため, 世界中のサイトを接続した世界分散解析網の構想が提案されます。その基盤技術となるのがコンピューティンググリッドでした。グリッドミドルウエアと呼ばれる一連のソフトウエア群が, 各国でそれぞれのポリシーで運用される計算機ファームの差異を吸収し, 一つの巨大な計算機システムのように見せることが出来ます。Worldwide LHC Computing Grid (WLCG)[23]は今日では 42 カ国 170 研究機関の計算資源が接続され, 100

万 CPU コア, 2EB (エクサバイト) の記憶装置を有します。LHC 実験だけではなく，Belle II をはじめ多くのプロジェクトに利用されており, 大規模実験物理の計算インフラとして不可欠のものになっています。グリッド構築の考え方として, 膨大な計算資源を必要とするプロジェクトを実現するためというのもありますが, もう一つの側面として, 世界のどこにいても平等に実験データにアクセス出来る，居場所によって差別されないと言うことが最初から意識されていたと思います。国際共同研究のあるべき姿を体現したのが WLCG でした。

17 Cloud Computing・仮想化技術との出会い

クラウドコンピューティング[24]は計算資源をネットワーク上に配置し，ユーザはクラウドサービスを通して CPU やストレージを使用します。この時, 従来のコンピュータクラスターではそれぞれの計算ノードは特定の OS がインストールされていて, それにあったプログラムしか走ることが出来ませんでした。それに対しクラウドでは仮想化技術[25]が導入され, 計算ノードでは OS まで含めた実行イメージを持ってきて走らせることが出来るようになりました。ユーザから見ると自分の開発環境と同じものをクラウドで走らせることが出来るので安心してジョブを走らせられます。管理者側から見ると, ある計算ノードで障害が発生したらそのノードを切り離し, 実行イメージを別のノードにコピーして再計算させることで, 稼働率を高く維持することが可能になりました。CERN が提供する OpenStack クラウド[26]は CERN ユーザが自由に利用でき, 私もいろいろなテストに使わせてもらいました。

仮想化技術はクラウドだけでなく，我々の日常業務にも大きなメリットを提供します。ウィンドウズパソコン上に VMWare で Linux を走らせるとか, MAC で Windows の CAD アプリを走らせるとか。私たちの開発環境を大きく変えてくれた技術で，今も重宝しています。

18 SoC デバイスとの出会い

幸い，ATLAS と CMS はヒッグス粒子を発見し，順調にルミノシティを稼いでいきます。その中で実験のアップグレードの話が始まりました。その頃, 私自身は定年が近づいてきて, 最後にもう一つだけ, 楽しめるデバイスはないかと考えていたところ, 出会いました。Xilinx の Zynq7000[27]という SoC(System on Chip)デバイスです。これは Xilinx 社の標準的な FPGA である 7 シリーズ FPGA と，携帯電話などでよく使われるマイクロプロセッサ ARM を同一チップ内に集積したもので, 大規模なハードウエアデザインをロジックエリアに置き, その中のメモリーやレジスタを ARM のバスインターフェースと接続することでプロセッサから読み書きが出来るようになっています。ARM では Linux を走らせることが可能で Ethernet を経由して外部の計算機とのデータ転送を行えます。一旦 Zynq を搭載したハードウエアが完成すれば, FPGA のデザインと ARM のプログラムで自由なシステム構築がいつでも可能です。私が学んできたディジタルエレクトロニクスとオンラインプログラミングが 1 つのチップで実現します。このチップを搭載した VME モジュール PT-Z[28]が私の関わる最後のエレクトロニクスになりました。Z は Zynq の意味と最後のものの意味を掛けています。これを使って定年後も自宅でデザインを楽しんでおります。

19 出会えなかった技術

在職中に出会えなかった技術として一つあげるならば, それは AI でしょう。技術としてはニューラルネットワーク (NN)・機械学習と呼ばれるもので，計算機の進歩により大量のデータ処理と計算が実用的なレベルの結果を生み出せるようになった。実は 2010 年代半ばに我々も調査研究を始めたところで退職を迎えました。NN は我々の業界では早くから取り組んでいた人々がいました。ただ, 実用的かと言われたときに, それを実装する技術的基盤が確立していなかった。2010 年代以降無尽蔵と言えるくらい膨大な計算資源が利用可能になり，大規模な，実用的な規模の NN が実現できるようになった。学習には大量の計算が必要ですが, 一旦学習した NN は一瞬で答えを出してくれる。加速器実験には非常に応用しやすい特徴があります。実験の様々な要素に適用してみたいと思いませんか。私自身としては, 心残りと言えば心残りですが, 現役の人たちにとってみるとこれから面白いことが出来るフィールドが新しく出現したわけで，若い皆さんの活躍を期待するところです。

20 技術の進歩を体感できたこと

ここまで，1970 年代後半から 2010 年代後半までの約 40 年間の私の研究生活の中で出会ったいくつかの技術を紹介してきました。思えば, ディジタルエレクトロニクスやコンピューティングの技術の比較的初期からその発展をずっと追いかけてきたように思います。おかげで, 一つ一つの技術についてブラックボックス (理解できない領域) を作ることなく, 技術の進展を吸収することが出来たと思っています。新しい技術というのは既存の技術の限界に挑戦しそれを乗り越えることで生まれてきました。私たちの世代はその一つ一つのステップを, 時間をかけてたどることが出来ました。今の若い人たちにとってみれば現在の技術は非常に高度で複雑であり, いきなりそれと取り組まなければならないのは大変だと思います。応援しています。

私の思う実験屋の醍醐味は，世の中の技術の進歩を常に眺め, これぞという技術を使って, これまで出来なかった実験を可能にすることです。もちろんそのすべてがうまくいくわけではありませんが, なぜうまくいかなかったかを考える

ことは非常に重要なプロセスです。その経験に基づいてまた次の挑戦がしたくなってくる。何か面白いデバイスはないかな。

付録

定年後の活動として, 若い人たちがエレクトロニクスやオンライン（いわゆる TDAQ）について学習するための助けになればいいなと思ってウェブ教材を作成しています。主に JavaScript を使って書かれており，インタラクティブに半導体技術などを学べます。以下をご覧いただければ幸いです。
https://www.icepp.s.u-tokyo.ac.jp/~sakamoto/education/TDAQ

また本稿の発表にあたり，引用情報を検索しやすくしたバージョンをウェブに掲載しています。上記 URL からたどれます。

参考文献

[1] Transistor–transistor logic, https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic
[2] TRS-80, https://en.wikipedia.org/wiki/TRS-80
[3] Computer Automated Measurement and Control, https://en.wikipedia.org/wiki/Computer_Automated_Measurement_and_Control
[4] PDP11, http://www.pdp11.org/
[5] RT-11, https://en.wikipedia.org/wiki/RT-11
[6] FASTBUS, https://en.wikipedia.org/wiki/FASTBUS
[7] Emitter-coupled logic, https://en.wikipedia.org/wiki/Emitter-coupled_logic
[8] TKO Specification, National Lab. for High Energy Physics, https://inis.iaea.org/records/q7567-hvz69
[9] VAX, https://en.wikipedia.org/wiki/VAX
[10] VAX FORTRAN User Manual, digital equipment corporation, https://vtda.org/docs/computing/DEC/VMS/AA-DO35E-TE_VAX_FORTRAN_User_Manual_Jun1988.pdf
[11] PC-9800 シリーズ, https://ja.wikipedia.org/wiki/PC-9800%E3%82%B7%E3%83%AA%E3%83%BC%E3%82%BA
[12] MS-DOS, https://en.wikipedia.org/wiki/MS-DOS
[13] Transputer, https://en.wikipedia.org/wiki/Transputer
[14] Occam (programming language), https://en.wikipedia.org/wiki/Occam_(programming_language)
[15] VMEbus, https://en.wikipedia.org/wiki/VMEbus
[16] XC3000 Series Technical Information, Peter Alfke and Bernie New, https://docs.amd.com/v/u/en-US/xapp024
[17] Unix, https://en.wikipedia.org/wiki/Unix
[18] 高エネルギー屋のためのユニックス入門, 坂本 宏, https://www.icepp.s.u-tokyo.ac.jp/~sakamoto/education/TDAQ/documents/unix4hep.html
[19] C++, https://en.wikipedia.org/wiki/C%2B%2B
[20] IBM PC compatible, https://en.wikipedia.org/wiki/IBM_PC_compatible
[21] Linux, https://en.wikipedia.org/wiki/Linux
[22] Linux repository https://ftp.riken.jp/Linux/
[23] Worldwide LHC Computing Grid, https://wlcg-public.web.cern.ch/
[24] Cloud computing, https://en.wikipedia.org/wiki/Cloud_computing
[25] Virtualization, https://en.wikipedia.org/wiki/Virtualization
[26] Where are they now? Superuser Awards winner: CERN, Ashlee Ferguson, https://superuser.openinfra.org/articles/cern-openstack-update/
[27] AMD Zynq7000 SoC, AMD, https://www.amd.com/ja/products/adaptive-socs-and-fpgas/soc/zynq-7000.html
[28] PT-Z 関連の解説記事, 坂本 宏, https://www.icepp.s.u-tokyo.ac.jp/~sakamoto/research/atlas/tgcelex/vivado/index.html