SSD Doujinshi 4

SSD の同人誌 4

TAKE FREE ¥0

業界初*！245.76 TBの SSDが新登場！
大容量化・高速化から AIへの応用まで、
より進化する SSD！大船に乗って出航だ！

Contents

SSD ネタ満載の「SSD 同人誌」の PDF 版は
以下のリンクから無料でダウンロードできます。
ぜひご覧ください！

https://www.kioxia.com/ja-jp/insights/ssd-doujinshi-202501.html

SSD 同人誌４号の発刊に寄せて

AI ぱぱ

今年で4回目を迎える SSD 同人誌のテーマは、「はたらく SSD・進化する SSD」。

この数年で IT と AI の世界は劇的に変化し、SSD も大きな進化を遂げました。日常生活でもウェブ検索をすると、AI が、自分が調べたいこと、興味を持っていることへうまく誘導してくれます。趣味のアマチュア無線では交信した証明と記念に、交信記録と自分の好きな図案を入れたQSLカードに交換するのですが、これもAIにイメージを伝えると、思ったようなイラストを自動であっという間に生成してくれます。(まだ納得いくものができてはいませんが。。。)

1993 年に私が初めて買った Windows3.1 PC の HDD 容量は 80MB。現在では、2 TB SSD を搭載した PC が販売されています。約 2 万倍の容量が、最先端のフラッシュメモリで低消費電力、高速動作するドライブとして実現しています。生成 AI システムで使われる GPU も、ここ数年で性能が 1000 倍も向上。世の中のデータを次々と学習し続け、2030 年には、インターネット上のテキストを全て学習できると言われています。その巨大なデータを瞬時に活用するため、さらに高速かつ大容量、そして経済性も兼ね備えた “SSD デバイスと技術 ” が必須となります。

エンタープライズ向けの 245.76TB という大容量 SSD をこの夏リリース。最新インターフェース PCIe® 5.0 で、大容量データを超高速で読み書きできます。そして次世代の省エネデータセンターの設計に変革を起こす基幹技術として、超高速化、省電力化、長距離伝送を可能とする広帯域光 SSD を大阪万博で展示。新しい挑戦の一端を皆さんにご覧いただきました。

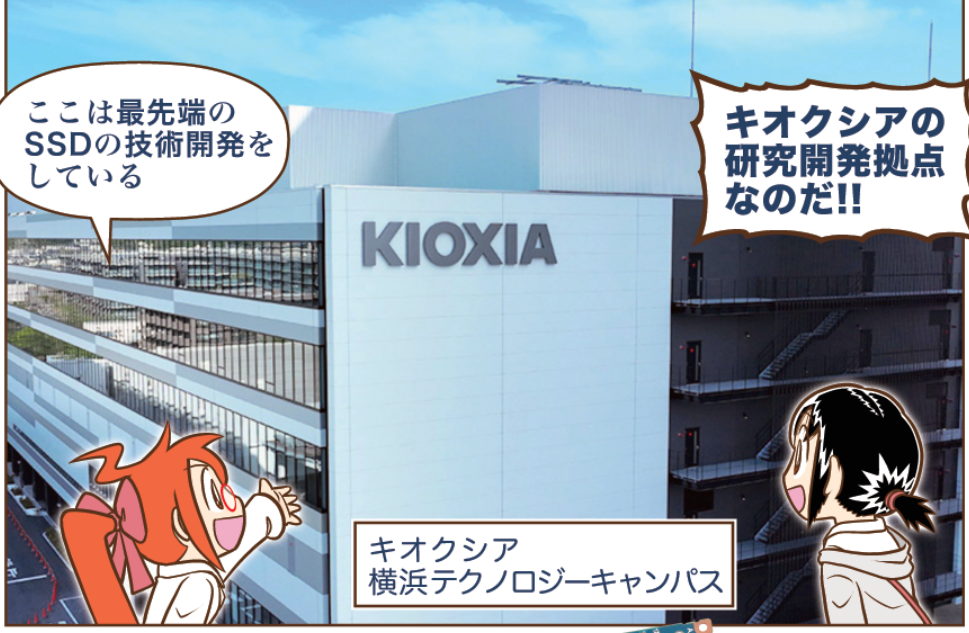
SSD は、データ社会の成長を牽引するキーデバイスとして進化し、今後も巨大なデータを真の価値創造につなげるために、さらに進化し続けなければなりません。

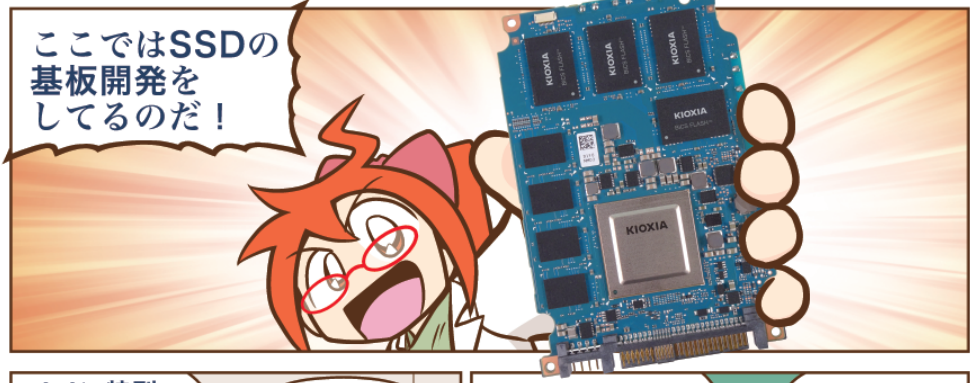
“「記憶」で世界をおもしろくする ” とのミッションを掲げる私たち「キオクシア」のこれからに期待してください。SSD をもっと働かせ、進化させることに全力で取り組みます。大きな可能性を持つ SSD 技術への熱き思いと、それに取り組む私たちの情熱を、この同人誌で少しでも感じていただければうれしいかぎりです。

AI ぱぱ

AI 社会の発展を牽引する SSD 事業を、日本の重要産業へと育てることに情熱を燃やす日々。半導体とストレージ技術を融合するのも、地道な技術者の努力。趣味の無線では、デジタル化も急激に進んでいるが、台風が来るたびに５メートルのアンテナを降ろしたり、上げたり、そしてチューニングしたりと、地道でアナログな作業で休日は忙しい。

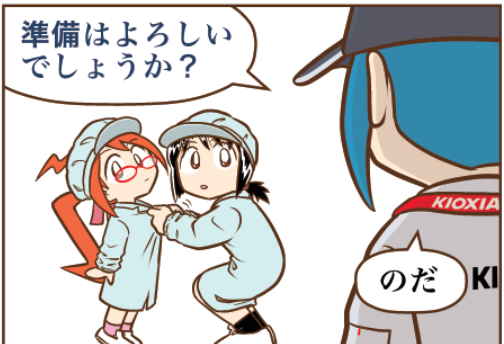

一緒に
SSD開発試作室を
見に行く
のだ!!
データセンター
TENCO
てんこ
SSDの申し子
宍戸ちゃん
最近は生成AI市場の成長で
SSDは大容量になっているのだ
最新SSDはこんな小さい
筐体に245.76テラバイト
なのだ!!
どうやって
はんだ付けしてるか
見たくないか!?
え？
はんだ付け？
これに!?
はんだごてなら
ウチでも
扱ってるけど？
…これで？
ホームセンターTENCO
店員てんこ
そんな巨大な
はんだじゃ
ないのだ！
巨大？
めっちゃミニマムな
スゴ技なのだ！
ここだけの話
またまた
大げさなぁ
ホントなのだ！
宍戸ちゃんを
信じるのだ！
おもしろそー！
大船に乗った
つもりで
GOなのだ!!
大船駅
Ōfuna Stat

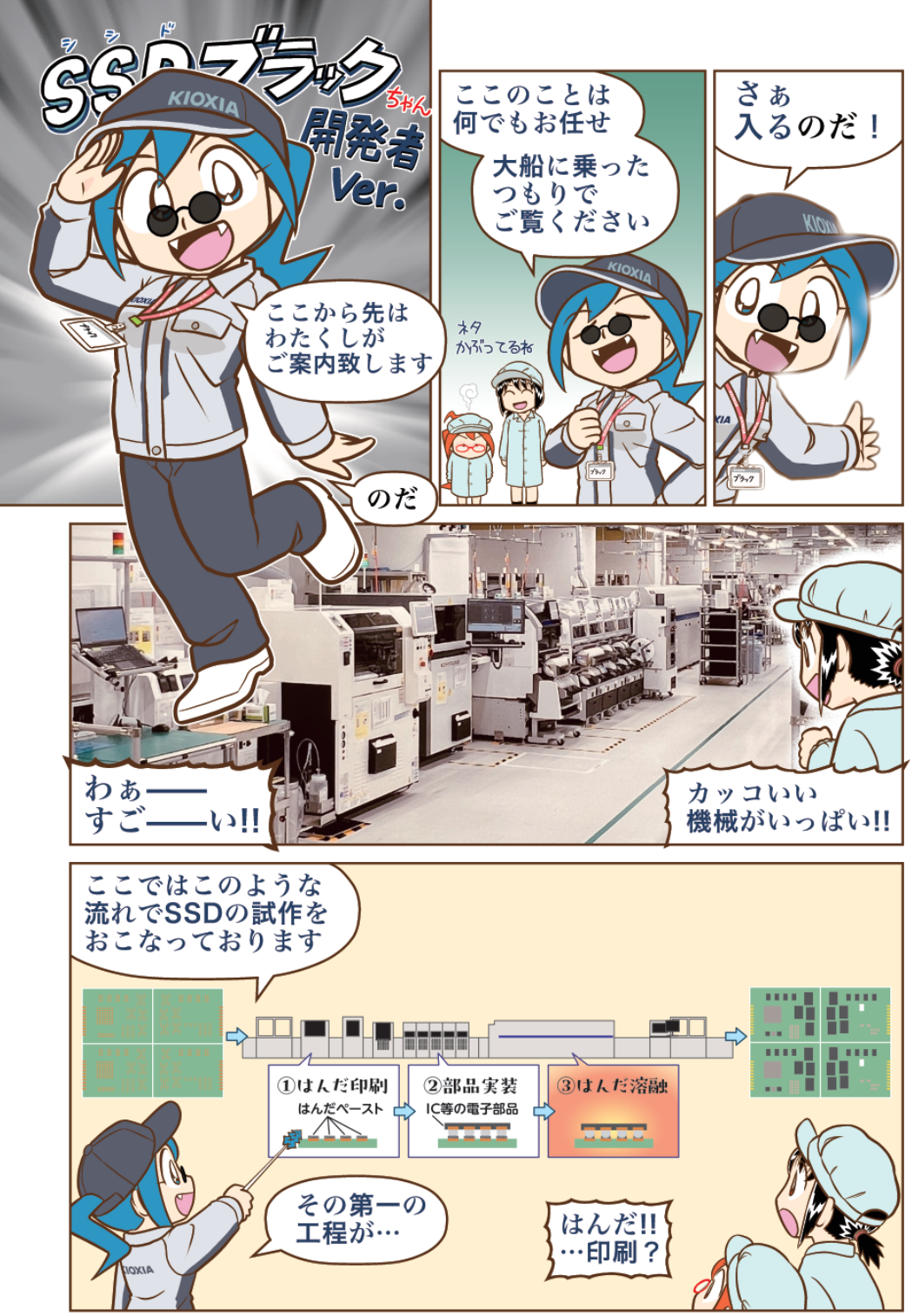

ここは最先端のSSDの技術開発をしている
キオクシアの研究開発拠点なのだ!!
KIOXIA
キオクシア
横浜テクノロジーキャンパス

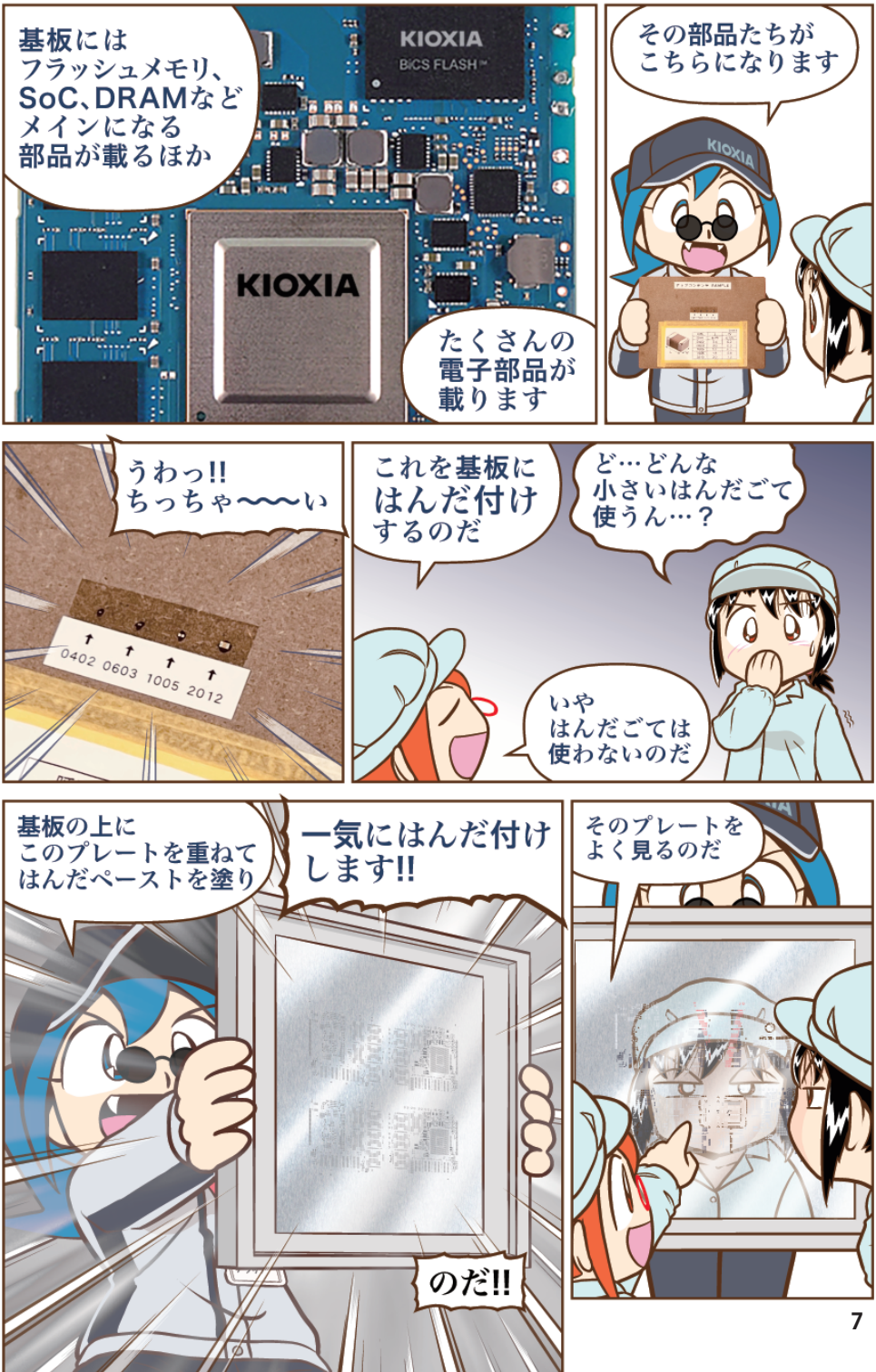

ここではSSDの基板開発をしてるのだ！
KIOXIA


パパに特別の見学の許可をとってもらったのだ
手荷物のX線検査が済んだらSSD開発試作室に入るのだ
わ――ドキドキする――っ


でもこんなとこではんだごてなんて
関係あるのかなぁ？


準備はよろしいでしょうか？
KIOXIA
のだ


SSD(シド)ブラックちゃん
開発者Ver.
KIOXIA
ここから先は
わたくしが
ご案内致します
のだ
ここのことは
何でもお任せ
大船に乗った
つもりで
ご覧ください
ネタ
かぶってるね
さぁ
入るのだ！
わぁ――
すご――い!!
カッコいい
機械がいっぱい!!
ここではこのような
流れでSSDの試作を
おこなっております
①はんだ印刷
はんだペースト
②部品実装
IC等の電子部品
③はんだ溶融
その第一の
工程が…
はんだ!!
…印刷？


基板には
フラッシュメモリ、
SoC、DRAMなど
メインになる
部品が載るほか
KIOXIA
BiCS FLASH™
KIOXIA
たくさんの
電子部品が
載ります
その部品たちが
こちらになります
KIOXIA
うわっ!!
ちっちゃ〜〜〜い
0402 0603 1005 2012
これを基板に
はんだ付け
するのだ
ど…どんな
小さいはんだごて
使うん…？
いや
はんだごては
使わないのだ
基板の上に
このプレートを重ねて
はんだペーストを塗り
一気にはんだ付け
します!!
のだ!!
そのプレートを
よく見るのだ

見えないくらい
ちっちゃい
穴があいてる!?

そうなのだ

プレートの厚さは
約0.1mm

この厚さだと
はんだの粒
4個分にあたる
のだ

はんだの
粒(だんご)

いい

【はけで塗るイメージ】

はんだ

プレート
基板

この装置で
はんだペーストを
プレートに塗る

のだ!

そしてこの装置で
部品を置いて
行く

のだ!

部品って
あの?

このリールを
見てください

KIOXIA

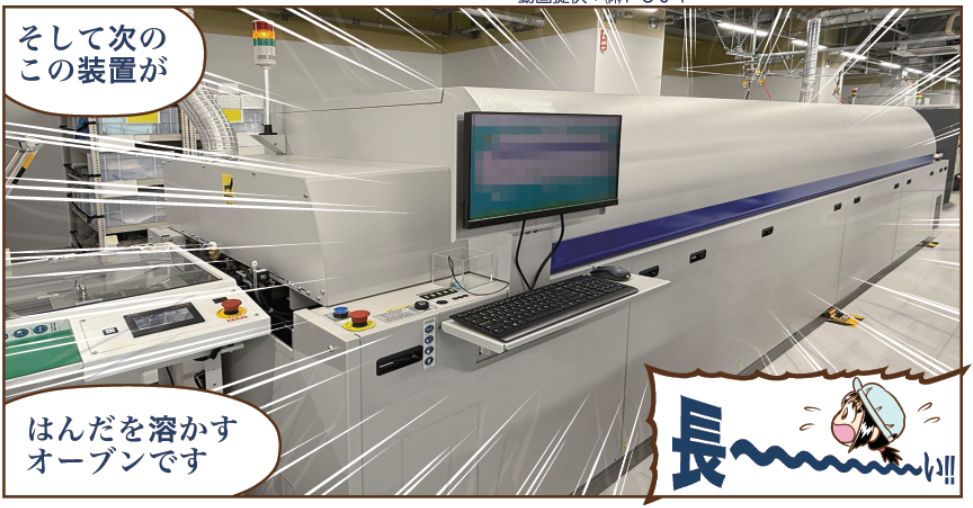
小さい点みたい
なのがいっぱい!

そのひと粒
ひと粒が
部品です


このリールがセットされたマガジンが
マシンガンみたいでカッコいいのだ!!


このマガジンをいくつもセットしたこの装置で
ものすごいスピードで部品を載せていくのだ!
ここで動画が見られるのだ

動画提供：㈱ＦＵＪＩ


そして次のこの装置が
はんだを溶かすオーブンです
長〜〜〜〜〜い!!


一気に熱されて基板が熱で反り返るのを防ぐため
微妙に調整しながら徐々に温度を上げていき
きれいにはんだ付けします


はんだ付けは「富士山のすその」みたいな形が理想だって聞いたことあるけど
SSDの中にはきっと小さい富士山がいっぱい入っとるんやね！
はんだ
はんだ
かわいい♥

じゃあこれが
焼き上がったら
完成ね？

まだなのだ
ここから
検査工程なのだ

そのとき
モノを言うのが
この装置です

そんなすごい
装置なんだ！

いや実際に
「モノを言う」のだ

どゆこと？

もし基板に
異常があったら

「ここがヘンですよ〜
大丈夫ですか〜？」って
指摘してくるのだ

ええっ！
すごい！

そこで技術者が
チェックして
問題がなければ

「これは大丈夫
異常じゃない
ですよ」と伝えると

装置は
「分かりました」
と言って

それ以降は
同じ状態を
見かけても
指摘しなく
なるのだ

勉強したこと
ちゃんと覚えて
いくんだ

すご〜〜い

あたしも
その機能
欲しい〜

※実際に声に出してしゃべることはありません


じゃあ今度こそ
これで完…
まだ
なのだ!!
まだ半分
なのだ!!
まだか!
まだ基板の片面にしか
部品が載ってないため
裏返して反対面にも
部品を載せて
熱していきます
これで同じ
SSDが４枚
出来ます！
AAAA
うら返し
A面パーツ
B面パーツ
片面にパーツが載る
もう片面にもパーツが載る
KIOXIA
両面焼きして
溶け落ちたり
しないの？
職人技
なのだ!!
そのためにも
微調整しながら
温度をあげて
いくのです
それをまた検査して
基板を4分割して
ケースに入れれば
試作品SSDの
完成なのだ…
そして…
KIOXIA
これが
製品化され
解放
量産工場で
作られる
ようになれば
世界征服も夢では
なくなるのだぁ!!
あー
やっぱり
ずっといい子で
いるのは
疲れるのだ


ではさらばなのだ
やっぱりブラックちゃんだったかぁ
すっごく面白かったのだ
うん！精密機器って熱に弱いと思ってたけど
SSD作るときって焼いたりするんやね！
ビックリ
あ――なんか急にピザ食べたくなっちゃった！
部品
はんだペースト
基板
具材
ピザソース
生地
だってSSDがピザに思えてきたんやもん
SSDは両面焼いてたけどね
両面ピザ!?
やってみるのだ!!
いいかも!!
裏のチーズが…
溶け落ちちゃった…

つづく

AI を支えるストレージソリューションの舞台裏

ragnag

キオクシアでは、生成 AI やそれに関連する SSD（ソリッドステートドライブ）についての技術開発に関するニュースが増えていて、SSD 広報担当はいままで以上にさまざまな対応が必要になりました。忙しいけど元気に楽しく毎日を過ごしています。

生成 AI 市場は急速に成長しており、文書の要約や翻訳などの言語処理、さらには画像生成技術を使った自動化ソリューションが、さまざまな業界で効率を向上させる新しい成果をもたらしています。AI の発展には GPU（グラフィック処理装置）が重要ですが、AI が学ぶために必要な情報・データを保管するストレージも同じくらい大切です。AI の学習や推論、例えば大規模言語モデル（LLM）を使った精度向上にはデータが欠かせません。AI の「頭脳」が GPU とするなら、「知識」がなければ良い考えは生まれません。AI の「記憶力」は、データを取得・保存し、必要な時に素早く取り出せる能力を指します。AI とストレージの関係はとても重要なのです。

ひとくちに SSD が必要と言っても、AI のワークロード（作業）には多くのプロセスがあるので、それぞれのプロセスに適したストレージが求められています。

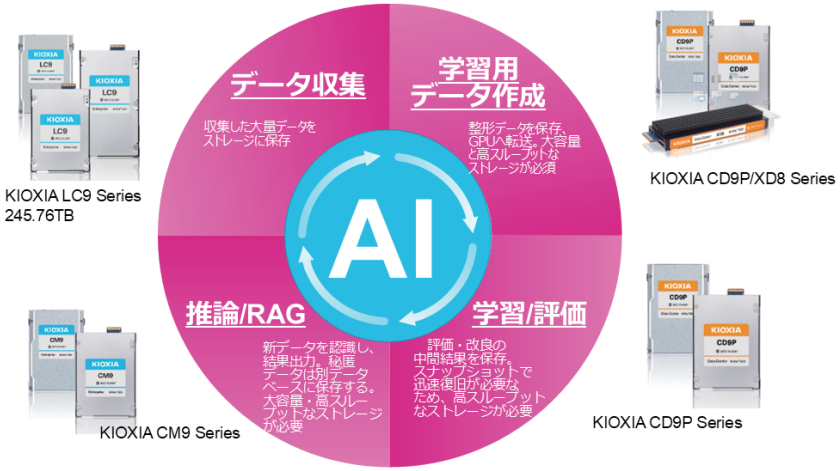


図 1 ：AI のワークロードとキオクシアの最新の SSD の例

気づくかな　　プレスリリースは余裕を持って

ragnag(@ragnag1109)

マンガを読むのと書くのをこよなく愛する広報担当。古本屋さんで子供のころに読んだマンガを見つけて大人買いした。相当な回数読んだのでセリフはもちろんだが、コマワリや欄外のメモやらをほぼ覚えていた（オドロキ）。しかし、主人公から主人公を見守る大人の視点で読んでいる・・・戻れないあの頃。

大船ってどんな所？

～名物・名所紹介～

KIOXIA LC9 シリーズのご紹介

カンミカ

KIOXIA LC9 シリーズってなに？

「KIOXIA LC9 シリーズ」とは、主に生成 AI（人工知能）向けに開発中の大容量の SSD（ソリッドステートドライブ）です（図1）。245.76TB という非常に大きなデータを保存できる特長があります。この大容量を実現するために、CBA[1]（CMOS directly Bonded to Array）技術によりビット密度の向上を実現した業界最大容量[2]となるキオクシアの 3 次元フラッシュメモリ BiCS FLASH™ 第 8 世代の 2Tb QLC を使用しています。2.5-inch[3] と呼ばれる HDD と互換性のある一般的なサイズや、EDSFF[4] と呼ばれるデータセンターや企業向けに設計された新しい形状で高い性能と放熱性を特徴とするフォームファクター[5]の中から、E3.S と E3.L をラインナップしています。PCIe® 5.0 という高速なインターフェースをサポートしています。また、デュアルポート機能[6]に対応しているため、採用システム全体の信頼性を高めることができます。

図 1：KIOXIA LC9 シリーズ

生成 AI 向け SSD に求められること

AI の進化に伴って、処理されるデータ量は増え続けています。例えば、大規模言語モデル（LLM）、膨大なデータセットの学習と保存、ベクターデータベースの利用、推論やファインチューニングを行う際の迅速な情報検索先などがあげられます。有名な生成 AI モデルが訓練に使用するデータセットはおおよそ数十 TB から数百 TB と言われており、今後も増え続けます。

1　CBA：ウエハーボンディング技術を用いて、別々に製造した CMOS 回路のウエハーとメモリセルアレイのウエハーを貼り合わせる技術。

2　2024 年 7 月 3 日現在。キオクシア株式会社調べ。

3　2.5-inch：SSD の出始めのころ、HDD との互換性を重視して、2.5-inch のケースのフォームファクターが作られました。同じスペースで、HDD と同様に SSD を使うことが出来ます。

4　EDSFF：Enterprise and Datacenter Standard Form Factor の略です。エンタープライズおよびデータセンター・ストレージの懸念に対処するためのフォームファクターとして、SNIA SFF Technology Affiliate によって策定されました。高性能で発熱の増加に対処するため、冷却効果も考えられた規格になっています。

5　フォームファクター:SSD の物理的なサイズや形状、接続方式を定義した規格のこと。一般的には、2.5-inch、M.2、PCIe カード型などがあり、それぞれ異なる用途や性能特性に応じた設計がされています。

6　デュアルポート：一つのドライブに対して、玄関口を二つにして、片方がダメになっても、もう片方からアクセスするといった使い道や、複数の Host System からの入り口を分けて、お互いに干渉しないようにするといった使い方があります。この機能の制御は難しいため、SSD のなかでも高機能・高信頼性を求められる製品に搭載されるものです。

QLC SSD と HDD の比較

単位体積当たりの容量 (容量密度) の観点から、HDD よりも SSD が生成 AI 向け用途に適していると考えられます。エンタープライズ向け HDD の最大容量は数十 TB に対して、KIOXIA LC9 シリーズの最大容量は 245.76TB を実現しました。

SLC と TLC と QLC の違いとは？

ここで、KIOXIA LC9 シリーズの大きな特徴の一つである QLC とはなにか、様々な NAND 型フラッシュメモリの種類と比較して理解を深めてみましょう。

半導体メモリにおいてデータを記憶する素子をメモリセルと呼びますが、単純なイメージではコップのようなものだとお考え下さい。

SLC

SLC は、Single Level Cell の略で一つのメモリセル、たとえば、コップに水が入っているかいないかの組み合わせで、2 種類の状態を表現することができます。これにより、表 1 に示すようにそれぞれの状態に「1」か「0」を割り当てることで、1bit のデータ (Single) を記憶することが出来るようになります。実際の半導体メモリでは、水の代わりに電子を使います。

表 1：SLC のイメージ

水の量	bit0
1/1(満杯) 	0
0/1(空) 	1

TLC

TLC は、Triple Level Cell の略で一つのメモリセル、たとえばコップの中の水の量の違いで 8 種類の状態を表現することができます。これにより、表 2 に示すように 8 種類の状態に 3bit のデータ (Triple) の組み合わせを割り当てることで、3bit のデータを記憶することが出来るようになります。

表 2：TLC のイメージ

水の量	bit2	bit1	bit0
7/7(満杯) 	0	0	0
6/7 	0	0	1
5/7 	0	1	0
4/7 	0	1	1
3/7 	1	0	0
2/7 	1	0	1
1/7 	1	1	0
0/7(空) 	1	1	1

QLC

QLC は、Quad Level Cell の略で一つのメモリセル、たとえば、コップの中の水の量の違いで 16 種類の状態を表現することができます。これにより、表 3 に示すように 16 種類の状態に 4bit のデータ (Quad) の組み合わせを割り当てることで、4bit のデータを記憶することが出来るようになります。

表 3 ：QLC のイメージ

水の量	bit3	bit2	bit1	bit0
15/15(満杯)	0	0	0	0
14/15	0	0	0	1
13/15	0	0	1	0
12/15	0	0	1	1
11/15	0	1	0	0
10/15	0	1	0	1
9/15	0	1	1	0
8/15	0	1	1	1
7/15	1	0	0	0
6/15	1	0	0	1
5/15	1	0	1	0
4/15	1	0	1	1
3/15	1	1	0	0
2/15	1	1	0	1
1/15	1	1	1	0
0/15(空)	1	1	1	1

つまり、SLC と TLC と QLC の違いは表現の組み合わせ（つまり、セルが保持できるビット数）にあることが分かりました。表現の組み合わせが増えると、一つのセルでより多くのデータを保存することができます。こうした理由から、QLC SSD は大容量データの保存には適しています。

まとめ

SLC/TLC/QLC を比較することで、ストレージ容量の向上というメリットがあることが分かりました。総じて、QLC SSD は、コストパフォーマンスを重視し、大容量ストレージを手軽に利用したいカスタマーにとって、魅力的な選択肢と言えるでしょう。今後も技術の進展により、さらに多くの人々に利用されることが期待されています。

カンミカ
年数回の海外旅行に向けて日々業務に奮闘中。
直近は GW のスペイン一人旅 2 週間。定番のマドリッド・バルセロナの他にサンティアゴ巡礼（イギリス人の道 112km）を踏破しました！

KIOXIA AiSAQ™ の話

立野 賢登

2025 年 1 月、キオクシアは生成 AI の回答精度を向上させるためのソフトウェア技術である KIOXIA AiSAQ™ を公開した[1]。メディアからの取材や展示会への出展など、ありがたいことにビジネス面での存在感を強めている。プレスリリースでは何やらとても大きな話をしているが、実のところは「SSD を活用して DRAM 使用量をゼロにするベクトル検索技術」である。ベクトル検索単体でアピールするよりも生成 AI の文脈に則った方が耳目を集められるだろうということのようで、実際その通りになった。AI ブームさまさまである。この KIOXIA AiSAQ は何を隠そう筆者のチームが開発したのだが、ここまでこぎつけるまでの紆余曲折をお話ししようと思う。

KIOXIA AiSAQ のしくみ

まず KIOXIA AiSAQ とは?という方もいらっしゃると思うので、簡単に紹介する。ベクトル検索あるいは近似最近傍探索（ANNS: Approximate Nearest Neighbor Search）とは、ベクトルデータベースの中からクエリ（質問）ベクトルに近いものを探し出す手法のことである。ベクトル検索の手法の一つにグラフ探索というものがあり、ノード（ベクトル）同士を結ぶエッジ（矢印）をたどってクエリと距離が近いベクトルを探し出す。この時に必要なデータは従来 DRAM 上に存在しているが、それを SSD に移すことによって、DRAM 使用量をほぼゼロにすることができる。詳しくは筆者が書いた論文[2]にまとめているので、お時間があればぜひ。

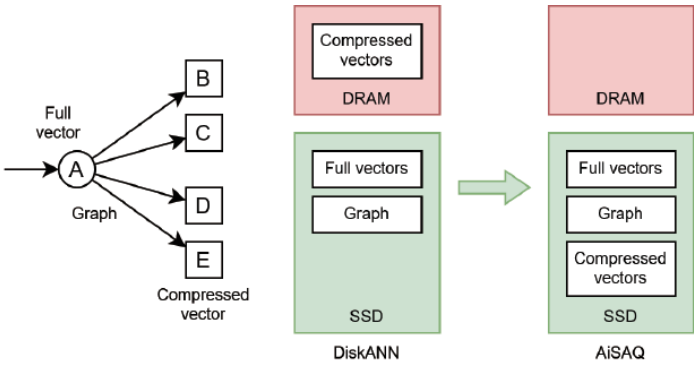


図 1：KIOXIA AiSAQ の仕組み

1 “生成 AI の回答精度向上に貢献する、SSD を活用したソフトウェア技術「KIOXIA AiSAQ™」をオープンソースとして公開,” 2025/1/28. [オンライン].
Available: https://www.kioxia.com/ja-jp/business/news/2025/20250128-1.html. [アクセス日 : 2025/8/7].
KIOXIA AiSAQ™ はオープンソースソフトウェアです。GitHub からダウンロード可能です。
GitHub：https://github.com/kioxia-jp/aisaq-diskann

2 Tatsuno Kento, et al., “AiSAQ: All-in-Storage ANNS with Product Quantization for DRAM-free Information Retrieval,” arXiv preprint arXiv:2404.06004, 2024.

KIOXIA AiSAQ 誕生秘話

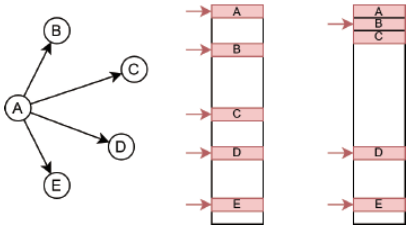


図 2：KIOXIA AiSAQ のベースとなった、ノード先読み手法

KIOXIA AiSAQ は Microsoft の DiskANN[3] をベースとしているが、そこから一歩進めたアイデアは筆者が以前行っていた別の研究[4]に端を発する。その研究は、グラフ探索において現在のノードの隣接ノードを SSD 上の同一ブロックに配置すれば、一度の SSD の読み出しでエッジの先のノードの「先読み」ができ、SSD 読み出し回数を減らして探索速度を向上させることができる、というものである。この先読み手法によって探索時間を最大 10% 削減できたが、その性能を出せる条件が限定的なこともあり、インパクトには少々欠けていた。

この研究がひと段落ついて次のステップを考えていたところ、一緒に仕事をしていた筆者の研究所の別のチームとの打ち合わせで、「DiskANN ではメモリ上に圧縮ベクトルが配置されているが、これを先の研究と同様に同一ブロックに配置することで DRAM 使用量をゼロにできないか?」との提案が上がった。ちなみにそのチームとは部署の統合を経て合流し、現在も一緒に仕事をしている。

この提案手法を実装して実験を行ったところ非常に良好な結果が出たため、この提案手法に名前を付けようという話になった。筆者は当時、研究所の別アイテムである RaLLe[5] という RAG フレームワークの開発を手伝っており、それに倣って覚えやすく親しみやすい名前を付けたいという思いがあった。キオクシアの SSD を使用してもらいたいのでストレージに関する単語を入れたかったこともあり、筆者は提案手法を All-in-Storage ANNS with Product Quantization, 略して AiSAQ（アイザック）と命名した。

ビジネス展開

この KIOXIA AiSAQ の研究成果が揃ったのと同時期に事業部の目に留まって、KIOXIA AiSAQ をビジネスとして展開することとなった。キオクシア社内のみならず、現地法人と協業して機能追加を行ったほか、ベクトル DB（データベース）ベンダなどの外部の企業も興味を持ってくれた。このようにもともと研究方面で始めた案件がビジネスで確かな手ごたえを得ることができたというなんとも面白い結果となった。やはり長所のアピールの仕方によって、刺さる場面が違ってくるのだろうか?

3　Jayaram Subramanya, Suhas, et al., “Diskann: Fast accurate billion-point nearest neighbor search on a single node.,” Advances in Neural Information Processing Systems 32, 2019.

4　立野賢登 , et al., “ストレージを用いた近似最近傍探索におけるデータ配置最適化手法の提案 ,” IEICE Conferences Archives. The Institute of Electronics, Information and Communication Engineers, 2023.

5　Hoshi Yasuto et al., “RaLLe: A framework for developing and evaluating retrieval-augmented large language models,” Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 52-69, 2023.

その間筆者は何をしていたかというと、KIOXIA AiSAQ の応用研究の傍ら、KIOXIA AiSAQ チームの飲み会の幹事をしたり、説明員として展示会に参加したりした。写真は 2025 年 6 月に開催された Interop Tokyo に出展した時のもので、KIOXIA AiSAQ は Best of Show Award のネットワークインフラ（AI インフラ）部門で審査員特別賞を受賞することができた[6]。筆者は初めて展示会の説明員というものを担当したが、足が棒になったもののよい経験となった。2025 年 8 月にサンフランシスコで開催される the Future of Memory and Storage (FMS) 2025 でも説明員として出張する予定となっており、12 年ぶり 2 回目の海外渡航に緊張しつつも楽しみにしている。

図 3：Interop Tokyo 2025 での KIOXIA AiSAQ ブース（上）と審査員特別賞（下）

おわりに

今回筆者は KIOXIA AiSAQ の開発者ということで本誌に寄稿する機会をいただけたが、KIOXIA AiSAQ がここまで大きなプロジェクトとなったのは、一緒に仕事をしている同僚・上司や部署をまたいだコミュニケーションなど、周りの方々によるところが非常に大きいので、この場を借りて感謝を申し上げたい。筆者は入社 5 年目というそろそろ若手から中堅になる年次であるが、お世辞にも物事を俯瞰してみる能力があるとは言えないので、今まで通り目の前の課題で上げた成果を上司や同僚にアピールすることで、活躍の機会を広げていけるようにしたい。

6 “「KIOXIA AiSAQ™」が「Interop Tokyo 2025」の「Best of Show Award」の「ネットワークインフラ（AI インフラ）部門」で審査員特別賞を受賞 ” キオクシア株式会社 , 2025/6/12. [オンライン].
Available: https://www.kioxia.com/ja-jp/business/news/2025/20250612-1.html. [アクセス日 : 2025/8/7].

立野 賢登
SSD 同人誌へは初めての寄稿になります。
暑くて外に出られないので愛馬にはなかなか乗れませんがそのぶん在宅勤務がはかどりますね。

KIOXIA SSD を使用した Raspberry Pi ファイルサーバー構築

いも

「我が家にもお手軽に構築できるファイルサーバーがあったらなぁ…」なんて、一度は思ったことがあるのではないでしょうか？今回は、そんな夢を Raspberry Pi 5 と SSD で叶えてみました！使用するのはもちろん、キオクシアの最新 M.2 SSD モデル EXCERIA PLUS G4 の NVMe™ SSD です。

とはいえ、私はサーバーエンジニアではないので、ChatGPT にすべて聞きながら構築してみました！ ChatGPT すごい。。。

図 1 ：Raspberry Pi 5

環境 Set up

【大まかな流れ】

① Raspberry Pi 5 で SSH 接続設定を行う
② nvme-cli をインストール
③ SSD のマウント設定
④ Samba のインストールと設定
⑤ Windows からアクセス、確認

Raspberry Pi と SSD の組み合わせは図1の感じです。早速立ち上げていきましょう。

Raspberry Pi 5 で SSH 接続設定を行う

ここは正直なくても OK です。自身の環境的に SSH 接続で Raspberry Pi を操作するほうがラクだったというだけで、直接 Raspberry Pi にキーボードさしてコマンド打ってもよいです。詳細は省きます。

nvme-cli をインストール

nvme-cli は、Linux 上で NVMe SSD を操作・管理するためのコマンドラインツールです。GitHub にソースコードが公開されており、誰でも Linux 環境で使用することができます。

インストール

- 必要パッケージのインストール :

```
sudo apt install -y build-essential git meson ninja-build pkg-config libudev-dev
```

- ソースのクローン :

```
git clone https://github.com/linux-nvme/nvme-cli.git`
```

- ビルド :

```
cd nvme-cli
meson setup build
ninja -C build
sudo ninja -C build install
```

ビルドできたら、試しにコマンド打ってみます。

```
- nvme list
```

```
Node                  Generic               SN                   Model
--------------------- --------------------- -------------------- --------------------------------------
/dev/nvme0n1          /dev/ng0n1            3D8KF0E1Z00C         KIOXIA-EXCERIA PLUS G3 SSD

Namespace  Usage                      Format           FW Rev
---------- ------------------------- ---------------- --------
0x1          1.00  TB /   1.00  TB    512   B +  0 B   ELFA01.2
```

現在接続されている NVMe SSD の一覧です。KIOXIA Exceria が接続されていることが分かりますね!

実行確認

```
nvme smart
```

SSD の状態を表示してくれます（図２）。細かい項目は解説しませんが、温度や %used（SSD が EOL を迎えるまでに実行できる Total 書き込み回数のうち、どれくらい書き込んだかの指標）が分かったりします。

```
Smart Log for NVME device:nvme0 namespace-id:ffffffff
critical_warning                    : 0
temperature                         : 25 ° C (298 K)
available_spare                     : 100%
available_spare_threshold           : 5%
percentage_used                     : 0%
endurance group critical warning summary: 0
Data Units Read                     : 116 (59.39 MB)
Data Units Written                  : 39 (19.97 MB)
host_read_commands                  : 2814
host_write_commands                 : 1059
controller_busy_time                : 0
power_cycles                        : 15
power_on_hours                      : 228
unsafe_shutdowns                    : 8
media_errors                        : 0
num_err_log_entries                 : 0
Warning  Temperature_Time           : 0
Critical Composite Temperature Time : 0
Temperature Sensor 1                : 25 ° C (298 K)
Thermal Management T1 Trans Count   : 0
Thermal Management T2 Trans Count   : 0
Thermal Management T1 Total Time    : 0
Thermal Management T2 Total Time    : 0
```

図２：nvme smart の実行結果

外部ストレージのマウント設定

- ディスク確認 : `lsblk`
- ext4 フォーマット : `sudo mkfs.ext4 /dev/nvme0n1p1`
- マウント : `sudo mount /dev/nvme0n1p1 /mnt/nvme1p1`

NVMe デバイス (SSD) の path はご自身のものをお選びください。

Samba のインストールと設定

- `sudo apt install samba`
- 設定ファイル `/etc/samba/smb.conf` に以下のセクションを追加：

```
[Share-Exceria]
   path = /mnt/nvme1p1
   browseable = yes
   writable = yes
   guest ok = yes
   force user = nobody
   create mask = 0777
   directory mask = 0777
```

これにより、Share-Exceria という名前の共有ディレクトリが認識できるようになります。
Samba ユーザー設定は下記からできます。

- `sudo smbpasswd -a nobody` によりパスワードを設定

Windows 側からのアクセス

- `\<Raspberry Pi の IP アドレス >\Share-Exceria` にアクセス
- 認証情報には、⑤で設定したユーザー：`nobody`、パスワードを入力

以上で共有サーバーとして SSD を活用できます！

Windows からはこんな感じで見えています。あとは好き放題ファイルを置くだけです。

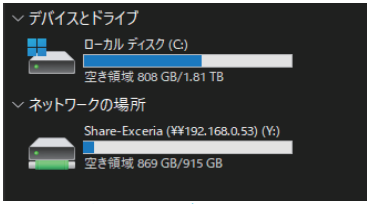


図3：共有サーバーに見えた！

まとめ

Raspberry Pi と SSD を組み合わせて、簡単に共有サーバーを立ち上げることができました！今回の方法はセキュリティガバガバなので、その辺りはこれから詰めたいと思います。SSD がより身近なストレージになればいいなと思う今日この頃です。

いも

ビールと実家のパグが大好き。最近健康診断で尿酸値を怒られはじめている。

フラッシュメモリの読み書きで作る人工知能

にちか

にちかと申します！去年は NAND 型フラッシュメモリの読み書きで 16bit CPU の処理を再現し、その上でゲームプログラムの実行に成功しました。記事の公開後にいっぱい感想を頂き嬉しかったです！去年の反響に味を占めて、今年は NAND 型フラッシュメモリの読み書きで、いま流行りの人工知能（AI）に挑戦します。

どんな人工知能を作るのか

SSD 同人誌 3 号で作った CPU で昨今の生成 AI のようなものを動かせたら良いのですが、その動作周波数はたった 1.9Hz しかなく、結果は望むべくもありません。そのため、CPU の性能アップは図るにしても、目標達成のハードルも限りなく下げたいと思います。ではどうすれば、なるべく簡単に AI が作れるでしょうか？

当記事では、”実際に AI と呼ばれたことのある技術 ” を、新しさは気にせず AI として扱うことにします。1991 年に刊行された AI 白書[1]を読むと、「探索」「演繹」「機械学習」といった現代技術と並んで、「ファジィ理論」が AI の基礎分野の一つとして挙げられていました。ファジィ理論とは、「はい」「いいえ」のような二値ではなく、人間が下すような曖昧な判断を数値として扱い、推論や制御を行う理論です。確かに家電業界にはかつてファジィブームがあり、ファジィ制御搭載を謳った洗濯機や掃除機が販売されていた時期があります[2]。つまり、ファジィ制御は少なくとも 1991 年当時はれっきとした人工知能の一種と考えられており、さらに当時のマイコンでも実行可能なほどに処理が軽量のはずです。以上から、当記事は NAND 型フラッシュメモリによるファジィ制御を目標とします。

制御シミュレーション

ファジィ制御をやるにあたって、何をどう制御するかを考えます。ファジィ制御は家電や鉄道、発電所など様々な応用例がありますが、私の低速な CPU でも扱えるシステムが理想です。例えば、計算が遅すぎると事故が起きてしまう乗り物や発電機は向いていません。そのため家電の事例を調査した結果、エアコンに着目しました。空気など流体の温度はゆっくり上下するため、私の CPU でも計算が間に合いそうな気がします。今回はモデル化のしやすさを優先し、空気の代わりに水を、エアコンの代わりに電熱ヒーターで温めるシミュレーションを行います。

1 「AI 白書 人工知能の技術と利用」財団法人日本情報処理開発協会 (1991)

2 比屋根一雄.「そういえば、ファジーなんて言葉があったよね」三菱総研 https://easy.mri.co.jp/19990427.html (参照 2025-08-18)

Raspberry Pi 5 上で C++ 言語を使い、シミュレーション環境を構成しました。その概要図を図 1 に示します。容器の中にある水 1kg を、底面から 1kW の電熱ヒーターで温めます。更に容器の中には温度センサが、電熱ヒーターにはリレーが取り付けてあり、水温の計測と制御が可能です。なおこのシミュレーションは空気中への放熱も模擬しているため、温度を維持するためにも一定の加熱が必要です。図 1 中央の制御器は水温を見て、ヒーターを On するかどうかを判断します。この判断に相当する計算を、フラッシュメモリ製 CPU が処理します。ただし、リレーは高速なスイッチングが難しい為、30 秒間ずっと On するか Off するのみの制御をします。制御則は PID 制御とファジィ制御の 2 種類で行い、両者の結果を比較します。

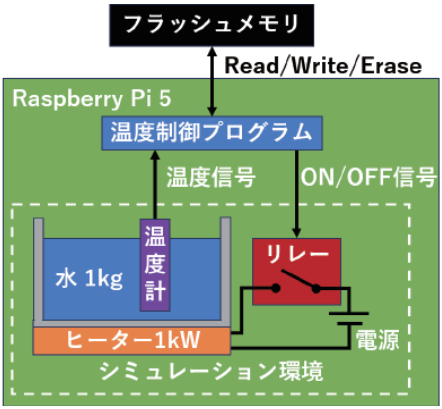


図 1：シミュレーション環境

フラッシュメモリ製 CPU のハードウェア

去年は CPU 開発に JISC-SSD(Jisaku in-Storage Computation SSD 学習ボード) を用いましたが、開発中にフラッシュメモリが劣化してしまうのが忍びなく、課題意識がありました。そんなある日、SNS で nandsim の存在を知りました。nandsim はその名の通り、NAND 型フラッシュメモリの挙動を模擬する、標準の Linux カーネルモジュールです。NAND 型フラッシュメモリで計算機を作りたい人々にとって、なんと便利なツールでしょうか！！（そんな人が私以外にいらっしゃるかはさておき。）

nandsim を有効化すると、/dev/mtd0 というファイルディスクリプタが現れます。これはどうやら、最初に OS から認識された MTD(Memory Technology Device) という意味のようです。MTD は SPI フラッシュやスマートメディアカードなど、各種メモリ製品にドライバを紐づけて抽象化してくれる有難い機能です。MTD は FTL やウェアレベリングなどはサポートせず不便ですが、裏を返せば、それらを介さずに OS から直接 Raw データを読み書きすることができます。つまり、特定のアドレスにデータを 2 回書き込めば、書いたデータ同士の AND も計算できます。フラッシュメモリで計算機を作りたい人々にとって、なんと都合の良いインターフェースでしょうか！！（2 回目）

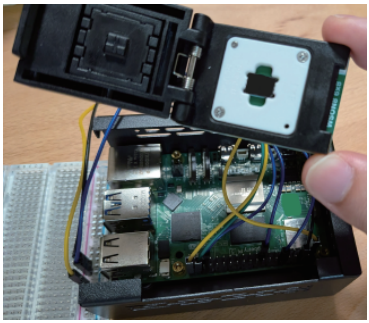


図 2：シミュレーション環境外観

今回用意した実験環境の外観を図 2 に示します。MTD の実機には、キオクシアの TC58CVG1S3HRAIJ を用いました。キオクシアのメモリ事業部の方に相談した際、「生産終了品でも良ければ」と貸してくださいました。3.3V の信号で動作するため、Raspberry Pi に直結可能です。ただし 6 × 8mm の小さなチップなので、はんだ付けは難しいと考え、治具を購入して取り付けました。

フラッシュメモリ製 CPU のソフトウェア

SSD 同人誌 3 号では、フラッシュメモリで 16bit 加算器を設計しました。しかし制御器を実現するには加算だけでなく、乗算やループ、条件分岐などの機能も必要です。これらの機能をフラッシュメモリで実現するにはどうすれば良いでしょうか？

折しも 1 月に電子情報通信学会の研究会へ参加した際、OISC（One-Instruction Set Computer）を知りました。OISC はその名の通り命令が 1 種類しかないのですが、驚くべきことにチューリング完全な CPU です。つまり、たった 1 種類の命令の繰り返しで乗算やループ、条件分岐も実行可能です。フラッシュメモリで計算機を作りたい人々にとって、なんと魅力的な設計でしょうか！！（3 回目）

さらに調べていくと、GitHub 上で SUBASIC[3] というプロジェクトを発見しました。SUBASIC は、Subleq という OISC 上で動作する小さな BASIC インタープリタです。Subleq について平たく説明すると、整数同士の引き算を行い、結果が 0 以下の時だけ別のアドレスへジャンプする、という命令のみを備えた CPU です。SUBASIC は標準入出力、24bit 整数の四則演算、条件分岐やループに対応しているので、これで制御器が実装できそうです。

以上の説明を整理したものが図 3 です。NAND 型フラッシュメモリを MTD として認識させ、その読み書きで NAND 演算を行います。またその演算の組合せで Subleq の処理を再現します。さらにその Subleq 上で BASIC を動かします。さいごにファジィ制御器を BASIC で実装し、人工知能を実現します。まるで " 風が吹けば桶屋が儲かる " みたいなシステムですね。

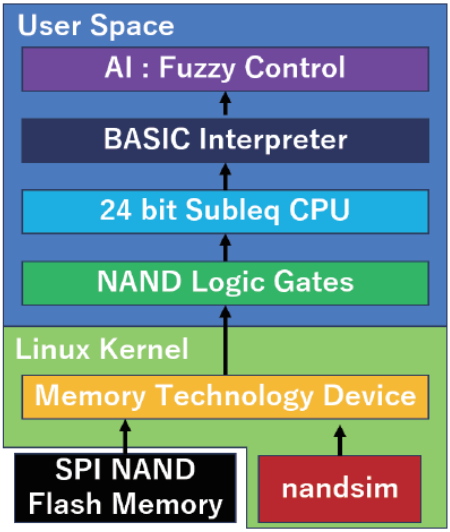


図 3 ：フラッシュメモリで AI を作る過程

3 https://github.com/mengstr/SUBASIC Copyright 2024 Mats Engström

フラッシュメモリ製 CPU の高速化

フラッシュメモリ製 CPU の性能を上げる為、今年は以下の改善を行いました。

1. ページバッファ

CPU 内に 1 ページ (2048Byte) 分のバッファを用意し、同時に読み書き可能なビットはまとめて操作する事で、読み書き操作にかかるレイテンシを削減しました。去年は 16bit 整数の足し算に 500 回以上の書き込み操作が必要でしたが、この工夫により、24bit の引き算が 37 回の書き込みで完了するようになりました！

2.ROM の領域を拡張

計算に使ったメモリセルは、消去しないと再利用できません。そのため前回は足し算する毎に都度ブロック消去を行っていました。しかし、消去操作は実行に数 ms かかる上に、頻繁に行うとフラッシュメモリの寿命に影響を及ぼす可能性があります。そこで ROM の領域を可能な限り増やし、121 回の計算につき 1 回だけ消去するように変更しました。こうすることで、消去操作にかかる時間を 1/121 に削減し、更にフラッシュメモリの長寿命化も期待できます。

3.ALU の設計変更

前回は単純なリップルキャリー加算器を採用しましたが、加算器が直列接続のため、1bit ずつの繰り上げによる処理時間の遅延が課題でした。そこで今年は、階層型キャリー先見回路（図 4）に変更し、書き込み操作を 37 回から 18 回までさらに削減しました。そのお陰で処理速度も 68.8% 向上しました。

上記 3 つの工夫により、CPU の動作周波数は、24.1Hz(12.7 倍) になりました！

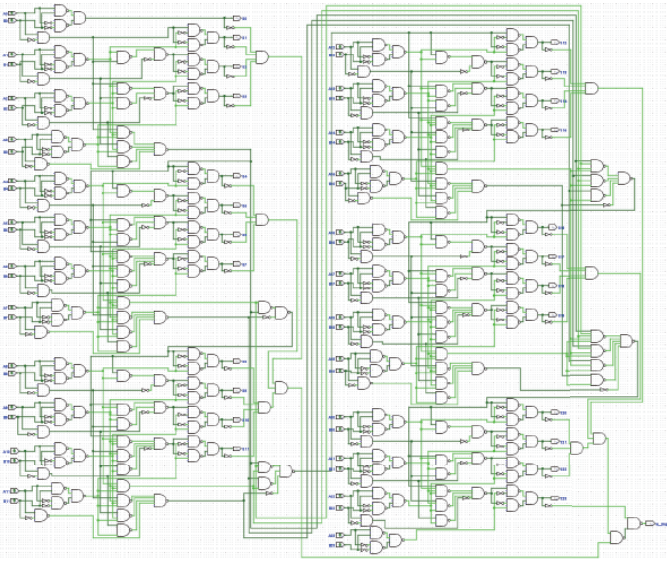


図 4：設計に 3 か月かかった階層型キャリー先見回路

結果

BASIC ベンチマーク

BASIC コンピュータの所有者は、マンデルブロ集合を描画させるのが定番のようでして、皆さん自慢のマシンで動作速度を報告しています[4]。12.7 倍も高速化した私のマシンなら、一体どんな結果が出るでしょうか！？

一晩走らせた結果、画面にたった 5 個の文字が並んでいました。実行完了まで半年程かかる見込みです。

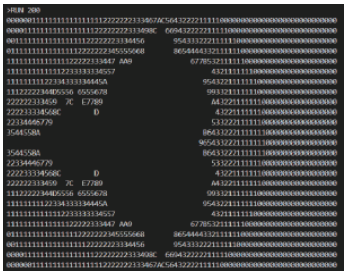


図 5：マンデルブロ集合（完成予想図）

ファジィ制御の実験結果

高速な nandsim を使って計算した制御シミュレーションの結果を図 6 および図 7 に示します。PID 制御を用いた場合は、温度の誤差の L1 ノルム (絶対値の総和) が 276.6 だったのに対し、ファジィ制御は 260.3 となり、より誤差が少なく良好な結果が得られました。

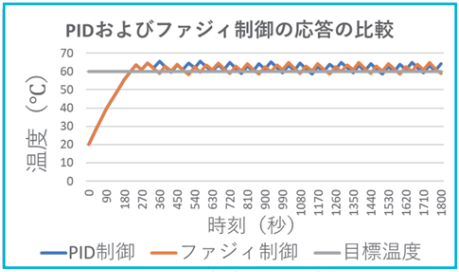


図 6：温度制御の様子

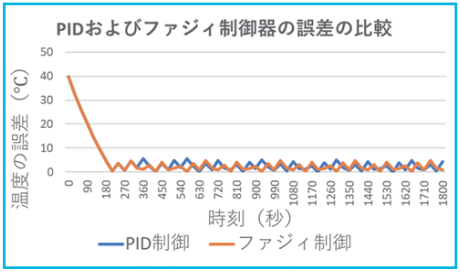
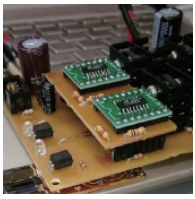


図 7：制御した温度の誤差

おわりに

当記事は NAND 型フラッシュメモリで動作する人工知能を作りました。具体的には、フラッシュメモリの読み書きで 24bit の減算器を構成し、その上で BASIC コンピューターを作りました。更に BASIC 上でファジィ制御器を実装し、シミュレーションでお湯を沸かすことが出来ました。ここまでお読み頂きありがとうございました。また当記事の執筆にあたって、偉大な先人達の作品・知見をいくつもお借りしました。この場を借りて御礼申し上げます。

4 「番外編：ASCIIART(マンデルブロ集合) ベンチマーク」Retro PC Gallery (by はせりん)
http://haserin09.la.coocan.jp/asciiart.html (参照 2025-08-20)



にちか（@lxacas）

大学院生の頃、「PID 制御も実質 AI だよね」という冗談がありました。もちろん AI と呼ぶのはおこがましいほど古典的な手法ですが、人工知能として自動制御を選んだのは、そんな記憶ががきっかけでした。

光 SSD

伊藤 晋朗

我が家には K 君という中二の子供がいる。本人は「俺は中二病だから。」と言ってはばからない。確かにそういった側面はあるのかもしれないが、「マイルド中二病」ぐらいの微妙なレベルな気がしている。中学生の最初の頃は「ネタ」として言っていた気配もあるが、本当の中二になってしまうとあまり言わなくなってきた。もしかしたら来年あたりは恥ずかしくなって言わなくなるのかもしれない。

最近、会社で「光 SSD って何?」という質問をされる。簡単に説明すると、将来的に SSD のインターフェースを光ファイバを使ったインターフェースにするという研究である。ただし、光 SSD が通信するプロトコルは、これまでの SSD と同じように PCIe® のプロトコルになる。そのため、SSD としての役割としては大きく変わる事はないのかもしれない。けれども、最近は PCIe の規格団体も第 7 世代で光伝送の規格を検討している。なので、今から 2、3 世代後にはデータセンターの世界では普通に使われている技術になるといいなあとは思うかな。まあ、先のことはわからないけどね。PCIe の世界には PCIe デバイスを切り替えて使う PCIe スイッチというものがあったり、CXL® という PCIe に DRAM を接続できるという規格が存在する。そのため、今後、コンピューターの内部 BUS であった PCIe 規格が、色々な機器を接続するインターフェースになっていく可能性はあるのだろう。そういった場合に、PCIe の規格も色々なものをつなぐネットワークのインターフェースになっていくのかもしれない。また、デスクトップ PC では SSD は、マザーボードに直付けになっていることが多いけど、サーバーの中では、ストレージは PCIe や SAS の電気ケーブルの先につながっている場合がほとんどだと思う。この電気配線ではスピードが上がると、どんどん距離が届かなくなるので、将来的に高速化すると、ノイズによって届かなくなることも考えられる。その場合には光伝送できると解決策になるのかもしれないね。

昨年、アメリカのサンノゼにあるコンピューター歴史博物館に行ってみた。中の展示は、機械式の計算機などの古いものや、弾道ミサイルの軌道計算する歴史の遺物があったり、みんなが知っている日本のゲーム機や、日本のメーカーが元気だったころの PC 製品の展示物がいっぱいだった。古い機械はすごさが正直わからないけれど、昔遊んだゲーム機を見るのはとても懐かしい。

その博物館の中に、当時のデータセンターのデモ室があった。そこには色々な大きな機械装置がある。そんな機械装置は、裏を見ると職人が手はんだで配線をつなげた回路基板のようなものがあるような機械だった。断線やはんだ不良とかあったら、絶対にわからないだろうから、故障で動かなくなった時の苦労を想像すると冷や汗しか出ない。そして、デモの部屋なので当時の部屋を再現しているだけなのだが、そのデモの部屋には「データ」がいっぱいだった。演算するような機器がどこにあるのかわからなかったが、目につ

くものは磁気テープ、パンチカードの束、そして読み取り機であり、結局部屋の大部分がデータ保存や読み取りのために使われていた。そこでやっと、データセンターが「Data Center」と名付けられたのが理解ができた気がする。今のデータセンターはその頃とは変わって、計算する装置であるサーバーがたくさん入ったラックが並んでいるのが目につく。そのため、「データ」がどこにあるというのは良くわからない。もしかしたら、どちらかというと日本の大学でよく目にしてきた「計算機センター」という言葉の方があっているのかもしれない。

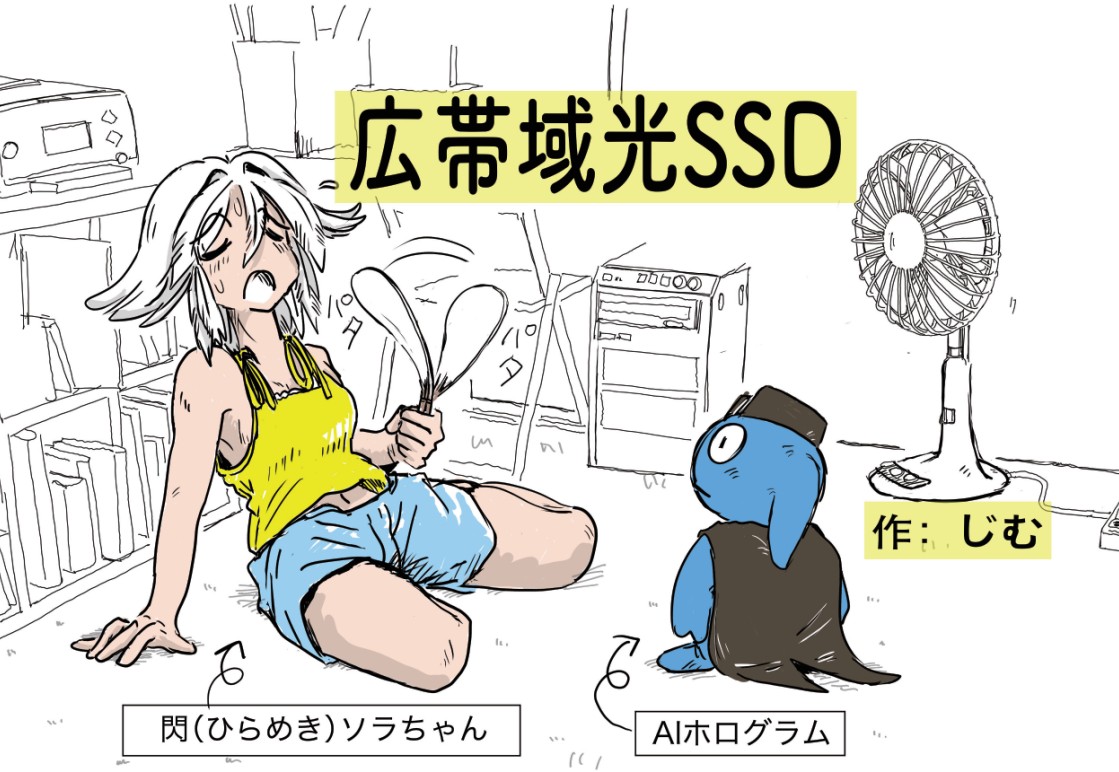
昔と今では、データセンターの中身は結構変わったのかもしれない。そして、サーバーの中身も変わってきていると思う。CPU の面積は、2000 年ぐらいと比べると、今では 4 倍くらいになっている気がするし、メモリのチャネル数も 6ch とか 8ch に増えている。そのため、サーバーの中は CPU やメモリが支配しており、これ以上部品がでかくなると配置できないよねと思ってしまう。そして、GPU サーバーになると、電力と熱が問題なので置き場所も考える必要があったりもする。GPU サーバーのために建物から考えて設計できればいいんだろうけど、今までの部屋に置こうと思ったら、冷却能力とか電力とか足りなくてスカスカの部屋に、GPU サーバーだけおいてあるんじゃないかと思ってしまうので、データセンターの建て方も新しくなっていくのかもしれない。

K 君は、最近新しい京都の会社のゲーム機を手に入れた。入手が難しいと言われたので、市場調査として、家電量販店に一緒に確認しに行ったら、なぜかあってよくわからないまま購入してしまった。K 君は喜んだが、お金をいっぺんにたくさん使うのはよくないと考えたのか、新しいゲームは買ってない。結局新しいゲーム機に手を出したいだけだったのかもしれない。でも、とりあえずこれからもなんでもよいので新しいことには手を出してもらいたいと思うし。K 君もそのうち大人になって、失敗だったと思うこともあるのだろう。どんな事をはじめるのか知らないが、ぜひ俺の知らないことを初めてほしいもんだ。

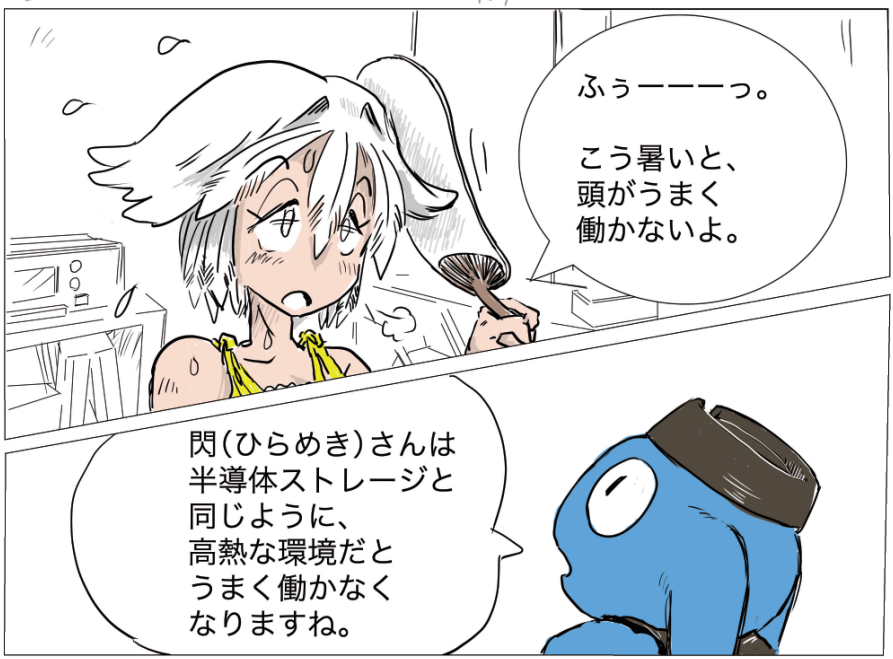
伊藤晋朗 (@ikainuk)

たまに「コーディング」。たまに「予算管理」。たまに家でチェーンソー。そんな生活。

広帯域光SSD


パタ
パタ
閃(ひらめき)ソラちゃん
AIホログラム

作：じむ


ふぅーーーっ。
こう暑いと、
頭がうまく
働かないよ。
閃(ひらめき)さんは
半導体ストレージと
同じように、
高熱な環境だと
うまく働かなく
なりますね。

閃(ひらめき)さん。
ここは、エアコンを
買った方がいいですよ。

データセンター
だって、
クーラーが
ガンガン効いて
暑さ対策をして
います。

データセンターと
我が家を一緒にしないでよ。

熱対策は半導体
ストレージの
要ですからね。

なんせ、
大阪万博会場でも
展示してましたし。

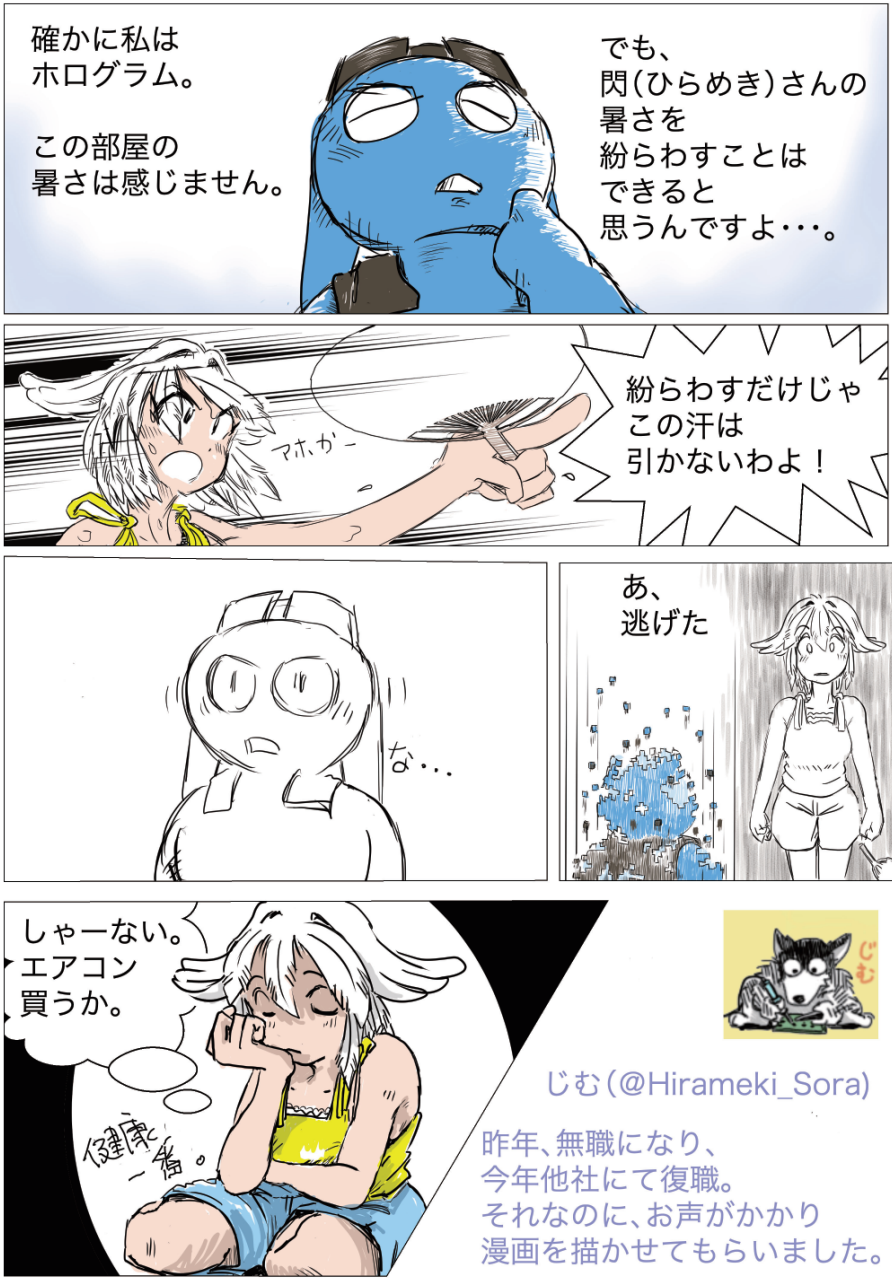
大阪万博？！

ハイ。
「フューチャーライフ
エクスペリエンス館」と
いう場所での
展示でした。

その名も
「キオクシア
広帯域光SSD」。

まだ試作段階ですけれど、従来の電気信号では無くて、
”光信号”を用いることで「省電力化」を図って
熱対策をしているんです。


コレ、
熱対策だけじゃなくて
光通信で、「ノイズ対策」や
「データの高速化」を
したり、
光ファイバーなので
データセンターの
配線の取り回しが
良くなったり。
……？
今後、
大量のデータを
処理するには、
”光SSD”が
必要になって
きますよ。
何言ってんの
ちょっと、
それと
私の部屋の
熱対策とは
関係ないじゃん。
ったく
AIホログラムの
くせに
役に立たない
わねー。
エヘン



じむ(@Hirameki_Sora)

昨年、無職になり、
今年他社にて復職。
それなのに、お声がかかり
漫画を描かせてもらいました。

コンシューマー向け SSD ラインナップとおすすめモデル

Hatto

キオクシアではエントリーからハイエンドユーザーまで幅広い方々に選んでいただけるよう、以下表のようにラインナップをそろえています。この記事では新製品である EXCERIA PLUS G4 NVMe™ SSD と EXCERIA PLUS G2 ポータブル SSD についてご紹介します。

用途	おすすめモデル
一般的な PC 用途／普段使い	EXCERIA G3 の NVMe SSD
ゲーム・動画編集	EXCERIA PLUS G4 の NVMe SSD
プロ向け・高負荷作業	EXCERIA PRO の NVMe SSD
PS5・Heatsink 付	EXCERIA with Heatsink の NVMe SSD
外出先でのデータ管理、外付け、スマホ	EXCERIA PLUS G2 ポータブル SSD

EXCERIA PLUS G4 の NVMe SSD – PCIe®5.0 環境の PC を組むならこれ！

EXCERIA PLUS G4 の NVMe SSD は、最新の PCIe 5.0 インターフェースに対応し、最大シーケンシャルリード性能 10,000MB/s という高速なデータ転送を実現。これまでの PCIe 4.0 SSD を凌駕するパフォーマンスを、手ごろな価格帯で手に入れることができます。さらに、高い電力効率と優れた放熱性も兼ね備えており、PCIe 5.0 に乗り換えてパフォーマンスを求めるゲーマーやクリエイターにも最適な SSD となっています。PCIe 5.0 対応のマザーボードが増えてきている中、PCIe 5.0 対応 PC をコスパ良く構築したい方に、おすすめの 1 台です。

図 1 ：EXCERIA PLUS G4 の NVMe SSD

EXCERIA PLUS G2 ポータブル SSD　– デザインと機能性を兼ね備えたポータブル SSD

EXCERIA PLUS G2 ポータブル SSD は、USB 3.2 Gen2 対応で最大 1050MB/s の高速転送を実現。小型・軽量で持ち運びやすく、MIL 規格準拠の耐衝撃性やアルミ筐体による放熱性も備えています。手になじむ凹面形状の洗練されたデザインは 2024 年グッドデザイン賞と 2025 年レッドドットデザイン賞を受賞。USB Type-C® to A / C to C ケーブル同梱で幅広い端末に対応し、SSD Utility によるパスワードロック[1]も可能。性能・デザイン・安全性を兼ね備えた SSD です。

図 2 ：EXCERIA PLUS G2 ポータブル SSD

まとめ

キオクシアのコンシューマー向け SSD では、エントリーからハイエンドユーザーまでお使いいただける製品をラインナップしております。使用目的や PC スペック、予算などに合わせて、最適な 1 台を選んでみてください。

また、COMPUTEX TAIPEI 2025 でお披露目した EXCERIA PRO G2 の NVMe SSD（PCIe 5.0 対応ハイエンド製品）をはじめ、各種ラインナップを拡大予定です。今後のキオクシアの活躍にご期待ください。

1　Windows11/10 x64 のみ対応（2025 年 7 月末時点）

＜コラム＞

YouTube での EXCERIA PLUS G4 の NVMe SSD の紹介

わしゃがな TV 様にて EXCERIA PLUS G4 の NVMe SSD をご紹介いただきました。声優の中村悠一様とフリーライターのマフィア梶田様に製品紹介のほか、ベンチマークテストの実演や SSD を使い始めたときの思い出などお話していただいています。ぜひご覧ください。

Hatto

自作 PC はパーツを選んでいるうちが一番楽しい。

Interop Tokyo 2025 に出展してきた件

余熱

キオクシアが 3 年ぶりに Interop Tokyo に出展しました[1]。今回の Interop Tokyo ではお笑い芸人の方と私（余熱）の 2 人だけがセミナーをやるということで、気合を入れてヨネミナーをさせていただいたので、プレゼンの内容を紹介しつつご報告させて頂きます。セミナータイトルは「SD メモリ界隈のご紹介」ということで、基本は SD カードについて話をしたのですが、一部 SSD と絡めた話もさせて頂きました。

図 1 ：ヨネミナーの様子

MIT との共同研究「BuzzCam」

「BuzzCam」は SSD 同人誌 3 で記事を書いてくださった高田さんが関わっている MIT[2] のプロジェクトで、2 つのマイクを使い周囲にいる蜂が在来種なのか外来種なのかを見分ける機能を持っています（図 2）。BuzzCam は、パタゴニアに 2 週間程度置かれるということでしっかりした作りになっています。Kero さんにイラストを描いてもらいセミナーで大きく掲載させて頂き、実物の展示も行いました（下図）。

図 2 ：BuzzCam

1 https://www.kioxia.com/ja-jp/about/news/2025/20250604-1.html

2 Massachusetts Institute of Technology

ここが変だよキオクシア

図３：コミュニケーションカラー

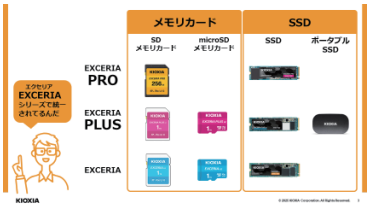


図４：ラインナップの説明

一般の方にどのくらい伝わっているのか未知数なのですが、キオクシアのロゴのブランドカラーは「シルバー」なのです[3]。それとは別にコミュニケーションカラーというものがあり、こちらは６色あって「鮮明な色により人々の高揚感（Uplifting）を表現」しています（図３）。

「キオクシアの製品はカラフルな色遣いがされている」というのは、なんとなく知っている方も多いと思いますが、それはこのコミュニケーションカラーを用いているからなのです。キオクシアの B2C ラインナップは「EXCERIA」で統一されていますが[4]、EXCERIA の中の「PRO」、「PLUS」、「無印」が SD メモリカードと SSD で全然異なっているのです。これは「コミュニケーションカラー内に優劣がない」という意図ではあるのですが、消費者からするとパッと分かりやすくはない事実であり、「色じゃなくて製品名を見てね！」ということを説明させて頂きました[5]（図４）。

（おまけ）FPGA 自作 SSD という妄想

ここのところたくさんの方が自作 SSD 基板（JISC-SSD）を使ってくださり、大変ありがたいと思っています。実は、当初考えた自作 SSD は「PCIe®/NVMe™ が動くと良いなぁ」と妄想しており、その実現の可能性を探っています。今回の SSD 同人誌 4 号を書いてくださった wipeseals さんをはじめとする方の協力により、ロジック回路の実装もある程度できそうで、クレイン電子さんは M.2 PCIe の基板を今回作ってくださったので、たとえば FPGA 版の自作 SSD っていうものをそのうちお披露目できると良いなぁ…と考えています。

3　https://www.kioxia.com/content/dam/kioxia/ja-jp/about/news/2019/asset/tmc-news-20190930-1-ja.pdf

4　ちなみに USB フラッシュメモリは「TransMemory」というサブブランドだったりします。

5　なお、覚え方として『赤飯大好き (SD) で買いまし (イエロー・マゼンタ・シアン) た。セサミソルトで (SSD) ごましお (マゼンタ・シアン・オレンジ) です』と紹介したのですが、覚えやすい！というコメントは無かったですね…。

余熱 (@yone2_net)

レトロ PC を触るのが趣味。今年は久しぶりに部門にインターン生を受け入れて、JISC-SSD を使って実習などさせて頂きました。毎度執筆 & 編集時間をくれる妻に感謝。

Maker Faire Tokyo 2024 出展と新たな出会い

Pochio

Maker Faire Tokyo 2025 の会場で、あるいは秋葉原をはじめとする各地の店頭にて本同人誌を見つけてくださった皆様、今年も SSD 同人誌の新刊を入手いただきまして、ありがとうございます。2021 年にスタートした SSD 同人誌は、このたび第 4 号を発行することができました。FlashAir™ 同人誌から数えて 11 冊目となります。こうして毎年、前回の出展の様子を振り返る記事を書き続けていますと、書き出しが毎度似た感じになってしまいますがご容赦ください（汗）。本誌につきましても、ぜひ感想をハッシュタグ「#SSD 同人誌」をつけて SNS などでお寄せ頂けますと、大変ありがたいです。その際、よろしければこれまでに入手頂いた同人誌の写真と共にツイートしてください。貴重なご意見は次回の同人誌制作の参考にさせていただきますので、よろしくお願いいたします！

2024 年は壁サーでした！

さて、Maker Faire Tokyo 2024 では「はたらく SSD、つながる SSD」をテーマに出展しました。データセンターではたくさんの SSD が 24 時間稼働していて、今日の情報化社会を支えています。したがって、SSD はもはや社会インフラの一部と言えるのではないでしょうか。そんな思いを込めて、いつものように量販店ではお目にかかれないデータセンター向けの SSD 製品や、パソコン等でお使い頂けるコンシューマー向けの SSD 製品、そして当社の SSD が国際宇宙ステーションでの科学実験で活用されていることをご紹介するパネルを用意しました（図 1）。もちろん、この年の新刊の SSD 同人誌 3 号も無事発行し、たくさんの来場者に手に取って頂きました。

図 1 ：各種 SSD 製品とパネルの展示

図 2 ：ついに壁サーになりました！

ところで、この年はめずらしく壁際にブースを構えました（図 2）。いわゆる「壁サー」ですね。ありがたいことです。「壁サー」が分からない方に説明しますと、日本最大の同人誌イベントが夏と冬に Maker Faire Tokyo と同じ東京ビッグサイトで開催されますが、非常に人気のあるサークルは多くの来場者が並ぶと予想されるため、壁際にブースが配置されます。このようなサークルは「壁サー」とよばれています。そんな壁サーの当社ブースですが、2 日間の開催を通しておよそ 2000 人の方にお越しいただきました。毎年お越しいただいている方も、初めての方も、本当にありがとうございました。

シリコンウエハーうちわの新作登場！

この年は久しぶりに本物のシリコンウエハーを展示しました。キラキラに輝くこの円盤は、当社の 3 次元フラッシュメモリ、BiCS FLASH™ の第 8 世代のシリコンウエハーです（図 3）。単結晶のシリコンをスライスしたウエハー上に、フラッシュメモリの回路を作りこんでいます。写真では虹色に輝いていますが、これはウエハー上に形成された電子回路のパターンによって光の干渉が生じているからです。

そしてこのシリコンウエハーを元に新たに制作したのが、「シリコンウエハーうちわ Ver. 3」になります（図 4）。SSD 同人誌 3 号に写真を掲載しました「シリコンウエハーうちわ Ver. 2」は、第 3 世代の BiCS FLASH™ のウエハー写真をベースに、表面に無地のホログラムフィルムを貼ったものでした。新作の Ver. 3 は第 8 世代のウエハー写真をベースに、光の干渉模様がよりはっきり見えるよう、格子状のパターンが入ったホログラムフィルムに変更して本物らしさ目指した渾身のうちわです（謎）。こちらは当社ブースにお越しの方に配布しました。直径 30 cm もあるため少し扇ぎづらかったかもしれませんが（汗）、みなさん記念に持ち帰られていました。ちなみに半導体の講義をされている大学の先生からは、「軽くて割れないので教材に最適」と、前作に続いてとても好評でした。

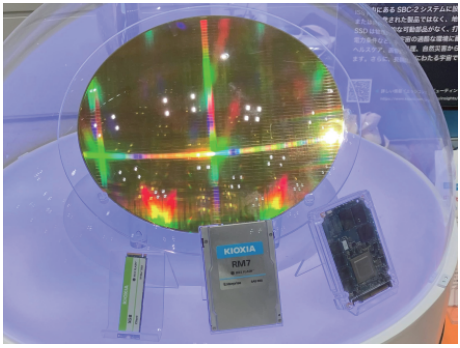


図 3 ：第 8 世代 BiCS FLASH™ のウエハー

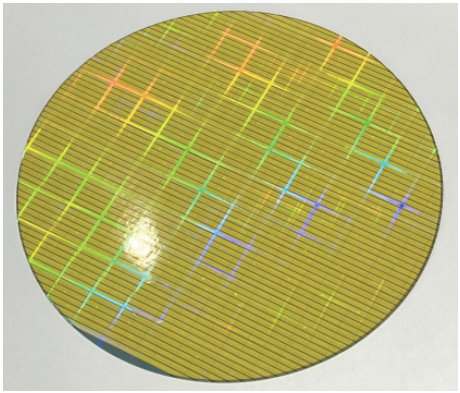


図 4 ：シリコンウエハーうちわ Ver. 3

SSD 同人誌 3 号の発行

図 5 ：SSD 同人誌 3 号

2024 年は SSD 同人誌 3 号（図 5）を制作し、開催 2 日間で約 2000 部を配布しました。連載中のまんが「データセンターてんこ」は、さくらインターネット様にご協力いただきまして、宍戸ちゃんとてんこがデータセンターを見学する特別編でした。また、CFexpress 変換アダプタを用いて当社製 NVMe™ SSD をミラーレス一眼カメラで試した「CFexpress でもキオクシアを使いたい」や、SSD 同人誌 1 号から連載中の「大容量 SSD と OpenStreetMap その 3」、そしてパタゴニアで環境把握のために自作 SSD 基板を使って製作したセンサノード初号機の概要をはじめとする、自作 SSD 基板関連記事を掲載しています。そのほか SSD 同人誌があちこちで露出している件や、当社の SSD が国際宇宙ステーションに行った件、まんが「SSD のガベージコレクション」と「バッファ発 SSD 着 FS 鉄道のお仕事」、NVMe™ over Fabrics(NVMe-oF™) や Open Compute Project といった SSD に関するトピックや、当社のコンシューマー向け SSD と SSD Utility をご紹介した記事などを掲載しました。またしても前号よりもページ数が増えてしまった SSD 同人誌 3 号は、SSD 同人誌 1 号および 2 号と同様に無料で PDF でお読みいただけます。よろしければぜひご覧ください。

SSD 同人誌 3 号はこちら

よりマニアックな自作 SSD 基板作品たち

ブースでは自作 SSD 基板を活用した作品をいくつか展示しました。まずは、「本体外観にメモを残せる SSD」です（図 6）。こちらはイサリさんの作品で、自作 SSD 基板に電子ペーパーディスプレイをつなげています。そして、自作 SSD 基板を USB 接続の SSD としてパソコンに認識させます。電子ペーパーディスプレイにはその SSD のボリュームラベルが表示されるようになっていて、Windows 上でボリュームラベルを変更するとディスプレイで確認ができるという実験的な作品です。ボリュームラベルをメモ代わりに使えますね。

つづいて、にちかさんの「NAND 型フラッシュメモリ製 16bit 自作 CPU」です（図 7）。こちらは NAND 型フラッシュメモリのメモリセルを使って、CPU を作ってしまおうというものです。「ちょっと何を言っているのかよくわからない」と思われそうですが、にちかさんがメモリセルを使って論理演算を実現するアイディア思い付き、実際に動かせるかを確認

しようと取り組んだ作品です。自作 SSD 基板に搭載されている NAND 型フラッシュメモリとマイコンで CPU を作り、動作する様子を展示しました。

しかし、メモリセルへのデータの読み書きが遅いため、動作周波数が約 2Hz と大変遅くなってしまいました（汗）。会場ではひたすら足し算をするデモを来場者にご覧いただきました。また、（メモリセルを使うと遅いので）自作 SSD 基板搭載のマイコンだけを使ってにちかさんのアイディアを再現し、ゲームを動かすデモもご覧いただきました。このデモをご覧になられた大学の先生から、お褒めいただいたと伺いました。これら 2 つの展示作品に関する詳細は SSD 同人誌 3 号に記事を掲載していますので、よろしければそちらをご覧ください。

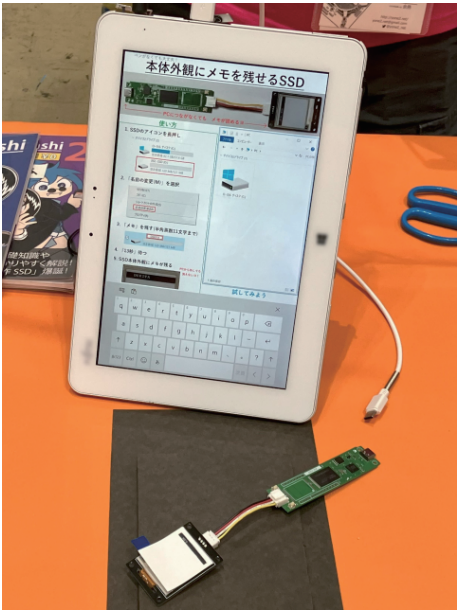


図 6 ：本体外観にメモを残せる SSD

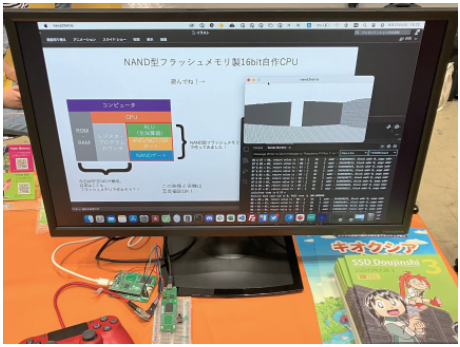


図 7 ：16bit 自作 CPU のデモ

電子情報通信学会の研究会で招待講演

電子情報通信学会は、電子工学や情報通信工学に関する研究や技術の発展を目的とした学会です。2025 年 1 月に、この電子情報通信学会の VLSI 設計技術研究会とリコンフィギャラブルシステム研究会が、当社の横浜テクノロジーキャンパスにて開催されました。その研究会の招待講演で、にちかさんに先述の「NAND 型フラッシュメモリ製 16bit 自作 CPU」を題材にお話し頂きました（図 8）。講演タイトルは、「フラッシュメモリの読み書きによる CPU 処理再現の試み」です。

NAND 型フラッシュメモリのメモリセルは、データが消去されている状態が「1」、データを書き込んだ状態を「0」と定義しています。一度でも「0」を書き込んだメモリセルは、消去するまでは「0」の状態になります。これを利用して、にちかさんはメモリセルに「1」か「0」を連続して 2 回書き込んだ後の状態が、論理回路の 2 入力 AND の演算結果と同じことに気が付いたのでした。

図８：にちかさんによる招待講演の様子

図９：研究会会場での質疑応答

これ以降は専門的な話で紙面が足りないこともあり、詳細は SSD 同人誌 3 号に譲りますが、にちかさんの構想は上記の考え方をベースに「メモリセルを使った CPU が作れるのではないか」、というものでした。この構想に基づいて、にちかさんは NAND 型フラッシュメモリを用いた CPU のアーキテクチャを考え、Flash Memory Only Processor（略して FMOP）と名付けました。にちかさんは FMOP のエミュレータを作成し、パソコン上で実験されました。詳細は割愛しますがゲームが動くことを確認し、その動作周波数は約 10.4 Hz だったそうです。講演後は先生方との活発な質疑応答がありました（図 9）。

以上、Maker Faire Tokyo の展示に向けて考えたアイディアが、どういうわけか研究会の招待講演につながり、参加されていた大学の先生や学生の皆様に楽しんで頂けたというエピソードでした。にちかさんの講演内容の詳細は VLSI 設計技術研究会の研究報告（VLD2024-94）、およびリコンフィギャラブル研究会の研究報告（RECONF2024-124）に掲載されています。一般の方には入手困難かもしれませんが、大学などでは入手できるかもしれません。

こうした活動を通して、今後も当社の SSD 製品に興味をもって頂ければと考えています。ちなみにこの話、あとちょっと続きがあったのですが本誌の印刷までに公開に至らなかったので、それはまた次回ご紹介できればと思います。

wipeseals さんたちとの出会い

さて、最後にこの SSD 同人誌 4 号に寄稿頂いています wipeseals さんについてのエピソードを書かないわけにはいきません。ある日にちかさんが自作 SSD 基板のデータシートを見るために「JISC-SSD[1]」をキーワードにして検索したところ、ランキング上位にとあるblog が表示されました。検索結果の表示は「Raspberry PI PIO で NAND アクセスを高速化する - wipeseals」とあり、気になってアクセスしてみると、なんと自作 SSD 基板を使ったかなりマニアックな記事だったのです。その記事では、自作 SSD 基板に搭載されている当社製の NAND 型フラッシュメモリのチップにアクセスするのに、GPIO を使うと遅い

1　Jisaku In-Storage Computation SSD 学習ボードの略称

から Programmable IO でアクセスを試みていて、しかもデバッグしやすいようにご自身でシミュレーション環境を作ってしまっていたのでした。にちかさんは、これほどまでに熱意をもって自作 SSD 基板を触ってくれる人がいることにとても感銘を受けたそうです。記事を書かれた方は、どうやら wipeseals さんという方で、検索しても人物像は謎でした。

2025 年の 5 月中旬、余熱さんと私はにちかさんから wipeseals さんの記事について教えていただきました。私が真っ先に感じたのは、FlashAir 同人誌の時にあった出来事のデジャブ感でした。2014 年に FlashAir 同人誌の第1号を Maker Faire Tokyo で配布して数ヶ月後、FlashAir を使って鉄道模型の制御と自動運転を実現してしまった綾瀬ヒロさんに出会いました。そして 2015 年に FlashAir 同人誌 2 号を配布しますが、今度は開発者が舌を巻くほど FlashAir を細かく解析し、FlashAir 上で実行する Lua スクリプト用のデバッグツールを開発してしまった北の大地の大学生、GPS_NMEA_JP さんに出会ったのでした。SSD 同人誌は自作 SSD 基板が少し難しめなせいか、なかなかユーザー側からのアクションが見えていなかったのですが、ついにまた新たな出会いが来た！と思いました。それも、かなりのツワモノ感です。

5 月下旬になると SSD 同人誌の制作がスタートし、ぜひ wipeseals さんに blog の記事を書いていただこうという話になりました。ひとまずメールを出してみて、お返事がいただけるかどうか。そしてどこにいらっしゃる方なのか、今度は四国か、九州か、もし必要なら GPS_NMEA_JP さんに北の大地まで会いに行ったように、沖縄にでも突撃する気でお返事を待ちました。

そして数日してご本人からお返事があり、そこに衝撃の事実が書いてあったのです。なんと wipeseals さんは、私の職場のある建物内でお仕事をされていたのでした（驚！）。こんなにすぐ近くにいらっしゃるとは、全く予想していませんでした。しかも自作 SSD 基板を発売当初に 2 枚購入されていたとのこと。さらに、wipeseals さんの上司の吉田さんは、自作 SSD 基板を発売直後に 5 枚ほど購入されていたことがわかりました。たしか自作 SSD 基板を最初に数枚販売したところ、すぐに売れたと聞いてニッチすぎる需要を不思議に思っていたのですが、こんなお近くで、しかもかなり局所的に 7 枚も購入されていたとは・・・(汗)。ちなみに吉田さんによると、発売が開始するのを待ち構えていらっしゃったそうです。吉田さんにも本誌に寄稿していただきました。

ようやく SSD 同人誌と自作 SSD 基板周りに様々な方が集まりつつあります。来年はどうなるでしょうか。また新たな出会いに期待しています！

Pochio（@I_love_nintendo）

元自称 FlashAir™ 芸人です。今年もお盆休みは小学生の自由研究に悩まされました。リニアモーターカーを作りたいというので説明書通りに組み立てたが動かず。安定走行させるまでに試行錯誤しました。これ、誰の宿題ですか？（汗）

JISC-SSD の NAND 型フラッシュメモリ制御を PIO で高速化

wipeseals

JISC-SSD (Jisaku In-Storage Computation SSD 学習ボード) の設計情報を初めて見たときに、メインのマイコンに RP2040 (Raspberry Pi Pico) が搭載されていることに気が付きました。RP2040 には PIO (Programmable I/O) と呼ばれる小型のシーケンサのような面白いハードウェアが搭載されています。これを使って NAND 型フラッシュメモリ[1]（以下、NAND Flash）との通信を高速にできたら楽しそうと考え、試した内容[2]を紹介します。

ハードウェアの概要

RP2040 PIO

PIO は CPU とは独立して動作する、主に I/O 処理向けの非常に小型のプロセッサ (State Machine) です。メインのプロセッサとは FIFO を用いてデータのやり取りを行い、プログラムは専用メモリに格納します。

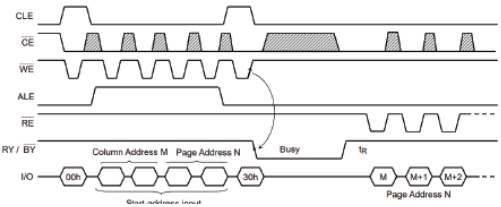


図 1：Read Timing

命令の種類や機能は少なく、プログラム用のメモリも 32 命令分しかありませんが、その表現力の高さから様々な機能が実装できます。RP2040 のデータシート[3]中でも SPI, UART, I2C 等の例が紹介されています。

NAND Flash

過去の SSD 同人誌でも取り上げられており、前回の SSD 同人誌 3 であれば宮内さん、GPS_NMEA_JP さんが基本的な使い方を実際に動作するコードと合わせて詳しく解説されています。通信は図 1 のような波形となっており、IO 線 8 本 + 制御線 (/CE, CLE, ALE, /WP, /WE, /RE, RBB) を使用します。/WE や /RE のエッジをトリガにし、残りの制御線や IO 線を使うことで、Read や Program、Erase をはじめとしたコマンドを発行し、併せてアドレスやデータを転送することでデータを入出力します。詳細は 図 2 に引用した Logic Table に記載があります。

Table 2. Logic Table

	CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$*1
Command Input	H	L	L	⎍	H	*
Data Input	L	L	L	⎍	H	H
Address Input	L	H	L	⎍	H	*
Serial Data Output	L	L	L	H	⎍	*
During Program (Busy)	*	*	*	*	*	H
During Erase (Busy)	*	*	*	*	*	H
During Read (Busy)	*	*	H	*	*	*
	*	*	L	H (*2)	H (*2)	*
Program, Erase Inhibit	*	*	*	*	*	L
Standby	*	*	H	*	*	0 V/V_{CC}

図 2：NAND Flash ピン状態と動作の関係

1 TC58NVG0S3HTA00 データシート https://www.kioxia.com/content/dam/kioxia/newidr/productinfo/datasheet/202502/DST_TC58NVG0S3HTA00-TDE_EN_31435.pdf

2 https://github.com/wipeseals/nandio.pio

3 https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

検証環境構築

手軽にデバッグしたい

設計内容は後述しますが、今回のチャレンジは PIO 向けのプログラムを書いて動かして一発 OK とはなり得ないことは容易に想像がつきました。一方で PIO のデバッグ環境はあまり充実していないため、実機で波形を見ながらのデバッグにならざるを得ません。これでは検証容易性を損なってしまうため、設計より前に検証環境構築に取り掛かりました。

Python でテスト環境を構築する

PIO の Emulator を調べるといくつか実装例がありましたが、今回は Python から利用できる rp2040-pio-emulator[4] を採用しました。合わせて WaveDrom[5] を使用し、図 3 のように PIO の状態や GPIO の状態を波形として出力できるようにしました。

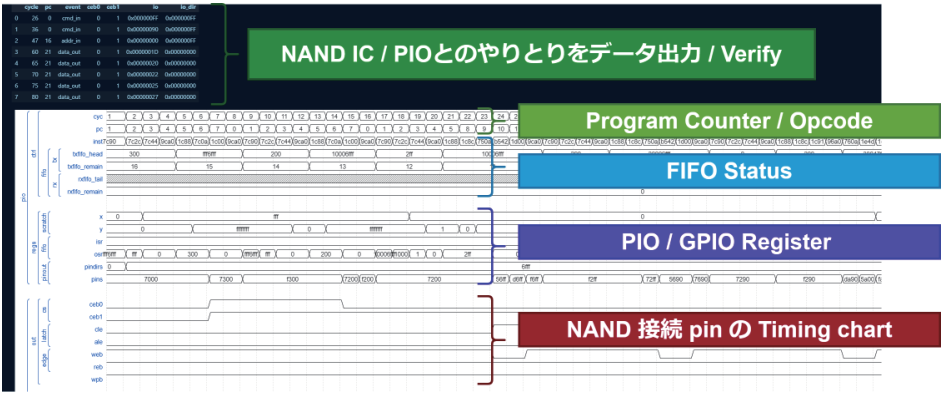


図 3：PIO の波形とデータ出力

波形を見ずに Pass/Fail 判断したい

波形を見ることで期待通りの動作か確認できるようになった一方で、毎回波形を確認するのは億劫です。そこで、各状態を pandas.DataFrame[6] に蓄積し、これをデコードして期待通りのコマンドやアドレス、データの転送が行えているか検査できるようにしました。

CI/CD 環境整備

ここまででもかなり開発の助けになったのですが、PIO や付随する実装を変更したときに意図せず不具合を入れ込まないように CI/CD 環境も整備しました。先述の Pass/Fail 判断を単体テストとして整備し、GitHub Actions で実行できるようにしました。波形の様子も見られるように、簡単な html report も都度生成し公開しています[7]。

4 https://github.com/NathanY3G/rp2040-pio-emulator
5 https://wavedrom.com/
6 https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
7 https://wipeseals.github.io/nandio.pio/

PIO 設計

方針

NAND Flash のコマンドに応じて個別の PIO プログラムを実行する方法や、GPIO の出力を Buffer 上に構築して流す案などいくつか想定はありました。いくつか試作を行い、32 命令に抑える必要があることや、Buffer 上のデータ構築をできるだけ CPU に優しい形で行えるほうが後々使いやすいことから、Buffer から流し込むデータに独自のコマンドセットを定義し、これを解釈しながら実行する PIO プログラムを作成することにしました。

独自コマンドセットの設計

NAND Flash のコマンド、アドレス、データ入力は CLE, ALE 信号状態で決定され、入出力トリガは /WE, /RE のエッジで決まります。そのため、PIO の sideset（命令を実行しつつ信号変化する機能）を用いて、表 2 のコマンドを定義し実装しました。

表 1：独自コマンド先頭ワード

Bits	Description
31:28	コマンドの種類
27:16	転送データ数
15:00	GPIO のピン方向

また、32 命令に収める都合と IO 線が双方向な都合から、表 1 のとおり先頭ワードは転送回数と GPIO の入出力も設定できるようにしています。実装は PIO アセンブラで行い、NAND Flash の AC 特性ギリギリになる部分にはタイミング調整用の NOP を挿入するなどして、高速動作できるようにしています。

表 2：独自コマンドの種類と機能

#	Command	Description
0	Bit Bang	GPIO へそのまま出力
1	Command Latch	CLE=High 状態でデータ出力
2	Address Latch	ALE=High 状態でデータ出力
3	Data Output	NAND Flash からデータ受信
4	Data Input	NAND Flash へデータ送信
5	Wait RBB	Busy の間待つ (Low の間は Busy)

独自コマンドシーケンス構築の実装

PIO は独自コマンドセットを処理するだけのシーケンサとして設計したため、NAND Flash の Read/Program/Erase などの操作ごとに、PIO に流し込む独自コマンドのシーケンスを構築する必要があります。最初は検証環境向けに Python で実装していましたが、コア部分を MicroPython でも実行できるよう調整し、検証環境 / 実機どちらでも同一のコマンドシーケンス構築ロジックで動作できるようになりました。

これを用いて Read を行った例を図 4 に示しますが、タイミングチャートに従いコマンドを並べておけば、あとは PIO と DMA が自動的にすべての転送を行ってくれます。Read 完了は DMA 転送が終わることで判断できます。Erase/Program の場合、シーケンス末

尾に Status Read コマンドを追加し、この結果が RX Buffer に転送されたら処理完了と判断でき、結果も確認できます。動作は各操作時のデータチェックとロジアナを使った波形で確認を行いました。

性能比較

FW（ファームウェア）実装（シンプルな MicroPython 実装）と PIO 実装の性能を図 5 にまとめました。ここから 2 つのことが考察できます。

1 点目は Read ID/Erase では時間差が少なく、Read/Program では大幅な改善が見える点です。これは転送データ量の多い Read/Program ではコマンドシーケンス構築で発生したオーバーヘッドより、転送時間の短縮分が大きいと判断できます。

2 点目は CPU 周波数を上げたときに PIO 実装側も時間短縮されている点です。これは、コマンドシーケンス構築に CPU 時間を多く消費していることを意味しています。対策として、構築したシーケンスを破棄せず、再利用（アドレス部だけ書き換える等）で 2 回目以後の時間短縮を狙えそうです。

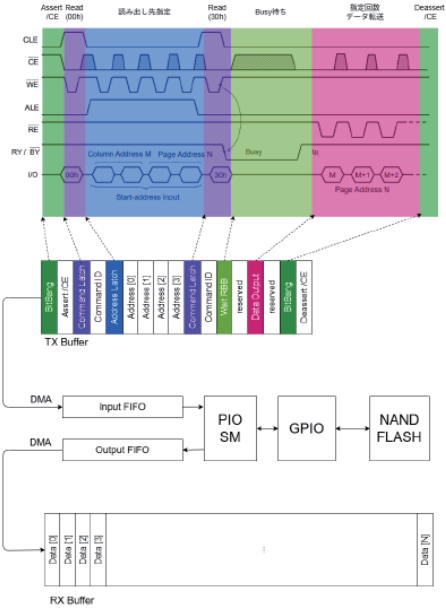


図 4：Read のデータフロー

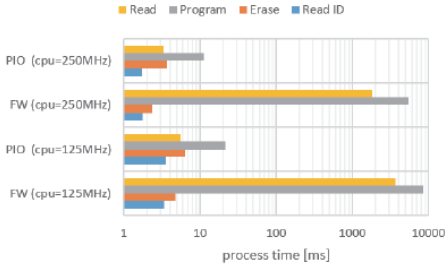


図 5：FW/PIO 実装で性能比較

おわりに

本稿では PIO を活用した NAND Flash 通信の高速化についてご紹介しました。元々は個人的な実験から始めた内容でしたが、このような発表の機会、並びに掲載におけるご支援を賜りましたことに深く感謝申し上げます。本稿が、現代のデータを支える NAND Flash や SSD、そして低レイヤー技術へ興味を深めるきっかけとなれば幸いです。

wipeseals (@wipeseals)
ものづくりと辛いラーメンが好きです。
SSD 同人誌制作メンバの熱量に圧倒されています。

キャッシュを制するもの、フラッシュメモリを制する！

宮内

はじめに

既に同人誌上で何度も取り上げられていますが、JISC-SSD(Jisaku in-Storage Computation SSD 学習ボード) は NAND 型フラッシュメモリを手軽に扱うことができる優れたツールです。前回 SSD 同人誌 3 では公開されている NAND 型フラッシュメモリのデータシートを見ながら、NAND 型フラッシュメモリを動かすための基礎となる命令の出し方を紹介しました。今回もデータシート[1]を見ていきますが、NAND 型フラッシュメモリの中身の構成について触れていきます。使っている環境は前回同様 Arduino IDE です。

不思議な現象？

前置きとして、次の 2 つのシーケンスを順番に動かしてみます。

- シーケンス 1：
  - 任意のアドレスに任意のデータを Program
  - そのアドレスに Read
  - データアウトプット
- シーケンス 2：
  - データアウトプット

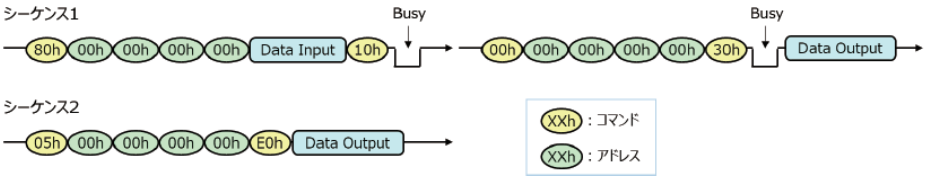


図 1：シーケンス概要

シーケンス 1 で出力されるのは、当然 Program したデータです。シーケンス 2 では NAND 型フラッシュメモリのどこのアドレスにも Read していませんが、何が出力されるのでしょう。結果はシーケンス 1 で出力されたデータがそのまま出てきます。

1　TC58NVG0S3HTA00 データシート https://www.kioxia.com/content/dam/kioxia/newidr/productinfo/datasheet/202502/DST_TC58NVG0S3HTA00-TDE_EN_31435.pdf

メモリの中にメモリ？データキャッシュの存在

ここで、データシート P.23 を見てみます。NAND 型フラッシュメモリのチップの中には、Program したデータを不揮発に保存するセル (NAND 型フラッシュメモリ領域) の他にも、“Data Cache(データキャッシュ)”、“Page Buffer(ページバッファ)” という記憶素子があります。データインプットや、データアウトプットするデータは、この 2 つの素子を通って外部とやり取りされます。

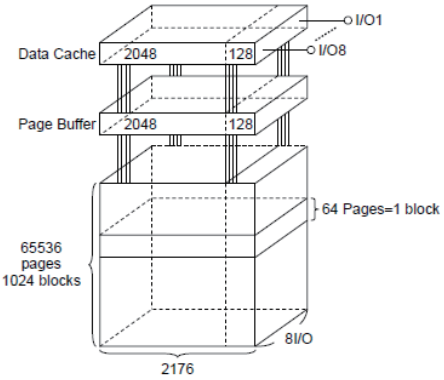


図 2 ：データシート P.23 より

データキャッシュは外部とデータの出し入れをする一時的な容器、ページバッファはセルにデータの読み書きをする一時的な容器、というような役割で、どちらも揮発性メモリです。電源が切れるとデータは失われますが、データキャッシュ・ページバッファ間でのデータ転送は非常に速く、一瞬と言ってもよいです。

さて、これらの揮発性メモリの存在を踏まえた上で、先ほどの不思議な現象を説明します。JISC-SSD は USB 接続状態＝電源投入状態では、NAND 型フラッシュメモリは常に通電状態になります。通電されていれば揮発メモリ内のデータは消えません。つまり、シーケンス 1 と 2 の間でデータキャッシュ内のデータが保持されていたため、同じデータが出力されたのです。

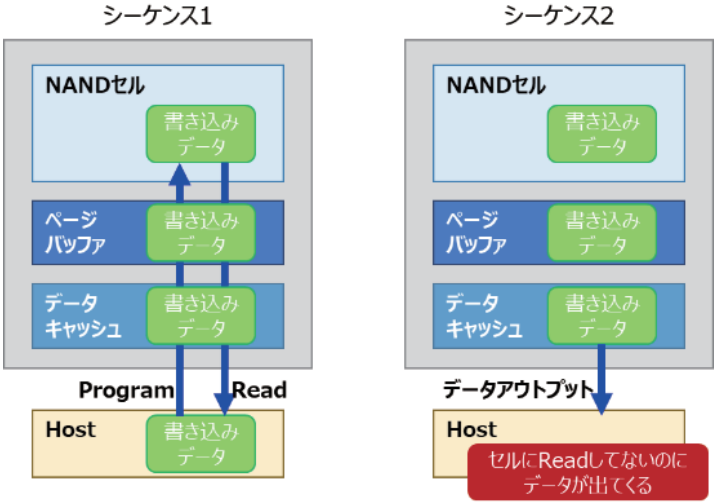


図 3 ：現象の起きる流れ

なにができるの？

これら 2 つの揮発性メモリはなんのためにあるのでしょうか。
NAND 型フラッシュメモリへの命令、Program を例に動作を分解すると以下です。

- **外部から NAND 型フラッシュメモリへデータインプット (外部が動く時間)**
- **そのデータが NAND 型フラッシュメモリ内部でセルへ書き込まれる (NAND 型フラッシュメモリ内部が動く時間)**

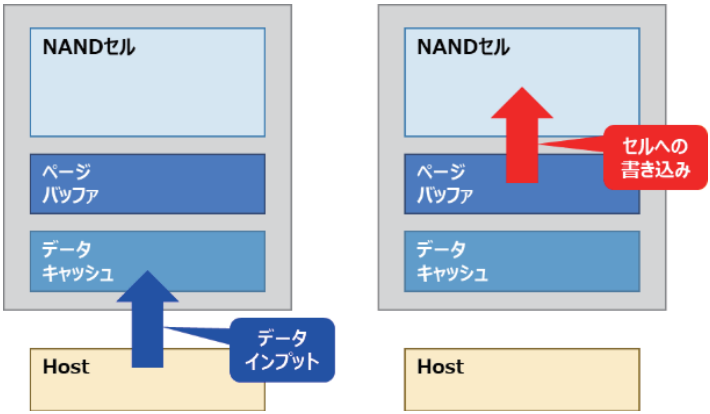


図 4 ：データインプットと書き込み

この 2 つの要素、動く部分が異なっており互いに干渉していません。同時にやってしまえば動作全体として高速化できると思いませんか?そこで 2 つの揮発性メモリ間のデータ転送は一瞬という特徴が役に立ちます。データキャッシュにインプットされたデータのページバッファへの転送が終わってしまえば、データキャッシュに残されたデータは上書きされても問題なくなり、次に書きたいデータをインプットすることができます。すると、ページバッファからセルへの書き込みと、外部からデータキャッシュへのインプットを並列で処理することができます。

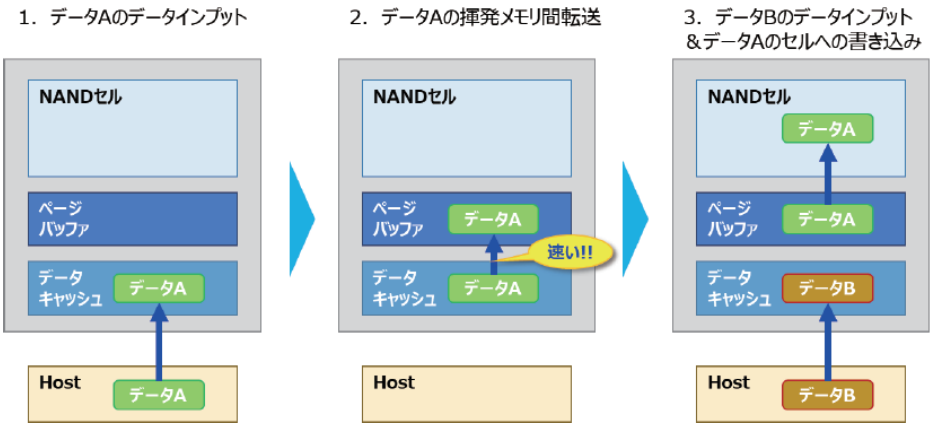


図５：Program におけるデータインプット・セル書き込みの並列処理

上記は Program を例に紹介しましたが、揮発性メモリを使った動作の高速化は Read でも可能で、Read はデータシートの P.27、Program はデータシートの P.29 に説明があります。

さらなる応用

データキャッシュの応用的な使い方として、ページコピーがあります。あるアドレスに Read を行い、そのデータを別のアドレスに Program します。このとき、データキャッシュにデータが存在することを理解していれば、Read したデータをアウトプットしたり、そのデータを再度インプットする必要はありません。データシート P.31 に例が記載されています。

さいごに

このように NAND 型フラッシュメモリのチップ内にメインの保存領域となるセル以外にも揮発性メモリがあり、データ転送の流れを理解すれば、より効率的に NAND 型フラッシュメモリを使いこなすことができます。

宮内

NAND 型フラッシュメモリにはデータシートで開示されていない隠しコマンドがあります。“上 X 下 BLYRA”と入力して「カカロット」という音声が聞こえたら成功です。

microSD Express を M.2 で！「PC-SEM2」

（株）クレイン電子 福屋新吾

M.2 スロットで microSD Express を使う基板「PC-SEM2」を開発しました（図 1）。開発の経緯などを書いていこうと思います。

microSD Express といえば、最新のゲーム機に採用され、ネット上で話題になったことは記憶に新しいと思います[1]。

SD Express・microSD Express は単なる従来の SD メモリカードを高速化したものではなく、SSD に使われている PCIe®/NVMe™ プロトコルを採用しています。M.2 スロットに刺さる変換基板が作れれば、PCIe I/F を持つ Raspberry Pi ともマッチすると思い、検討を開始しました。

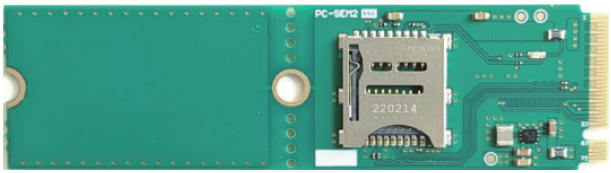


図 1 ：PC-SEM2 基板

SD アソシエーションの資料

SD アソシエーションの公開資料[2]に SD Express についていろいろと書かれています。ブロックダイアグラムでは、SD Express 内には SD I/F と PCIe/NVMe I/F の両方が内蔵されていることが書かれています（図 2）。

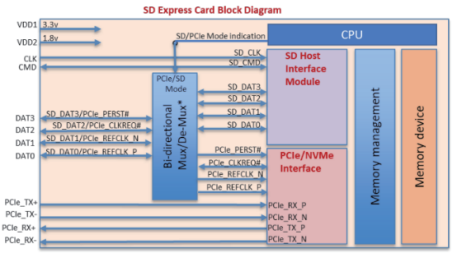


図 2 ：SD Express のブロック図

資料を読み進めていくと、SD Express は PCIe/NVMe モードのみで動かすことが可能ということがかかれています（図 3）。配線についても書かれている（図 4）ので、この通り作れば M.2 スロットで microSD Express を使う基板を作れそうです。

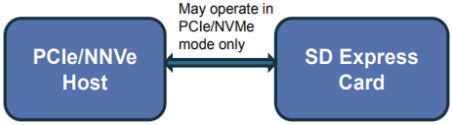


図 3 ：PCIe/NVMe モードのみで動かせる

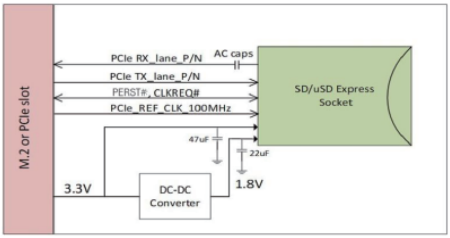


図 4 ：配線図

1　キオクシアさんでは 2025 年 10 月現在 microSD Express を作っていないです。ぜひ作ってください！

2　SD Express Host Implementation Guideline–Version2
https://www.sdcard.org/wp-content/uploads/2025/05/SD-Express-SD7.x-SD8.0-Host-Implementation-Guideline-Version-2.pdf

基板設計

M.2 スロット用基板はおおまかには Key と呼ばれる欠け pin の位置と、長さで分類されます。今回は、M.2 SSD で一般的な "M-Key 2280" サイズと、Raspberry Pi の M.2 HAT の "M-Key 2242" サイズの両対応で製作しました。部品や配線などすべて 2242 サイズ内に収めています。2242 サイズにする場合は、基板のミシン目にそって分割します。

microSD Express の 1.8V 電源はピーク電流で 600mA 程度消費します。これを満たす DC/DC コンバータとして Torex XC9286 を採用しました。XC9286 は、1A max 出力・フットプリント 1.8x2.0x0.6mm・セラミックコンデンサ対応入出力と今回の用途にぴったりでした。

動作確認

初回の電源投入はドキドキです。動作確認は PC に取り付けて試すしかなく、それは基板からも PC からも煙が立つ可能性があるわけで…(笑)

結果は、何事もなく普通に動作しました。

Ubuntu PC に CrystalDiskMark ライクな KDiskMark をインストールし、ベンチマーク測定を行いました(図 5)。使用した microSD Express カードは、Samsung BEE-A-SD01B(256GB) で、READ 性能で 807.47MB/s の速度が出ました。microSD Express の PCIe/NVMe モードは PCIe 1 レーンなので PCIe Gen3 x1 の理論値約 1000MB/s との比較となりますが、この性能は悪くない結果ではないでしょうか。また、このカードに Ubuntu をインストールして起動ドライブとして使ってみましたが、特に不具合もなく快適に Web ブラウジングなどができました。体感的には SATA SSD のマシンに近い印象です。Raspberry Pi など組込み系にうまく利用できないか期待が膨らみます。

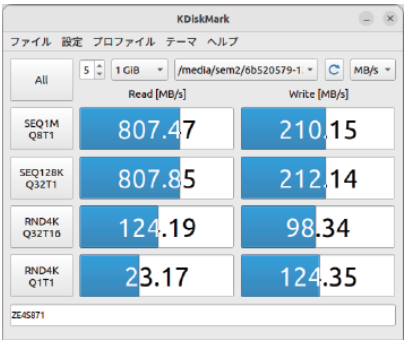


図 5 ：ベンチマーク結果

あくまで学習用の基板だった JISC-SSD とは違って、今回は PC 周辺機器の開発にチャレンジして成功です！ この変換基板「PC-SEM2」は今後発売予定です。

株式会社クレイン電子 (@crane_elec) 福屋新吾
基板の半田付けをする会社をしています。
最近は Minecraft の建築と装置作りにハマっています。

JISC-SSD に画像処理をやらせてみよう

吉田 和司

本誌でおなじみ JISC-SSD(Jisaku In-Storage Computation SSD 学習ボード) はお好きでしょうか。毎日 SSD の中でコンピューティングしたい、と思いながら生活していた私は発売日に 5 台も買いました。SSD 同人誌制作メンバーの皆様もびっくりです。今回、ご縁があって本誌制作に携わらせていただく運びとなったため、念願の夢をかなえるべく SSD の中で画像処理をしてみたいと思います。

全体の構成

SSD の中で画像処理を行うにあたり、主に以下の 3 点の機能の実装を進めることにしました。

- MassStorage クラスによる SSD 化（FreeRTOS ＋ TinyUSB）
- Bitmap ファイルの画像処理（モザイクフィルタ）
- ファイルシステムの解釈とユーザーデータの管理（図1）

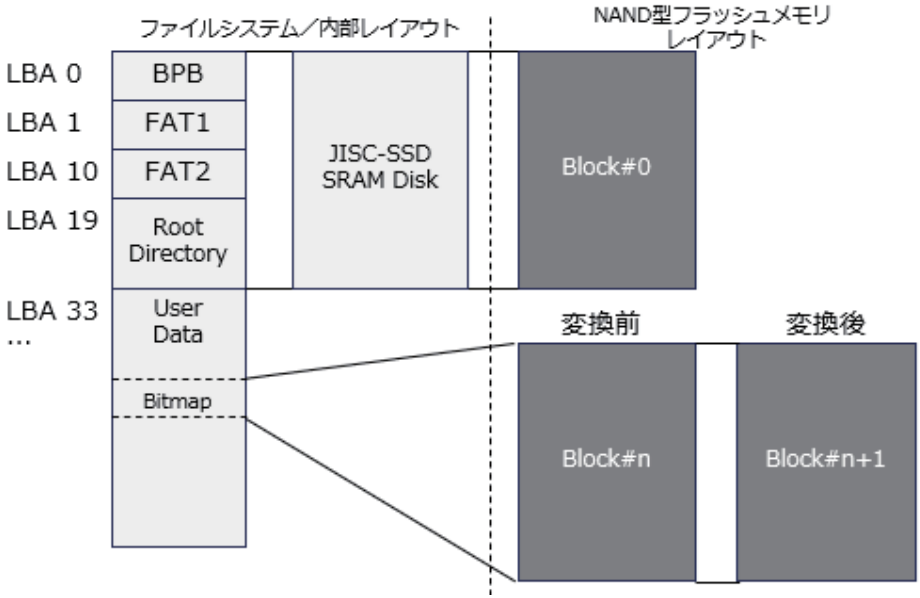


図 1：論理／物理フォーマット

やりたいことの難しさ

難しいポイントの一つにパソコンから書き込み要求が来たデータは何なのか？を理解することが挙げられます。本来、SSD はパソコン側から「このアドレス（Logical Block Address）にこのデータを書け」と言われるだけで、そのデータが何の意味を持っているか？を解釈しません。言われるがまま文句も言わずに書き込みます。実に働き者です。

しかし、今回の実装ではパソコンから書き込みが行われたデータに対して SSD の中で画像処理をおこなうため、受け取ったデータが何なのか？を理解しなければなりません。この課題を解決するために今回はファイルシステムをスーパーフロッピー形式＋ FAT12 の決め打ちにすることで解決を図りました。これにより LBA が固定され、パソコンから書き込まれたデータがファイルシステムの何の情報なのか？を把握できるようになり、Bitmap ファイルが書き込まれる場所を特定することが可能となりました。

続いて問題になるのが、いつ Bitmap ファイルの書き込みが終わるのか？を判断することです。これが分からなければ画像処理を始めることができないので重要な判断です。ファイルの書き込みが完了するまでの流れはパソコン側のオペレーティングシステムの挙動に強く依存するため、今回は以下の観点で Windows11 のファイル作成時の挙動を調査しました。

- **ファイル情報（ディレクトリエントリ）の作成／更新**
  - **ファイル名**
  - **ファイルサイズ**
  - **タイムスタンプ**
- **FAT1, 2 の更新**
- **Bitmap ファイル本体の書き込み**

調査は、図1の SRAM DISK を活用して Windows11 が書き込んでくる LBA とデータの中身を URAT 出力して行いました。これをもとに後述の全体の処理の流れを検討しています。

余談ですが、この調査の中で「Windows11 は書き込み中に電源が落ちてもいかに一般利用者のデータを1バイトでも多く SSD に残そうとしているか」の挙動も見てとれて、日々データの消失に怯えている身としては新しい気付きがありました。

全体の処理の流れ

前述までの内容で図2のシーケンスを検討し実装を進めました。FreeRTOS 上で 2 つのタスクを作成し、それぞれ MassStorage クラスの処理タスク、画像処理タスクとして定義しています。

画像処理についてはモザイク処理を行うフィルタを作成しました。昨今、プライバシー保護のため動画や写真に個人を特定できるような情報にモザイクをかけることが多いと思います。もし SSD の中で勝手にモザイク加工をしてくれる機能があると面白いかな?と思ったことがきっかけでこのフィルタを選択しました。

未来の自分には OpenCV などと連携し、顔を自動認識して SSD の中で勝手にモザイクをかけるなどの機能の昇華も期待したいところです。

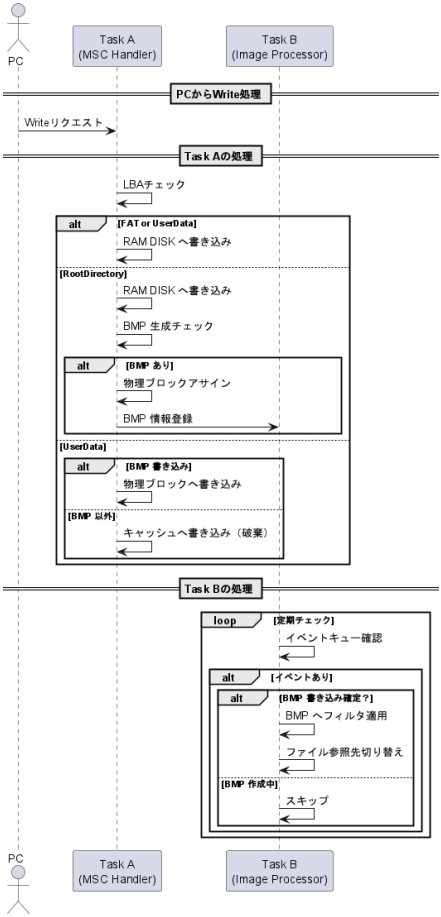


図2：全体のシーケンス図

動かしてみる

実際に書き込んでみる

今回は、自分のプライバシーを守るため、自画像を JISC-SSD に書き込んでみたいと思います。やることは簡単です。Windows11 に JISC-SSD を接続し、図3のようにエクスプローラから Bitmap ファイルを書き込むだけです。（今回は 1 MB の極小ドライブの制限）

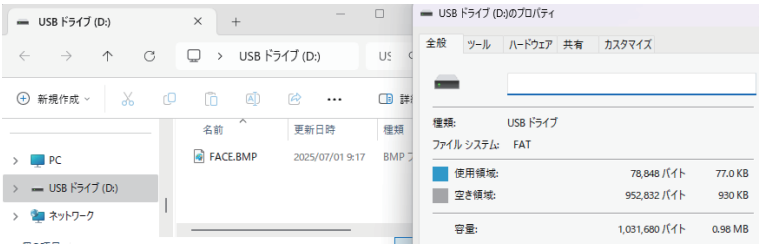


図3：自画像書き込み

実際に読み込んでみる

パソコン上では画像処理が完了したかどうかを知る術がないため、UART のデバッグログで画像処理が終わったことを確認したのち、Bitmap ファイルをエクスプローラから開いてみました。

なんということでしょう。開いた Bitmap は図4のようにまったく変化もなく、私のプライバシーが守られていません。

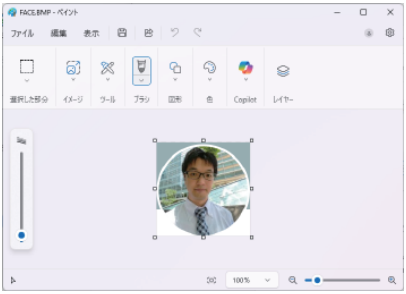


図4：自画像読み込み

なぜ失敗したのか

薄々気付いていたのですが、Bitmap ファイルを開くときに、パソコン側のディスクキャッシュにヒットしている状態なので SSD から再読み込みをしてくれませんでした。Windows11 も「SSD の中のデータが勝手に書き換わってるの!?」という気持ちだったと思います。ごめんね Windows さん、と思いつつ、今回は Windows の API、DeviceIoControl を使って強引にキャッシュのフラッシュを行い、再度 Bitmap を開きました。

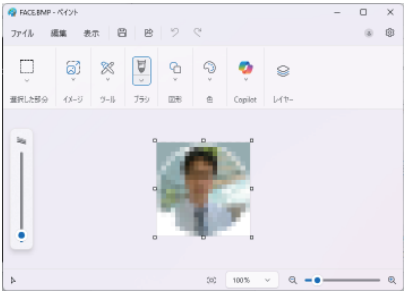


図5：強制読み込み

すると、図5のようにモザイク処理済みの自画像を開くことができ、これで私のプライバシーも無事に守られました。

最後に

今回は全国1億人の SSD ファンの長年の夢であった、演算する SSD !を自作する、をトライしてみました。この AI 時代、膨大なデータをいかに効率よく、速く、早く処理していくかは非常に重要な技術要素だと思っています。今後も様々なはたらく SSD、進化する SSD が AI 時代の基盤として活躍すると思うと楽しく嬉しいですね!

吉田 和司
ストレージにたずさわって 20 年を超えました。

消去ボタン付き見える SSD

@isariyokurari

USB メモリを棄てるとき、データは消していますか？見られて困るデータは確実に消去して、他人から読まれない状態にしてから破棄したいところです。しかし、どう消すと良いのでしょうか？ファイルを消去する、ファイルを完全に消去する、クイックフォーマットする、通常のフォーマットをする、専用の消去ソフトで消す…。ぱっと見で違いがよく分からないし、手間も時間もかかりそうな気配がします。そこで、ボタンを押すだけですぐに消去でき、消去できたことを目で見て確認できる「消去ボタン付き見える SSD」を作ってみました (図 1)。

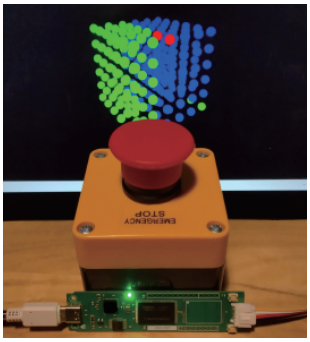


図 1 ：制作物外観

ハードウェア構成

制作物の実体配線図を 図 2 に示します。SSD 本体 には、JISC-SSD (Jisaku In-Storage Computation SSD 学習ボード) を使用し、この基板の GPIO20 と GND とをショートするプッシュスイッチを消去ボタンとしました。また、GPIO21 をソフトウェアシリアルの送信端子として使用し、表示器と接続しました。表示器は、最終的に 6 × 6 × 6 の LED キューブを用いる予定ですが、この記事向けには PC でシミュレーションした LED キューブを用いました。 1 個の LED で 128KByte ブロックの状態を表現し、LED の色は青を消去状態、緑を書込状態、赤を使用不可としました、3MByte 分の予備ブロックを持たせ、24MByte のストレージとして動作させることとしました。

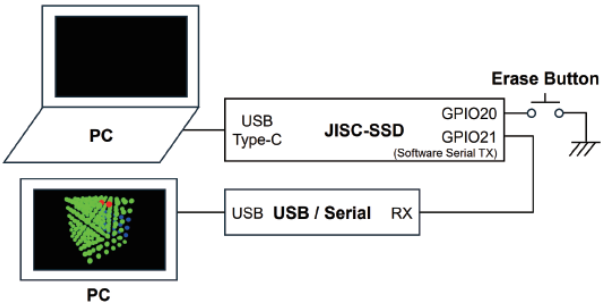


図 2 ：実態配線図

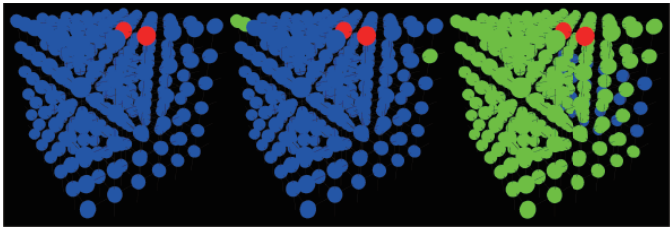
ソフトウェア構成

JISC-SSD 向けには、GPS_NMEA_JP さんのレポジトリ[1]の msc.ino を加工して使用しました。加工はまず、JISC-SSD 起動時にストレージの状態を確認して LED に反映し、

1 https://github.com/gpsnmeajp/jisc_ssd commit：3c2d4da

消去時および書き込み時も LED に反映するようにしました。次に、消去ボタンが押下されたときにデータを全て消去してストレージシステムをリセットするようにしました。6 × 6 × 6 の LED キューブを表示するプログラムは、にちかさんのリポジトリ[2]の RGB_LED_Cube.pde をベースに、シリアル通信の設定を変更して使用しました。

動作の様子



3-a　3-b　3-c

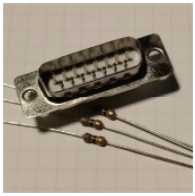
図 3：各操作後の LED キューブの様子

今回使用した個体は使用不可なブロック(赤)が 2 個含まれていました。この 2 ブロック以外消去状態 (青) というのがデータの書かれていない状態です（図 3-a）。クイックフォーマットをすると 3 秒かけて 3 ブロックに対して書き込みが行われました（図 3-b）。通常のフォーマットをすると 90 秒かけて 192 ブロックに対して書き込みが行われました（図 3-c）。対して、消去ボタンによる消去は即座に図 3-a の状態となりました。圧倒的簡単かつ高速に消去でき、そのことを目で見て確認することができました。

電子ペーパから LED キューブへの変更で自己主張強めな SSD へ

昨年、ボリュームラベルを表示する電子ペーパディスプレイ搭載 SSD を作った際、ボリュームラベルを ALLCLEAR にすると SSD 内のデータを全て消去するよう組んでありました。電子ペーパには ALLCLEAR と表示され、消去済みであることが通電なしに「見える」ストレージが実現しました。今年は消去をもっと分かりやすく！ということで、「消去ボタン」を搭載し、にちかさんの「RGB_LED_Cube」と接続、今回の記事に至りました。Maker Fair Tokyo 2025 では、本物の LED を使った LED キューブでの展示を予定しています。簡単に消去でき、それを目で見て確認できる SSD、是非体験ください。

2　https://github.com/xacas/jisc_ssd_ledCube/tree/main/RGB_LED_Cube : 423dcac



@isariyokurari

イサリです。ハードウェアとソフトウェアの両方に跨る開発が好物です。今回の開発も楽しめました。

輪唱テルミンをつくろう

山田 英樹

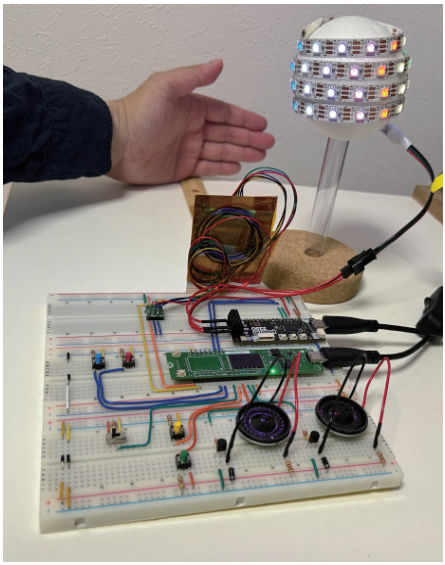


図 1：輪唱テルミン with ミラーボール

趣味のバイオリンも授業が難しすぎてついていけなくなった私は、楽器とは暫く無縁の世界で生活していた。しかしながら日々生活の中で、無性に何かを弾きたくなる時がやってくる。「ゆるふわしたのを弾きたい。」、そんなことを思っていると、モニタ画面に Mr.マリックのショート動画が流れてきた。彼はご自身のハンドパワーを用いて、コップ内に沈む 500 円玉を自由自在に操っていく。「何でも良いから、とにかくハンドパワーを感じたい!」そんなことを思っていると、私の脳裏に、ハンドパワーを使ってテルミンを優雅に演奏する情景の姿が見えた。「これは!」と思い、私は早速テルミンの製作に取り掛かることにした。

今、このページを読んでいる読者諸君も、電子部品ショップの中で立ち止まっているか、電子部品通販ホームページサイトをチンタラ見ながら、何を買おうか当ても無く彷徨っていることだろう。君もハンドパワーを感じたいだろう?一緒にテルミンを作ろう!

テルミンの構造を調べていくと、複雑なことが分かってきた。だがしかし、先人たちも私と同じことを考えていたようで、Arduino や Raspberry Pi 等を用いたテルミンの実現方法の話は数多く出てくる。とりわけ距離センサを用いた方法や明るさセンサを用いた方法があるようだ。私は距離センサベースでテルミンを作ってみることにした（図1）。ここでは Raspberry Pi Pico ベースの NAND 型フラッシュメモリ拡張ボードである JISC-SSD(Jisaku in-Storage Computation SSD 学習ボード) を使用する。

テルミンで使う部品群

実は、ここから紆余曲折があった。市販される距離センサは、超音波型とレーザー型の大きく 2 種類があり、テルミンとの相性はレーザー型の方が良いように感じた。これは今回の評価対象が、壁のように一面に広がる平面ではなく、ゴツゴツした手との相性関係に見える。今回の着地に至る使用部品例を表 1 と図 2 に記載します。（誤った箇所があればご容赦ください。これら実装は自己責任でお願いします。）

表 1 ：使用部品例

部品	個数	説明
ブレッドボード	適宜	テルミンの土台となる部分
JISC-SSD ボード	1	テルミンの演算器となる部分
距離センサ (※) （レーザー型 AE-VL53L1X）	1	テルミンのセンサとなる部分
タクトスイッチ・ スライドスイッチ	5	音の制御で使用
抵抗 (4.7k Ω)	4	スイッチ側抵抗 [スイッチ] ― [VDD] [スイッチ]―[GPIO]―[4.7k Ω]―[GND]
ブレッドボード用スピーカ (MSI28-12R)	2	ライブ再生用と録音再生用の 2 系統。 Arduino IDE tone 関数 （デジタル矩形波）前提 [VDD] ― [10 Ω] ― [スピーカ]
トランジスタ (2N2222)	2	スピーカ周りに使用 [Base] – [470 Ω] – [GPIO] [Collector] ― [スピーカ] [Emitter] ― [GND]
ダイオード (1N4007)	2	スピーカ周りに使用（逆起電力保護） トランジスタのコレクタ側と VDD に逆接続
抵抗 (470 Ω)	2	スピーカ周りに使用
抵抗 (10 Ω)	2	スピーカ周りに使用
plasma2350	1	ミラーボールで使用 JISC-SSD ボードと UART 接続
LED テープ (SK6812)	1	ミラーボールで使用
球状のものと棒状の物	1	ミラーボールの土台で使用

テルミンのコードを書いていく

コードは、Arduino IDE 環境で記述した。レーザーセンサで得られる数値は、実測値との比較をする限り、仕様通り 40cm の解像度から精度高く推定できることが分かった。但し、演奏で感情も高まった手のブレを含めた範囲だと、8 等分位が実用のところ限界のようだ。つまり 8 個の音階だと、C-Major 基準ではドレミファソラシまでしか表現できない。なので「半音上がる＃」機能を黄色ボタンに実装し、「1 オクターブ上昇機能」を緑色ボタンに実装した。これで、半音 12 諧調のド (A4) ～シ (A5) まで表現できる。n=0 が 442Hz のラ (A4) を基準に、音階 f は $f=442 \times (2^{n/12})$ で表現できる。

JISC-SSD は 3.3V ベースであり、この電圧で音出しも頑張りたいので、Arduino IDE 標準ライブラリで提供されるデジタル矩形波関数 tone() 関数を使うことにした。先程の計算で求めた音階 f を関数に指定し、青色ボタンで音を鳴らすようにする。

JISC-SSD は NAND 型フラッシュメモリが実装されているので、そこに演奏情報を全て記録させて輪唱をさせてみたい。SD メモリカードとは違い、IC のパッケージが直で搭載

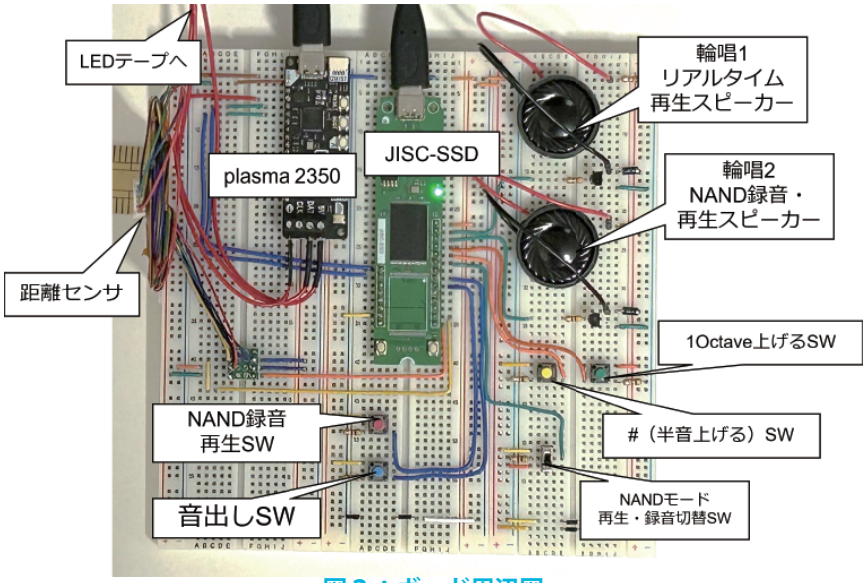


図２：ボード周辺図

される。搭載 NAND 型フラッシュメモリには Block 単位での消去、Page 単位での読み書きができる。少し判りにくいが、搭載 NAND 型フラッシュメモリは 1024Block を持ち、その 1Block には 64Page を持ち、さらにその 1Page には 2176Byte を持つ。Block 単位で消去して、そこに 2176Byte の Page 単位でデータを順次書いて、それら Page 単位で読んでいく形式となる。

GPS_NMEA_JP さんの note[1] にて、JISC-SSD 上の NAND 型フラッシュメモリが動作する API を公開下さっているので、このコードを参考にした。

ここでは 2Byte 単位で距離データとスイッチのオン・オフの全データを記録させていくことにした。ここで赤色スイッチとスライドスイッチが登場する。スライドスイッチを録音モードにして赤いボタンを押すと Block の消去をした上で、これら記憶をしていき、2176Byte 単位もしくは、赤色ボタンを押されたタイミングで、Program コマンドを叩くようにした。配列に一時的に Dump したデータを NAND 型フラッシュメモリに書いていく形だ。スライドスイッチを再生モードにして、赤いボタンを押すと Block の読出しを Page 単位で順番にしていき、配列に格納されるようにした。再生モードに入ると、NAND 型フラッシュメモリから読み出した距離情報とボタン押下情報結果からスピーカーに反映させる仕組みである。

当初 NAND 型フラッシュメモリ側が上手く動かず、暫く格闘した。格闘の末にようやく分かったのは、Erase コマンドを打ち忘れて Program コマンドをしていたことだった。ここを修正し、やっと一通り動作するテルミン輪唱楽器が完成した。

1　https://note.com/gpsnmeajp/n/n7a85a5118065

ミラーボールをつける

テルミンも出来上がり、輪唱も良い感じだが、気分を上げてくれる何かの風味が足りない。「ミラーボールが足りない!」と、その味気無さに気づいた私は、LED テープと plasma2350 を新たに用意した。ここでは、JISC-SSD 間とで UART で接続するようにし、記録したデータにあわせて LED を光らせるようにした。

球状のものと棒状の物を用意し、球体を棒に立てた状態から球体に LED テープを貼っていく。私が不器用なのか、当初、想像したように球体全面に貼れず、とても難しい (図 3)。この無様な不器用加減には、どうかご愛敬で許してほしい。

いざ光って輪唱が始まると、場も締まって、輪唱に伴う LED が気分アゲアゲ♂に盛り上がる。一方で、アゲアゲなのは良いが、LED の光り方がとても重要なことに気づく。ムーディーな演奏をしているのに LED がド派手に光ってくる。演奏で空気読んでくれる光り方制御が次の課題となった。

図 3 ：ミラーボール

山田 英樹
ポテトサラダばかり食べているプログラマ

SCA プロトコル紹介

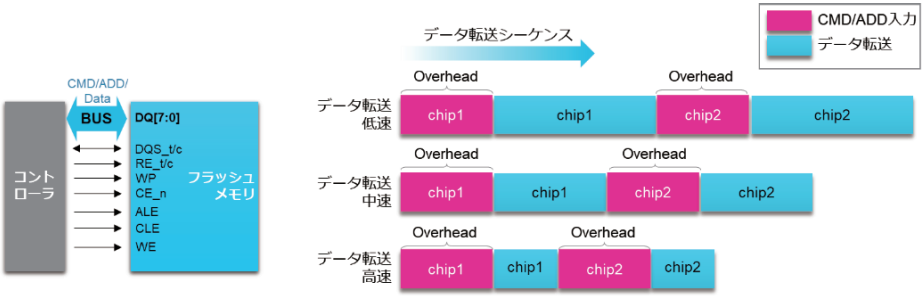
ばいむ

NAND 型フラッシュメモリのインターフェース速度高速化に伴い、データ転送時のコマンドアドレス入力時間が占める割合が相対的に大きくなり、システム性能向上のためにその対策が重要です。

一般的な NAND 型フラッシュメモリでは共用しているコマンドアドレス入力用のバスとデータ転送用のバスを分け並列で使い、データ入出力の時間を短縮可能な新機能 Separate Command Address(以下、SCA) プロトコルを紹介いたします。

データ転送時のコマンドアドレス入力オーバーヘッド

図 1a) は一般的なコントローラとフラッシュメモリの信号接続を表しています。データ転送の際は、コマンド（以下、CMD)、アドレス（以下、ADD）及びデータ自体を、8bit で構成される DQ バスを経由して送受信を行います。CMD/ADD を送信している間は DQ バスを占有するため、この間データは転送できず、その送信時間はバス効率を低下させるオーバーヘッド（以下、O.H.）要因となります。図2b) はデータ転送速度毎にデータ読出し時の CMD/ADD 入力時間とデータ転送時間の割合がどう変わるかを模式的に表しています。データ転送速度を高速化することで、データ読出し時間は短縮されますが、CMD/ADD 入力には制限があります。そのため、データ転送速度が速くなるに従い O.H. 比率が大きくなり、せっかくデータ転送速度を向上させてもトータルで必要な読出し時間は大きくは低減しません。

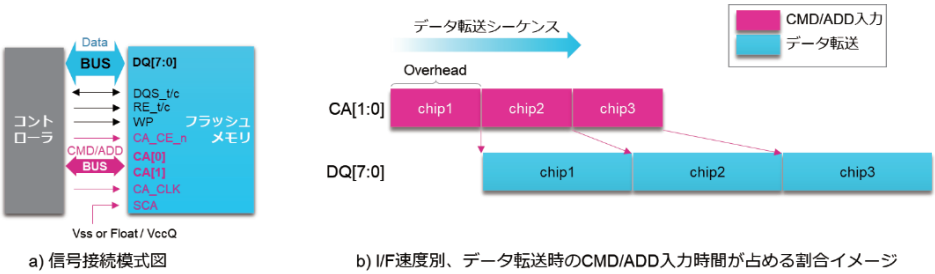


a) 信号接続模式図　　b) I/F速度別、データ転送時のCMD/ADD入力時間が占める割合イメージ

図 1：従来方式における信号接続及びデータ転送時の CMD/ADD 入力 O.H.

Separate Command Address（SCA）プロトコル

SCA プロトコルでは従来フラッシュメモリの既存の信号を生かして新たなバスを定義しています。ALE と CLE を CMD/ADD 入力用の CA（Command Address）バスとして、WE を CA[1:0] のストローブ信号（CA_CLK）として定義しています（図2a）。これにより、CMD/ADD 入力とデータ転送が完全に異なるバスに分離されます（図2b）。



a) 信号接続模式図　　b) I/F速度別、データ転送時のCMD/ADD入力時間が占める割合イメージ

図2：SCA プロトコルにおける信号接続及びデータ転送時の CMD/ADD 入力 O.H.

次に SCA プロトコルにおける CMD/ADD 入力手法を、図3にて説明します。従来手法では CMD 入力であれば CLE、ADD 入力であれば ALE を制御することで、DQ バスに入力された情報がどちらかを判別できました。一方で SCA プロトコルでは CLE 及び ALE をバスとしての役割に変更したため、このままでは CA バスから入力された情報が CMD なのか ADD なのかの判別ができません。そのため SCA プロトコルには Header と Body を 1 単位とする Packet が定義されています。Packet の先頭 4bit が Header により、後続 8bit の Body の性質を判別可能です。

Packet 制は Header 入力に時間がかかる課題がありますが、CA バスには DDR 方式が採用されており、CA_CLK 1cyc の時間が 4ns の場合、 1 Packet 入力にかかる時間は 12ns に抑えられています。

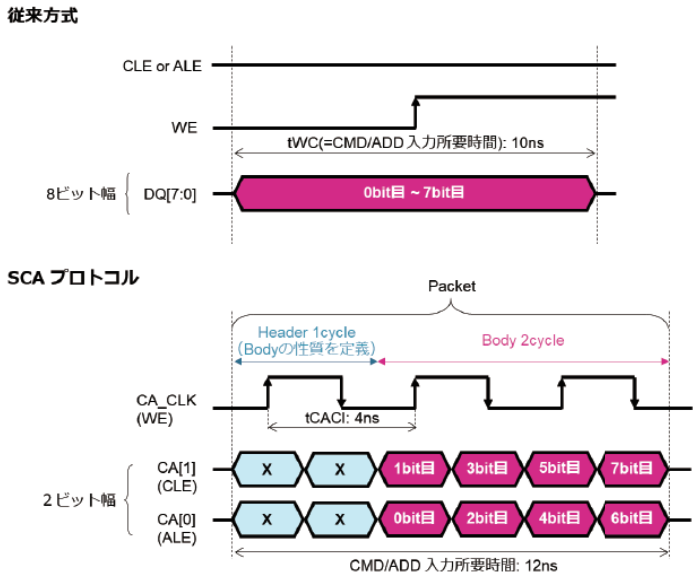


図 3 ：従来方式と SCA プロトコルにおける CMD/ADD 入力手法の違い

SCA 方式による SSD の性能向上

SCA プロトコルを採用することで Read 性能がどのように向上するかを説明します。読み出し動作における DQ バスの占有率は、メモリセルからデータを読み出す時間 (tR) と、DQ バスを通じて外部出力するデータアウトプット時間 (tDout) の関係に依存します。図 4 に、チャネル（以下、CH）あたりに接続されているフラッシュメモリが 2 つの場合、8 つの場合の例を示します。高速 I/F 速度が動作する環境では、メモリセル読み出し時間 (tR) は、データアウトプット時間 (tDout) よりも長いことから、CH あたりに接続されているフラッシュメモリが少ないと、読出しの待ち時間が発生します。この待ち時間があると、そもそも DQ バスが効率的に利用できていないため、SCA プロトコルによる効果は限定的になります。一方、CH あたりのフラッシュメモリの接続数が多いとセルからの読出しを並列で実行できることから、読出しの待ち時間が少なく、バス占有率が向上します。この環境では、DQ バスにおいて CMD/ADD の入力時間が占める割合が多くなり、その時間を削減できる SCA プロトコルの恩恵を受けやすくなります。このため、一般に SCA プロトコルは、CH あたりの接続数が多くバス占有率が高い環境ほど、効果を発揮します。

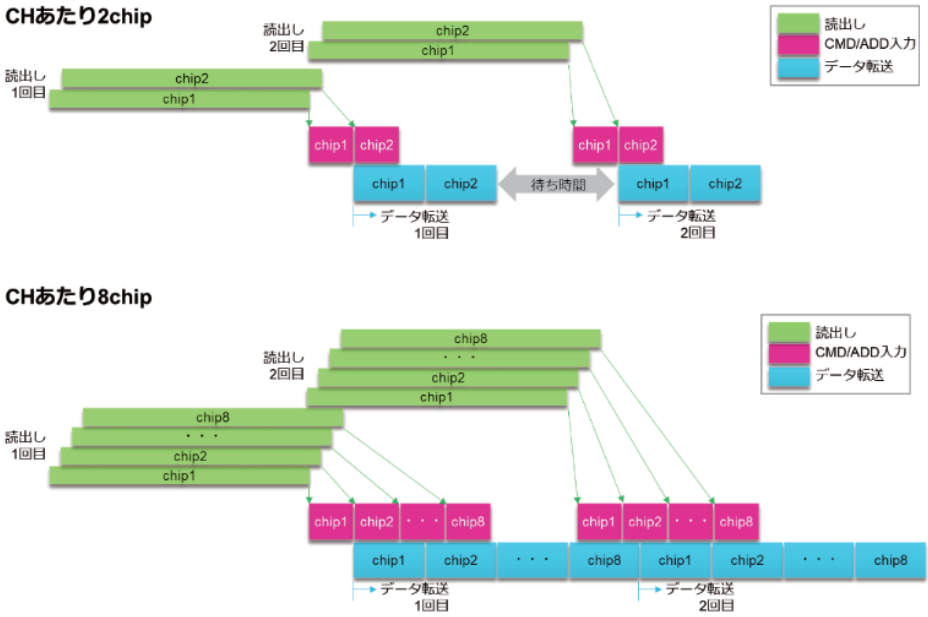


図４：CH あたりに繋がっているフラッシュメモリ数別、
読出しとデータ転送の関係図

おわりに

SSD の大容量化の流れの中でその高速化に活用される技術で、電子工作向けとしては細かすぎる話かもしれませんが、進化していく NAND 型フラッシュメモリや SSD に少しでも興味を持っていただけたら嬉しい限りです。

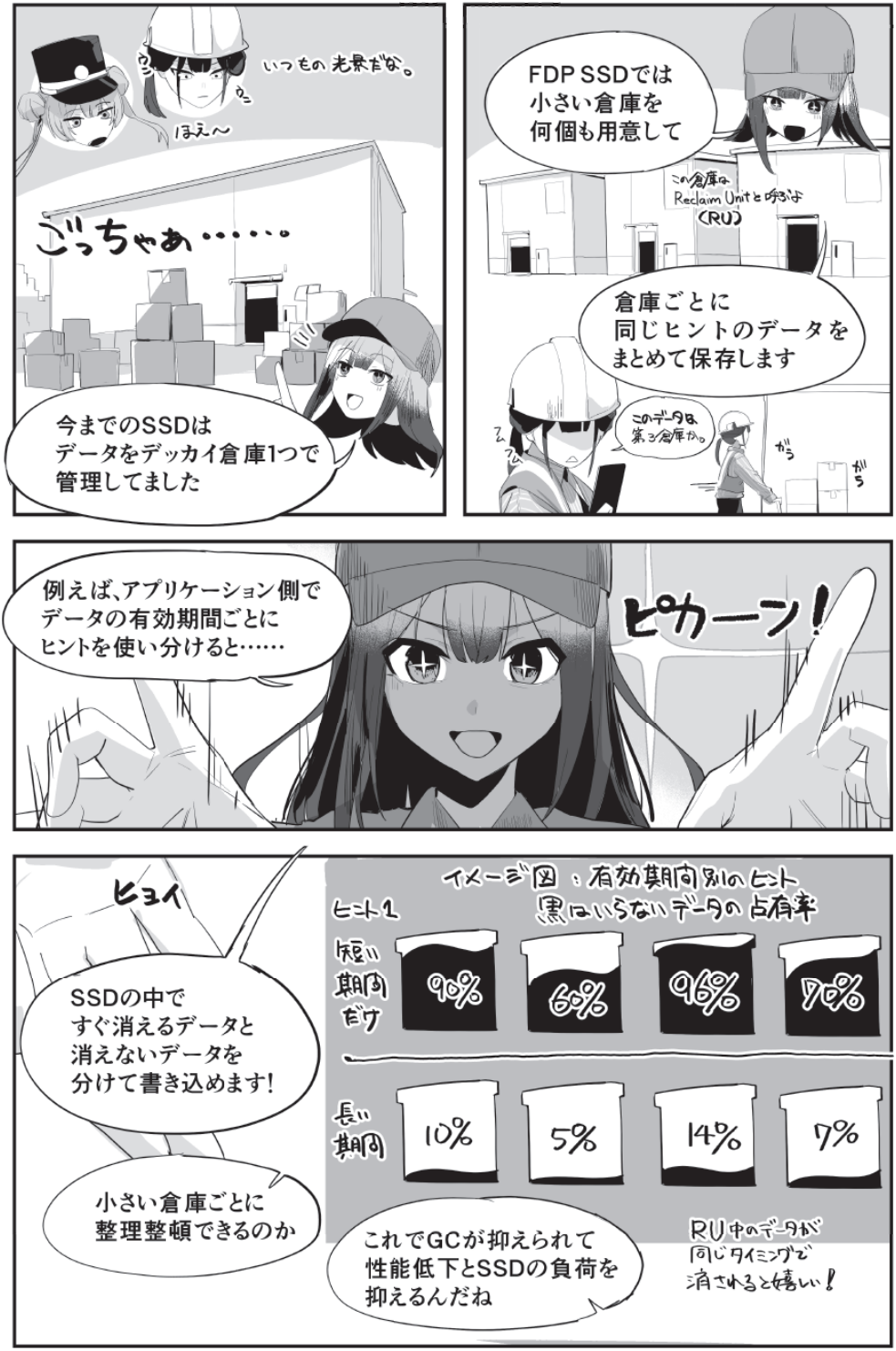
当記事の作成に多大な協力をいただいた皆様に、この場を借りて感謝申し上げます。

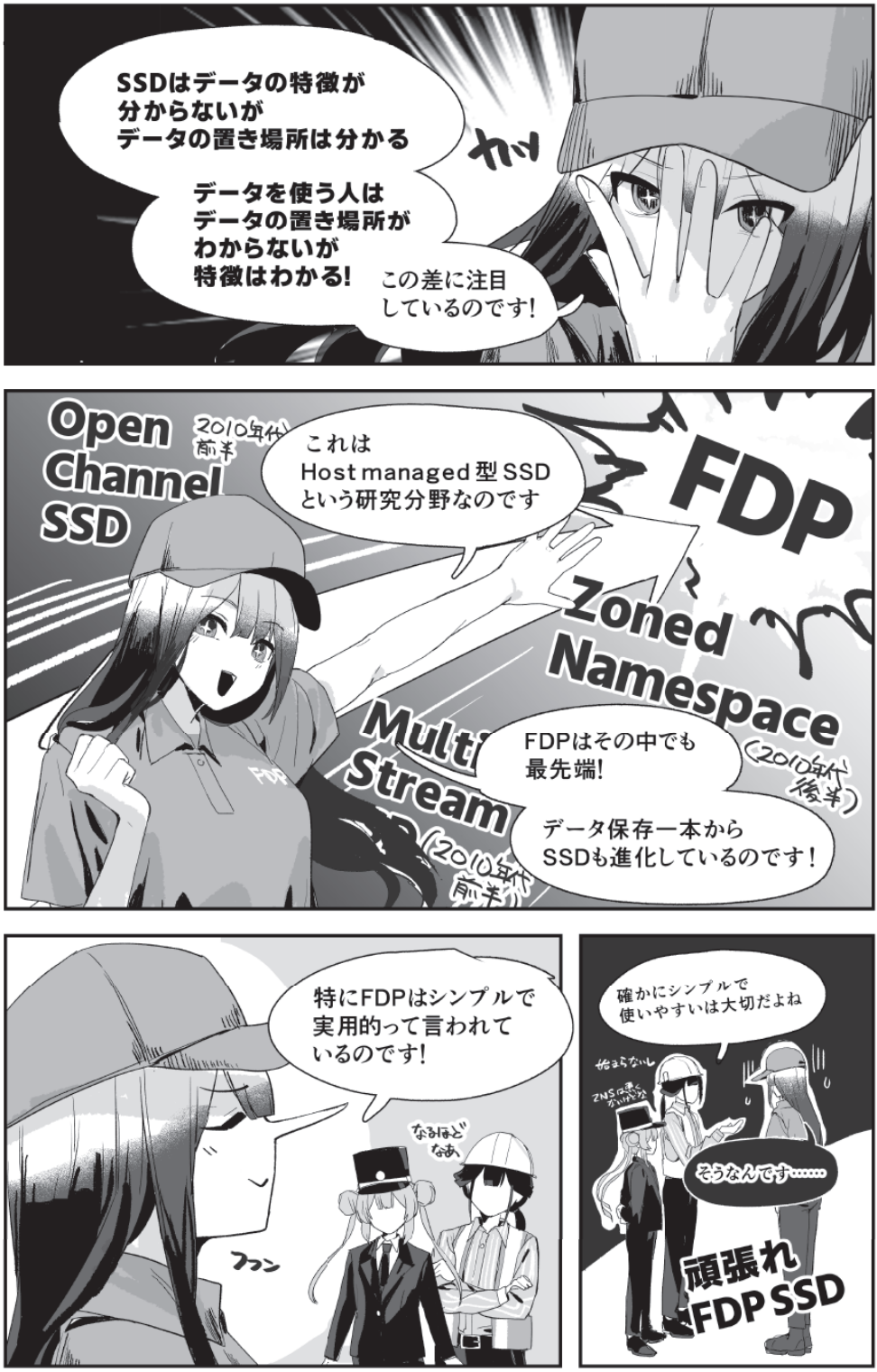
ばいむ

Maker Faire 深セン出展 (2017 年、2018 年) は良い思い出です。電子工作憧れがあり、そのうち息子達と一緒にやりたいです。名前は名字の中国語読み。


SSDコントローラー
データ倉庫管理部
荷物を運ぶFSちゃん、向かう先は――
重…
ゼェハァ
整理整頓！♪
まだデータ来るぞー
また!?
バッ
あのー!?
サッ
それは削除!
それは移動!
SSDってたまーに遅くなるけど何やっているの？
ハァ
Garbage Collection だ！（GC）
データを保存するっても場所がないから整理をしてるの！
ギュッ


コツコツ
うーん
HDDなら
その場でデータを
書き換えできるが
SSDだとそうも
行かないからねェ
大変ですねェ。
お困りですか?
えっ?
どうも
FDP SSDです!
キラッ
FDP
?
?
FDP?
何ができるの?
アプリからSSDの
データ配置を
制御できます!
Flexible
Data
Placement
の略です!


いつもの光景だな。
ほえ〜
ごっちゃあ……。
今までのSSDは
データをデッカイ倉庫1つで
管理してました
FDP SSDでは
小さい倉庫を
何個も用意して
この倉庫は
Reclaim Unitと呼ぶよ
(RU)
倉庫ごとに
同じヒントのデータを
まとめて保存します
このデータは
第3倉庫か。
がら
がら
例えば、アプリケーション側で
データの有効期間ごとに
ヒントを使い分けると……
ピカーン!
ヒョイ
SSDの中で
すぐ消えるデータと
消えないデータを
分けて書き込めます!
小さい倉庫ごとに
整理整頓できるのか
イメージ図：有効期間別のヒント
黒はいらないデータの占有率
ヒント1
短い期間だけ
90%
60%
95%
70%
長い期間
10%
5%
14%
7%
これでGCが抑えられて
性能低下とSSDの負荷を
抑えるんだね
RU中のデータが
同じタイミングで
消されると嬉しい!


SSDはデータの特徴が
分からないが
データの置き場所は分かる
データを使う人は
データの置き場所が
わからないが
特徴はわかる!
この差に注目
しているのです!
Open
Channel
SSD
2010年代
前半
これは
Host managed型SSD
という研究分野なのです
FDP
Zoned
Namespace
(2010年代
後半)
Multi
Stream
(2010年代
前半)
FDPはその中でも
最先端!
データ保存一本から
SSDも進化しているのです!
特にFDPはシンプルで
実用的って言われて
いるのです!
なるほど
なあ
フフン
確かにシンプルで
使いやすいは大切だよね
始まらないし
ZNSは速さが
そうなんです……
頑張れ
FDP SSD

- 画像は説明用です。実際の商品・サービスとはデザイン・仕様が一部異なる場合があります。
- 記憶容量 1 MB (1 メガバイト) ＝ 1,000,000 (10 の 6 乗) バイト、1 GB (1 ギガバイト) ＝ 1,000,000,000 (10 の 9 乗) バイト、1 TB (1 テラバイト) ＝ 1,000,000,000,000 (10 の 12 乗) バイトによる算出値です。しかし、1 GB=1,073,741,824 (2 の 30 乗) バイトによる算出値をドライブ容量として用いるコンピューターオペレーティングシステムでは、記載よりも少ない容量がドライブ容量として表示されます。ドライブ容量は、ファイルサイズ、フォーマット、セッティング、ソフトウェア、オペレーティングシステムおよびその他の要因で変わります。IOPS:Input Output Per Second (1 秒間に読み書きできる回数)
- 「2.5-inch」は SSD のフォームファクターを示し、SSD 自体の外形寸法を示すものではありません。
- CXL　は Compute Express Link Consortium, Inc. の商標です。
- Debian は、Software in the Public Interest, Inc. の商標です。
- Dell は、Dell Inc. またはその関連会社の商標です。
- Intel は、Intel Corporation またはその関連会社の商標です。
- Java は、オラクルおよびその関連会社の商標または登録商標です。
- microSD ロゴ、microSDXC ロゴは SD-3C LLC の商標です。
- Linux は、米国およびその他の国における Linus Torvalds の商標です。
- Microsoft、Minecraft、Windows は、Microsoft group of companies の商標です。
- NVMe、NVMe-oF は、NVM Express, Inc. の米国またはその他の国における登録商標または商標です。
- OCP、OPEN COMPUTE PROJECT は the Open Compute Project Foundation の商標です。
- PCIe は PCI-SIG の商標です。
- Python は、Python Software Foundation の商標または登録商標です。
- Raspberry Pi 、Raspberry Pi Pico は、Raspberry Pi Foundation の商標です。
- USB Type-C および USB-C は、USB Implementers Forum の商標です。
- YouTube は、Google LLC の商標です。
- その他記載されている社名・商品名・サービス名などは、それぞれ各社が商標として使用している場合があります 。
-
- 製品の仕様、サービスの内容、お問い合わせ先などの情報は 2025 年 10 月時点の情報です。予告なしに変更される場合がありますので、あらかじめご了承ください。

本同人誌に含まれる技術およびアプリケーション情報は、最新の該当するキオクシア製品仕様が対象となります。本同人誌に掲載の情報や内容については十分に注意を払っておりますが、その利用によって利用者にいかなる損害や被害が生じましても、一切の責任を負いません。必ず利用者ご自身の責任においてご利用ください。本同人誌記載の部品性能は部品単体での性能であり、自作 SSD の動作、性能等を保証するものではありません。

とだ勝之 (@katsudoren)

いよいよ本丸、キオクシア横浜テクノロジーキャンパスの SSD 開発拠点を見学させてもらいました！
技術的に凄いのは予想通りだったけど、現場での様々な工夫やベテランの経験が詰め込まれていて大興奮！面白かった〜♪

てんこが主人公の漫画「ホームセンターてんこ」は下記の二次元バーコードから是非。

Kero

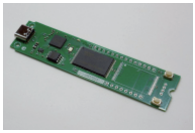
前回のきつねさんに引き続き、今回は蜂さんのイラストを描かせていただきました！

最近の業務では 3 歩進んで 2 歩下がるような日々をおくっておりますが、半歩でも前に進めるよう頑張ります！

J (@J_anonymousarts)

仕事の傍らにマンガとかイラストを描いている人。

「データセンターてんこ」の作成にあたり株式会社ＦＵＪＩ様より動画をご提供いただき、ご協力をいただきましたことを御礼申し上げます。



＜コラム＞ JISC-SSD

本同人誌に掲載される、自作 SSD（JISC-SSD）は、キオクシア製 NAND 型フラッシュメモリを搭載した、SSD の基本原理などを理解することを目的とした株式会社クレイン電子の学習向け拡張ボードです。その名も Jisaku In-Storage Computation SSD。略して JISC-SSD です。

＜コラム＞キオクシアインサイトで、広帯域光 SSD についてもっと詳しく知る

次世代グリーンデータセンターのキー技術「広帯域光 SSD」

日本発の「次世代グリーンデータセンター」を作れ！
鍵を握る「SSD ×光技術」開発の最前線

この SSD 同人誌に対する皆さまのご意見・ご感想をお待ちしています。
つぶやく時は、#SSD 同人誌 をつけてつぶやいてくださいね☆

■ SSD Doujinshi 4 - SSD の同人誌 4

2025 年 10 月 4 日　第 1 版第 1 刷発行
2025 年 12 月 25 日　第 1 版第 2 刷発行

著者 : AI ぱぱ / とだ勝之 / ragnag / J / カンミカ / 立野 賢登 / いも /
にちか / 伊藤 晋朗 / じむ / Hatto / 余熱 / Pochio / wipeseals /
宮内 / 福屋 新吾 / 吉田 和司 / @isariyokurari / 山田 英樹 / ばいむ
本文イラスト（余熱記事）: Kero
表紙イラスト : とだ勝之
表紙デザイン : 余熱 / ホーリー
編集 : にちか / Pochio / 余熱
発行 : キオクシア株式会社
連絡先 : kioxia-Ex-SSD@kioxia.com
印刷 : 株式会社 プリントパック

KIOXIA

キオクシア株式会社
〒108-0023　東京都港区芝浦 3-1-21 田町ステーションタワー S
www.kioxia.com

* 2025年8月6日現在、キオクシア調べ。