ZIPC ユーザーインタビュー（１）

通信系組み込み世界への ZIPC の適用

沖電気工業(株) LSI 事業部 ソフトウェア開発部 通信応用チーム 宮崎康弘様

－ それではまず現在の御社の開発状況につきましてですが、いかがな形で行っていらっしゃるのでしょうか？

私達のチームでは、弊社で開発しているコミュニケーション LSI に対応したソフト開発＆サポートを行っています。

－ そのコミュニケーション LSI というものを使用するためのソフト開発とサポート、と。

そうです。弊社で開発しているコミュニケーション LSI の評価ボードまたはデモボードの開発、そしてそのボードの制御ソフトの開発をしています。また、モデム LSI のような DSP(OAK※1)を内蔵している LSI やその他弊社オリジナルの 16 ビットマイコンの nX-8/500S コアや ARM※2 コアを内蔵しているシステム LSI の通信ミドルウェアの開発も行っています。

－ 開発にお使いになっているデバイスやツールの方も、それらのデバイスなのでしょうか？

評価またはデモボードの制御ソフトやミドルウェアは、弊社CPUコアを使用しています。16 ビットマイコンだと nX-8/500S コアを持つ 66K シリーズ，32 ビットマイコンだと ARM になります。また、DSP コアを使う場合は OAK になります。開発環境は、16 ビットマイコンの場合弊社オリジナルのデバッガとツール，または横河ディジタルコンピュータ株式会社様製の microVIEW-G と advice の組み合わせを使用しています。32 ビットマイコンの場合は、ARM オリジナルのツールキットや 16 ビットマイコンと同様に YDC 製の開発環境です。DSP コアの場合は、DSPG※3 で提供されているツールキットになります。

※1：米 Oak Technology,Inc.／(株)オークテクノロジー社

※2：英 Advanced RISC Machines Limited／アドバンスド・リスク・マシーンズ(株)

※3：米 DSP Group, Inc./ 日本DSPグループ(株)

― RTOS などはいかがですか？

使用しています。現在使用している OS は μ ITRON3.0 です。nX-8/500S コアでは弊社オリジナルの μ ITRON3.0 に準拠した RTOS665S を、ARM コアの場合は、(有)宮崎システム設計事務所様にて提供していただいている NORTi を使用しています。

― そういったものを使って開発される際は、どのような流れで開発されているのでしょう？

大まかな流れとしては、仕様設計→コーディング→デバッグ→総合評価と進みますが、現実にはプログラマが仕様を理解したらコーディングを始め、コーディング終了しだい評価ボードにとりあえず完成したプログラムを載せてデバッグを開始してしまいます。当然動かない(笑)。自分の作成したプログラム以外に評価ボードのデバッグも含んでいますから、いきなり動くということはまずありません。当然ツールを使って、どこが動かないの？ボードはOK？プログラムは？と調べていき、最終的に動くように持っていくといったやり方です。

― つまり、力技的な方法で、と？

そうですね。開発時間がプログラマの技量に大きく左右されます。

― すると、スキルによって作業量がばらばらになりませんか？

残念ながらそうなります。また品質も問題になります。システムとして基本的に動作していても、どこまで保証できるのかが、プログラマの技量に影響されてしまいます。作業を進めていくうえで設計者間でも共通な設計手法や評価手法を確立しないと開発や品質においてどのように進んでいるのか？どこまでできているのか？を設計者だけではなくチーム内に判断する手段がありません。

― すると、それを解決したものが理想の開発スタイルになっていくのですね。

そうですね。そこで世の中には色々CASE ツールがありますから、調査検討してきた、となりますね。

■ ■ ■

― ではここで弊社の ZIPC に対してのご質問なんですが、どちらの方で ZIPC というツールをお知りになりましたか？

まず ZIPC という名前自体を最初に私が聞いたのは、弊社マイコンをお使いになっているお客様の中で「沖

電気のツールはZIPCとはつながらないのか」という問い合わせがあったらしく、それがきっかけです。その時はZIPC って何だろう？で終わっちゃったんですが、それからいろいろCASEを検討していく中で、雑誌やショーをいろいろ見に行き存在を知りました。実際に本格的に検討したのはその段階からです。

－ では、最初の取っ掛かりというのはお客様からのお問い合わせという形だったのですね。

そうですね。

－ どうしてその際にZIPCを導入しようと思われましたか？

CASE を検討するうえで通信関係のソフトが対象となる場合、状態遷移表が主な仕様書となります。私自身が担当しているISDNだとITU-Tから定められた規格の説明の中に状態遷移図やSDL図や状態遷移表で表現されているものがあります。それを見て自分で理解して、じゃあそれをどうやってプログラムに落とそうか？また、開発するシステムに適応した状態遷移表は？という事になるわけです。そしてソフトを設計するという形を取っていたわけです。通信のプログラム開発をやる場合は状態遷移表というのは一番ポピュラーで、一番わかりやすいものですからね。視覚的には状態遷移図の方もいいんですが、状態遷移表は評価の上でも抜け目のない形で表せますしね。また、状態遷移表を書くにもExcelなどを使って表を書いて仕様書を作るんですが、せっかくそこまでやるのであれば何か生かせないかな、というところが一番興味を引いた点ですかね。

－ そうしますと、「今まではドキュメントとしてしか使えなかったものがその先のフェーズでも使える」という点が、ZIPC に興味を持っていただいた点であると言えるのでしょうか？

ええ、それが最大の理由です。特に状態遷移表でシミュレーションができる点が大きなポイントだと思います。仕様設計段階で仕様の理解，確認，仕様の評価，仕様漏れの発見等行うことができることが魅力です。またソフト設計する担当者のみでなく、チーム内やユーザ先でも状態遷移表という溶け込みやすい仕様書を提示することができます。

－ 御社では主に通信関係のシステムを開発されているわけですが、その中で ZIPC を適用したい部分といいます

のは、どの辺になるのでしょうか？

まず一番初めに導入を考えているのは、通信プロトコルの制御系ですね。現在私が担当している分野なので一番取りかかりやすい場所ですね。先程も言いましたが、規格書の中に状態遷移図やSDL図，状態遷移表といったものがありますから。

－ ZIPC の評価というのはどういった形で行われているのでしょうか？

残念ながら現状は、まだ結果を得るまでには行っていません。導入するためにいろいろ勉強しているという状態です。あとはやっぱり、状態遷移表と言っても今まで書いていたものは、純粋に「どの状態の時にどれが来て何をする」という事が書いてある、ただそれだけのものだったんですが、ZIPC を使うとなると拡張階層化状態遷移表設計手法（EHSTM）を完全に理解してないと効率的な表を作成する事ができないんで、まずはそこの技術の習得ということに重点をおいて進めている段階です。

－ この ZIPC というツールは他のツールと違い大手のツールベンダーが出したというようなものではなく、国内の小さな会社が出したツールなわけですが、評価の際にその辺でご不安になったところとかございませんでしたか？

特に有りません。弊社内でも別の部門でもすでに導入されていて使っている実績がありましたので。また日本の会社であるということは、我々が分からないところを質問しやすいというのがありますし、それと、状態遷移表が使えるというツールが調べた限りZIPC しかなかったというのもありますね。

他のツールも何点か見てきたんですけれども、状態遷移表が使えるという魅力と、あとはそれによるシミュレーションですね。そこが大きい点だったんで。どうしても、今まではできたプログラムを評価するのに実機にプログラムを載せちゃってガリガリやっていくという方法しか取ってきていないんで、やはり品質を上げるという作業が非常に難しいですし、プログラマの経験によるところが大きく出てくるんで、それを実機に載せる前にどこまで品質を上げることができるかという点に関して、シミュレーションができるというのが非常に大きい点ですね。

－ なるほど。やはり実機に載ってから何か出たのでは遅い、と？

遅いというよりもより確実な仕様を固めるという点で、「シミュレーション

できて、かつ実機でも使える」という点が大きいですね。

ー 両方でできるという点ですか。

ええ、その点は大きいと思います。実機でできる CASE というのは ZIPC 以外では見つけられなかったですね。上流設計をいろいろやれる CASE は結構あるんですけど、それを実機でやれるものは他にありませんでした。

まぁ、上流設計も大事だということは認識していますが、どうしても今まで実際にボード上でガリガリとデバッグをやってきた人間なものですから、やっぱり実機で動かないと…。

ー信用しない？

安心しないというのがあるんでしょうね。そうすると、ZIPC の場合はどちらかというと「中流・下流をターゲットとした CASE ツール」と認識しているんですけど、そこに中流の段階のシミュレーションもできて、ツールをつないで実機でも見ていくことができる、しかもそれが状態遷移表で見れる。つまり、「実機で動いているものが仕様書で見れる」というイメージなんですよね。ツール（デバッガ）側から見た場合プログラムのどこを走っているかという事はわかるんですけど、そこから仕様書を見ながら、つまり紙を別に見ながら追っていかなきゃいけない。すると、どうしてもそこはプログラマの技量によっては見落としがでるかも知れません。特に複数のプログラマがかかわっていると…。

ーでは、ZIPC で苦労している点などございますか？

御社の協力のおかげで、いろいろセミナーをやらせてもらって、基本概念というのは理解していると思ってるんですけれども、一番難しいのが拡張階層化状態遷移表設計手法（EHSTM）の考え方ですね。それを理解するのが一番苦労している点です。本などを見て、「これはこうなんだな」というのは、読めば理解できるんですけれども、自分達が ZIPC で表を書くということはそれでプログラムのスケルトンができちゃうということじゃないですか。すると、どう書いたらどう落ちるのか？という事が…。どうしても、今までいきなりプログラムを書いているんで、プログラムがどういう風になっていくのかというところまで想像しながら状態遷移表を書くことができない。まぁそれは当然使っていって慣れていかないとできない点なんでしょうけど、どうしても状態遷移表を書こうとするときにプログラムを意識してしまうん

ですよね。

それと、あとは組み込みシステムにおいて、できる限りプログラムサイズを小さくしたいとかプログラムスピード考えながらコーディングするわけですが、そうすると状態遷移表の書き方ひとつでサイズが大きくなったりだとか。やたらめったらコードが増えてしまったりだとか。そうすると、きつい部分っていうのもシステムによっては出てきますから、そこら辺も含めてどう考えて作っていけばいいのかっていう点が、まだイメージできないところなので、どうしてもまだ完全に ZIPC 導入ができてないんです。まぁそれが一度済めば、またいろいろ違ってくるんですけれども、導入の最初にあたってはやっぱり EHSTM2.0 の壁っていうのが結構高いですね。ツールの操作性とかは今のところ不満はないのですが、EHSTM2.0 考え方をどう理解するか、どううまく書けばいいのかっていう点が一番難しい点ですね。

ー コードを意識しながら作れた方がよろしいのでしょうか？

本当は CASE ツールを使うわけですから、そこまで意識する必要は無いと思うんですが、システムによっては時間的な制約が非常に高い部分、サイズ的に厳しい部分というのが出てくるわけです。その中で時間的なことで言うと、例えば関数がいっぱいできちゃうと、それだけいっぱい関数をコールしてしまうわけじゃないですか。RAMの制限もありますから、スタックのサイズも気になります。大枠で、大体こんな感じのコードに落ちるんだよ、こうやって書いた時にこうやって関数に落ちるんだよというイメージがしっかり認識できていれば、そこを考慮しながら設計することができるかな、というのはありますね。

16 ビットのマイコンなどを使用した組み込みシステムでは、やはり ROM/RAM が厳しい世界ですから、サイズは非常に気になります。我々が設計した制御ソフトやミドルウェアの部分のサイズが大きいとユーザのシステムにおいて困りますから。できる限りメモリを小さくできるものは小さくする、速くできるものは速くするという事があるんで、そういう意味からするとコードを意識できた方がいいかな、と。

ー 手法でつまずくということは、今後は EHSTM という手法に機能を追加していく方向よりは「まずどうやって使うのか」といったサポート面をしっかりさせた方がよろしいのでしょうか？

手法が効率アップする事でコード効率が良くなるだとか、そういうことに関しては当然進めてもらいたいと思います。基本的な（例えばRTOSを使うという上で必要な）機能としてはすべて押さえられていると思っているので、そういう意味じゃそこの使い方、コードの落ち方の例だとかサポートがあると取りかかりやすいと思います。

ー　総合的に、ZIPC の良い点や悪い点としてはどんな事がございますか？

機能に関しては「状態遷移表が使える」「実機のデバッグができる」「シミュレーションができる、かつそこにVIPというものがある」という具合に、ソフトの品質を上げるには十分な機能が揃っていると思っていますし、今のところ十分だと思っています。ただ、手法（EHSTM）もそうですけれども、例えば「VIP でシミュレーションの時に色々できますよ」というのはわかるんですが、VIPはVIPでまた別に設計しなきゃいけないという面が当然出てくるんですよね。もう少しその辺が楽にならないかな、というか、もっと取りかかりやすくならないかな、と。色々な部品があるとなおさら取り込みやすいというのがありますね。機能を使いやすくするための技術を習得するまでに、ちょっと時間がかかっちゃってるというのが一番ネックの点ですね。例えばVIPはVIPで専用の人がいるだとか、システムの開発人数がいっぱいいるわけではないんで。一人でEHSTM を習得しなきゃいけない、それが済んだらVIPを習得しなきゃいけないというように、一人で全部を覚えていかないといけないんです。すると、なかなか習得に時間がかかっちゃうと。

我々の努力がどこまで行くかという問題なんですけど、何かうまい習得の仕方を見つけていくことが我々にとって一番の課題となっている点ですね。

ー　マニュアルですとかその辺の資料の充実を望まれる、と。

そうですね。

ー　その他、気づかれた点やご要望などはございますか？

我々も CASE を導入しはじめたばかりですが、将来的に期待している点としては上流設計ツールとのリンク、という点がありますね。今やっているシステムが小さいシステムなんで、どちらかというとZIPCだけでも充分なシステムなんですが、だんだん大きくなってくる場合、やっぱり上流の設計と

いう点においても CASE ツールの導入というのも考えていかなきゃいけないと思うんですよ。そういう意味で、例をあげますとラショナル製の ROSE のような上流設計の CASE とリンクして ZIPC が動くとトータル的に設計システムが構築されるので、そういった面を今後は期待しています。

■ ■ ■

― 今までは ZIPC というツールについてお話いただいたんですけども、現在検討していただいているその他のツールや手法というのは何かございますか？

今のところ検討してきた中では、上流的には ROSE、と。またソフト構成管理には、ClearCASE を、と考えています。

― 通信の世界で主流の SDL ではなく、そこであえて UML 採用の ROSE を選んだ理由というのは？

えーとですね、SDL を使うのはどちらかというと中流／下流のところだと思っています。SDL はイコール状態遷移表なんで、そこは SDL で設計しようが状態遷移表で設計しようが、どちらでも良いと思ってるんです。どちらの方が見やすいかというと、それは SDL の方が図なんで見やすいとは思うんですけど、プログラムイメージで見ると状態遷移表の方が抜けの無いというか、考えやすいというか、わかりやすいと思います。すると、上流というのはその上の方を重点に置きたいんで、そういう意味じゃ UML の方がシステムの全体構成として表しやすいと思っています。UML で設計したユースケースやクラスと状態遷移表がつながってくれるとイメージ的にも一番良いだろうと。またタスクの構成を考えるとき、「タスクの中身はこの状態遷移表になってますよ」というような形ができると、一番わかりやすい、と思っています。

―上流のほうでタスク構成図があって、タスクをダブルクリックすると状態遷移表が立ち上がってくるというような、そのようなスタイルというのは？

そうですね。とても理想的だと思います。システムのイメージがわかりやすいですね。

― その他検討されている物として、先ほど ClearCASE という名前が挙がりましたが、これはソフトウェア構成管理ツールですよね？

そうですね。ソフトウェアの構成を

管理するうえで、弊社の場合、開発の部隊が東京と大阪と宮崎という風に分かれていますので、最悪三ヶ所で一つの同じ物をやっているという事も考えられますから、そう言った意味では ClearCASE のような、バージョン管理やファイル管理などのソフトウェア構成管理ができるというのは、やはり大きな魅力なんです。それが自動化できると。そこに一番メリットを感じてますね。

ー すると、ZIPC にもそういったバージョン管理やファイル管理といった機能が必要でしょうか？

そうですね、将来的には是非検討していただきたいと思っています。できあがった状態遷移表は、仕様書でもあるし、そこからコード生成するので状態遷移表＝プログラムだと思っていますので。また TEV/REV 設計書のような仕様書等も含めて管理したいと思っています。細かい仕様書に変更があると当然それはコード生成に影響してしまいますから、そういう部分の管理が自動化できれば，将来的にはいいかな、と。

ClearCASE については、導入しようというのは決定しているんですけども、我々のシステム開発において今後 ZIPC の導入がどんどん進んでいけば、そこら辺にも影響してくるかな、と。

ー 御社でのオブジェクト指向に対しての今後の取り組みというのは、どのようにお考えですか？

現在はまだ検討段階です。我々が担当しているのが組み込みシステムの中で、オブジェクト指向というのは将来的には、というのは当然認識あるんですけど、まだ実際にはオブジェクト指向を踏まえて作るというところまでは行ってないです。実際、オブジェクト指向になると当然 C++というのも言語的には出てきますし、マイコンがサポートしているコンパイラも C++対応となります。32 ビットならば良いですが 16 ビットとなると弊社 CPU ではサポートしていません。組み込み用の C++というのもありますので、将来的には取り入れなきゃいけないというのは充分わかってますし、取り入れることで上流からの設計がやりやすくなるというのは充分わかっているんですけども、現実としてはまだ我々のシステム開発場合 16 ビットマイコンが半分以上ありますので。将来のために勉強だけは進めていこうと考えてます。

ー まだ勉強中といったレベルでしかな

い、と。

そうですね。だから、例えばオブジェクト指向をやっていく上での勉強手法のひとつとして、UML などを取り入れて考えていこうと思っています。

ー 組み込みという世界へオブジェクト指向という考えが入ってくるのに、何が障害になっているのでしょう？

ひとつはコード効率だと思いますよ。

ー コード効率ですか。

一口にコード効率といってしまうと語弊があるかもしれませんが、我々の場合、先程もお話ししましたように 16 ビットマイコンによる開発が半分以上をしめています。開発環境的にもC++対応のコンパイラがないこともありますが、システムによってはメモリやスピードの制限が厳しいものがありますから、オブジェクト指向でプログラムを開発しようとすると、開発環境やシステム全体の価格等も含めてシステムを制御するCPUの選択も難しくなると思います。

RISC を使ったシステムならば開発環境もありますし、特に大きなシステムを開発するとなるとオブジェクト指向を使った方が開発効率も良くなると思います。

ー 8 ビット 16 ビットのいわゆる「旧世代」のハードウェアに、最先端の考え方を合わせることが困難であるというのが理由で、と？

ええ、今ちょうどその移行時期だと思ってるんですよね。ちょうど16ビットと 32 ビットの中間位置みたいな。実際我々の開発でも半々ですし。

ただ、今後 32 ビットでの開発は当然増えていきますから、できる限りオブジェクト指向の考え方を取り入れたいと思っています。

■ ■ ■

ー それでは最後に、皆様へのアピールしたい点ですとかはございますか？

そこが非常に悩むとこだったんですけど(笑)、我々のチームとして、まず自分たちの開発環境を変えていくという点、ソフトの品質を上げていくという点で、ZIPC を導入しました。

そうすることにより、弊社としてただ単にLSIを提供するだけではなく、品質の良い LSI を制御するソフトウェアやミドルウェアをお客様に提供していきたいと思っています。その中でも例えばモデムLSIやISDN, PHSなどの

弊社として得意分野であるコミュニケーションソリューションに関するLSIのソフト的なサポートをしていくことが我々のチームのミッションとなりますので、それらの通信LSIやCPUを利用してシステムを構築されているお客様，またこれから構築しようと思われているお客様に対して、品質の良いソフトウェアやミドルウェアを提供することをはもちろんのこと、将来的には、ZIPCをお使いになっているお客様に、弊社LSI制御用の状態遷移表などもサポートしていきたいと思っています。

また、システムLSIを構築するうえで弊社が提唱しています SPA(Silicon Platform Architecture：沖電気の保有する総合技術を結集した、標準化されたプラットフォーム)の中の、コミュニケーションにおけるSPAとして、周辺回路，バスインタフェース以外にも、例えばARMをコアにしたソフトウエア，ミドルウェア，開発システムなどをメニュー化し、お客様が自由に選択し組み合わせることで、お客様に最適なシステムLSIを簡単に、かつ短TATで開発できるようにしていきたいと思っています。

（みやざき やすひろ）