設 計 自 動 化 81－9
(1996. 10. 18)

FPGA および CAD ツールの同時評価システム

宮崎敏明　筒井章博†　石井健司†　太田直久†

NTT システムエレクトロニクス研究所
〒243-01 神奈川県厚木市森の里若宮 3-1
e-mail: miyazaki@aecl.ntt.jp
†NTT 光ネットワークシステム研究所

あらまし　本稿では、FPGA と専用 CAD ツールを同時に評価するシステムを述べる。ユーザは、専用言語を用いて FPGA アーキテクチャを記述することにより、スイッチボックス内のスイッチパターンや配線要素などを評価することができる。さらに、本システムは、ミドルウエアと称する専用 CAD ツールを容易に作成する枠組を提供している。ユーザは FPGA アーキテクチャを評価しながら、その FPGA の製造前に専用 CAD ツールを構築できる。これは、バランスのとれた FPGA と専用 CAD を開発する上で非常に有意義である。はじめに、評価システムの概要を述べた後、いくつかの適用例を示す。

キーワード　FPGA, CAD, アーキテクチャ定義, 同時評価

Co-evaluation System for FPGA's and CAD Tools

Toshiaki Miyazaki,　Akihiro Tsutsui†,　Kenji Ishii†,　Naohisa Ohta†

NTT System Electronics Laboratories
3-1 Morinosato Wakamiya, Atsugi, 243-01 JAPAN
e-mail: miyazaki@aecl.ntt.jp

† NTT Optical Network Systems Laboratories

Abstract　A concurrent design and evaluation system for an FPGA and its dedicated CAD tools is presented. By describing an FPGA architecture in a description language, the user can evaluate it, e.g., switch pattern in a switch box and routing resource balance. In addition, this system has a middle-ware which enables users to develop dedicated CAD tools rapidly. While they are evaluating their FPGA's, they can obtain original CAD tool sets before the FPGA's are manufactured. This is the best way to realize a well-balanced FPGA and its CAD system. This paper overviews the system and introduces some examples.

key words　FPGA, CAD, Architecture Definition, Co-evaluation

1 はじめに

Field Programmable Gate Array (FPGA) は、その便利さから広く使用されるようになり、多くの種類が市販されている。しかし、市販FPGAの多くは、汎用を目指しており、分野によって適用が困難な場合がある。そのため、現在でも特殊な要求を満たす独自のFPGAもいくつか開発されている [1],[2]。

一般に、FPGAは論理実現部と配線要素からなる [3]。FPGAの設計に際しては、論理と配線要素のバランスやその使用効率を十分考慮する必要がある。バランスのとれたアーキテクチャを見い出すために、理論的な手法も幾つか提案されている [3], [4]。それらの成果は、各要素の上限または下限を知る上で役立つ。しかし、仮定しているモデルが単純過ぎて実際のFPGA設計に適用することが難しい場合が多い。より現実的な評価手法として、シミュレーションを基本としたものも提案されている。Albaharna等 [5] は、FPGAの評価に既存のCADツールを適用した。彼らは、異なるアーキテクチャをもつFPGAを定量的に評価するという意味において成功している。しかし、彼らが用いたCADツールはFPGA用ではなくASIC設計用に開発されたものであった。そのため、FPGAのアーキテクチャを詳細に評価するには困難を伴う。

一方、FPGAに所望の回路を実現するには専用CADツールが必須である。実際、FPGAの性能はCADツールの良否にかかっているといっても過言ではない。よって、FPGAを設計する際にはCADツールがいかにFPGAの構造を容易に扱えるかを考慮することも重要である。さらに、専用CADツールはFPGAの設計と並行して、できるだけ迅速に開発されることが望ましい。各々のCADアルゴリズムを改善する必要がしばしば生じるからである。

市販CADツールの中には、専用CADを迅速に構築するための示唆を与える幾つかの興味深い機構を備えたものが存在する [6],[7]。Xilinx社が提供するXACTシステム [6] 内の設計エディタは、起動されるたびに"part type file"と呼ぶファイルを読み込む。設計エディタ自身は、彼らが提供する全種類のFPGAに共通に使用する。そのため、CADツールと個々のFPGAアーキテクチャを分離する手法をとっている。しかし、Xilinx社は、"part type file"を公開していないため、ユーザがそれらを独自に開発したり編集することはできない。

図研 [7] は、"Software BUS"という概念をPCB-CADシステムに採用している。これは、ユーザが独自のソフトウエアモジュールを開発し、それをPCB-CADシステムにプラグインすることを可能にしている。しかし、残念ながら彼らのシステムはFPGAの設計を対象にしていない。

バランスのとれたFPGAと専用CADを開発するためには、それらの同時設計および評価が最も良いと筆者等は考える。この方針に従い、我々は *FACT* (FPGA Architecture and CAD system Testing environment) と呼ぶFPGAアーキテクチャとCADツールを同時に評価する環境を構築した。本稿では、システムの概要を述べた後、幾つかの適用例を示す。

2 システム概要

図1にFACTシステムの概要を示す。本システムは、CADツール開発環境とFPGA評価ツール群からなり、評価ツール自身もCADツール開発環境を用いて構築されている。

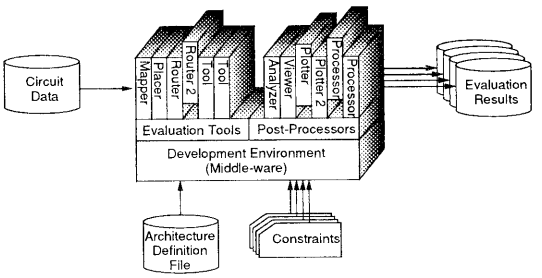


図 1: システム概要

CADツール開発環境は、多くのデータアクセスメソッドを持つミドルウエアからなり、ユーザがアルゴリズムの核となる部分を開発するだけで容易に新たなツールを構築できるようになっている。

FPGA評価ツール群は、さらに、実際の評価を行なうツールとポスト・プロセッサに二分される。前者には、テクノロジ・マッパ、配置配線ツールなどが含まれ、実際に評価対象のFPGAが開発された後は、そのまま専用CADツールとして使用できるものである。一方、後者は、評価結果の統計処理や作図処理を行なうプログラムなどが含まれる。

FPGAアーキテクチャは、ネットリスト形式の回路データを用いて評価される。入力回路は、テクノロジ・マッピング、配置配線処理を経て、アーキテクチャ定義ファイル (Architecture Definition File: ADF) で記述されたアーキテクチャにプログラムされる。すなわち、本システムは、あるFPGA用の専用CADシステムが行なっている過程を完全に模擬することができる。これは、評価対象のFPGAが実際に開発された場合、評価ツールそのものが専用CADとして使用できることを意味している。もちろん、個々の評価ツー

ルは独立に変更できるため、FPGA アーキテクチャを固定した場合、CAD アルゴリズムの評価環境としても本 FACT システムは利用できる。

本システムの特徴は、アーキテクチャ定義を評価ツールに関係なく行なえることである。ADF は、ある文法を持ったテキストファイルであり、ユーザに開放されている。ユーザは、ADF を書き換えることにより独自の FPGA 構造を定義することができる。各評価ツールは起動時に ADF を読み込み、アーキテクチャ情報を入手する。本機構は、アーキテクチャ変更に伴う評価環境の再構築作業を軽減する。すなわち、少しの FPGA アーキテクチャの変更では、各評価ツールを一切変更する必要がない。

ポスト・プロセッサは、評価の中心的なツールではないが、評価結果を図的に表示したり、まとめたりすることができ、評価結果の傾向を直観的に知る上で有効なツールである。ポスト・プロセッサもアプリケーションプログラムであり、ユーザ独自のものも FACT システムに組み込むことができる。

3 アーキテクチャ定義ファイル: ADF

FACT システムを使用するためには、最初に ADF を作成しなければならない。ADF は、評価すべき FPGA アーキテクチャを記述したファイルである。ADF は拡張性に富む独自の文法をもっている。ADF の特徴を以下にまとめる。

- 階層構造定義:

  ADF はオブジェクト指向を採り入れており、モジュール (クラス) を定義した後は、それらを階層を含む任意の場所で使用することができる。

- 属性の拡張性:

  任意の属性を各クラスに記述でき、それらはインスタンスに引き継がれる。現在、我々は"cost"と"view" という属性を、配線要素の評価とアーキテクチャをグラフィカルに表示するために用いている。

- 繰り返しパタンの定義容易性:

  配列定義と添字表現をサポートしている。多くの FPGA は繰り返し構造をしているため、アーキテクチャ定義が容易となる。

文法は、LISP や EDIF に似ている。文法の主要部分を BNF 形式で図 2に示す。

また、ADF の記述例を図 3に示す。図 3(B) は、図 3(A) の構造を表している。2-16 行は、FPGA 全体のクラス定義であり、17 行目の "a_chip" は、そのインスタンスを表している。あるクラスの中で、以前に定義されたクラスのインスタンスを用いることもできる。例えば、9 行目の "i[0..2]"は、4 行目で定義されたクラスのインスタンスであり、3 ビットの入力ピンを表している。

ADF はミドルウエア内のアクセスメソッドで読み込まれ、内部データ構造に変換される。全てのクラスまたはインスタンス内で記述された属性は、バッグ (bag) 構造としてその内部データ構造に格納され、それぞれ key によってアクセスすることができる。例えば、"(VIEW (LOC 100 100))" は "VIEW" を key として "(LOC 100 100)" を属性データとする形で格納される。(ここで、"LOC" は、さらに "100 100" のための key となる。) それゆえ、あらかじめ定めた属性とは別に、必要があれば任意の属性を定義することができる。新たに定義された属性を知らないツールが ADF を読み込んだ時は、その属性は単に無視されるだけである。また、クラス内で定義された属性と同じ key を持つ属性がそのインスタンスで再定義された場合、それらが有効になる。

4 ポスト・プロセッサ

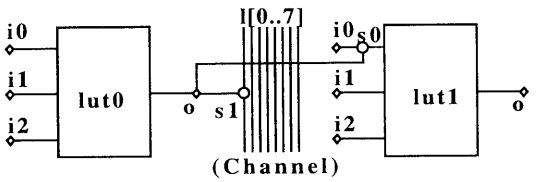
ポスト・プロセッサは、アーキテクチャ評価のためのメインツールではないが、評価ツールが出した結果を吟味する時に役立つ。ここでは、我々が用意したポスト・プロセッサの出力例を 2 つ示す。どちらも、Xilinx 社の XC4000[8] や *PROTEUS*[1] に代表される対称配列 (symmetrical array) 構造を仮定している。

最初の例を図 4に示す。これは、配線要素の消費の度合いを表したヒストグラムである。2 対で表された左側のヒストグラムは、1 本のネットで結ばれた 2 つの論理ブロック間のマンハッタン距離を示している。一方、右側のヒストグラムは、実際に配線された結果を示している。ここでは、y 方向と x 方向を別々に表示した。このヒストグラムから、幾つかのネットは迂回し、最短距離、すなわちマンハッタン距離で配線されていないことがわかる。これは、配線遅延の観点から良いとはいえない。

図 5に示した 2 つ目の例は、分散図である。図は FPGA 全体を表しており、各々の数字は、対応する配線チャネル内で実際消費されたトラック数を示している。影の付いた箇所は、比較的込んでいるチャネルを示している。本プロットは、どの程度配線チャンネルが消費されているかを概略知りたいときに役立つ。

```
adf ::= '(' DEFARCHITECTURE chip_name '(' class  instance ')' ')' ;
class ::= '(' DEFKEYWORD class_name '(' class_defs ')' ')' ;
instance ::= '(' KEYWORD instance_name '('  '(' TYPE class_name ')'
             instance_attributes ')' ')' ;
class_defs ::= class_def  |  class_def class_defs ;
class_def ::= attribute  |  class  |  instance ;
instance_attributes ::=   |  '(' attributes ')' ;
attributes ::= attribute  |  attribute attributes ;
attribute ::= '(' KEY  data ')'  |  '(' KEY  '(' attribute ')' ')' ;
data ::= alpha_numeric | alpha_numeric data ;
```

図 2: ADF 文法 (主要部)



(A)

```
1  (DEFARCHITECTURE A_Chip (
2      (DEFMODULE ChipDef (
                    :
3      (DEFPIN      Out      ((TYPE OutPut) (VIEW (SHAPE DIA)...)))
4      (DEFPIN      In       ((TYPE InPut) (VIEW (SHAPE DIA)...)))
5      (DEFLINE     LocalL   ((COST 1) (VIEW (...))))
6      (DEFLINE     MidL     ((COST 2) (VIEW (...))))
                    :
7      (DEFMODULE   Lut      ((LINE  i[0..2]  ((TYPE LocalL)))
8                             (LINE  o        ((TYPE LocalL)))
9                             (PIN   i[0..2]  ((TYPE In)))
10                            (PIN   o        ((TYPE Out)))))
11     (MODULE      lut0     ((TYPE Lut) ((VIEW (LOC 100 100)))))
12     (MODULE      lut1     ((TYPE Lut) ((VIEW (LOC 200 100)))))
13     (LINE        l[0..7]  ((TYPE MidL)))
14     (SWITCH      s0       ((LINES lut0.o lut1.i0)))
15     (SWITCH      s1       ((LINES lut0.o l0)))
                    :
16     ))
17     (MODULE    a_chip    ((TYPE ChipDef)))
18 ))
```

(B)

図 3: ADF による記述例

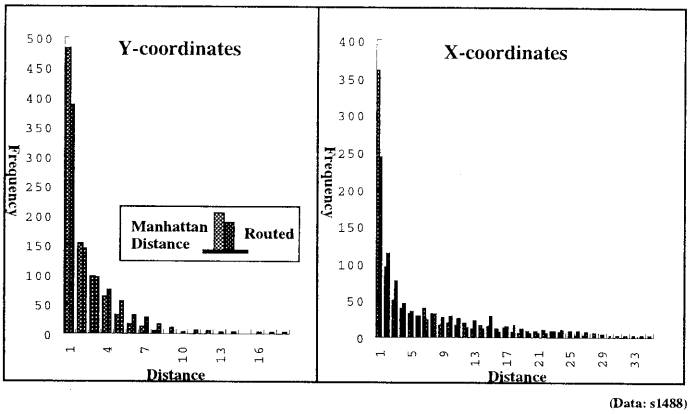


図 4: ポスト・プロセッサ出力例 1

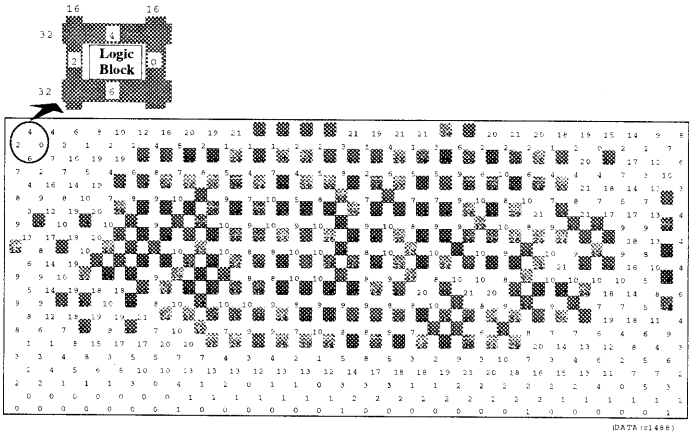


図 5: ポスト・プロセッサ出力例 2 (分散図)

5 適用例

本章では、FACT システムの適用例を 2 つ示す。最初の例は、配線要素の評価である。まず、図 6 に示す 2 つのアーキテクチャを考える。各アーキテクチャは symmetrical array 構造をしている。すなわち、図示した論理ブロック、配線チャネルおよびスイッチボックスを 1 単位として、それが二次元平面に配列された構造を考える。本例では 16 行 32 列の大きさを仮定した。さらに、2 つのアーキテクチャで相対するスイッチボックス内のスイッチ配置パタンは同一とし、唯一の違いは縦方向のチャネル幅である。アーキテクチャ A1 では 8 であり、A2 では 16 である。他の条件は変更していない。この 2 つのアーキテクチャに対して論理回路の配線を試みた。ここで用いた回路は、事前に文献 [9] で述べた手法で、テクノロジマッピング、配置処理をしている。

結果を表 1 に示す。ここで、"#Nets" は配線すべきネット数であり、"Ex. Channel" は、配線処理で全てのトラックが消費されたチャネルの数を示している。値は、縦方向 (vertical) と横方向 (horizontal) で分けて示した。"Unrouted" は、配線できなかったネット数を示している。アーキテクチャ A1 に比べ、A2 では未配線のネット数、"Ex. Channel" 数とも減っている。これより、A2 は配線アーキテクチャの面で A1 より優れていることがわかる。

第二の適用例は配線チャネルとスイッチパタンの評価である。まず、図 7 に示す 3 つのアーキテクチャ・タイプを考える。全てのアーキテクチャは symmet-

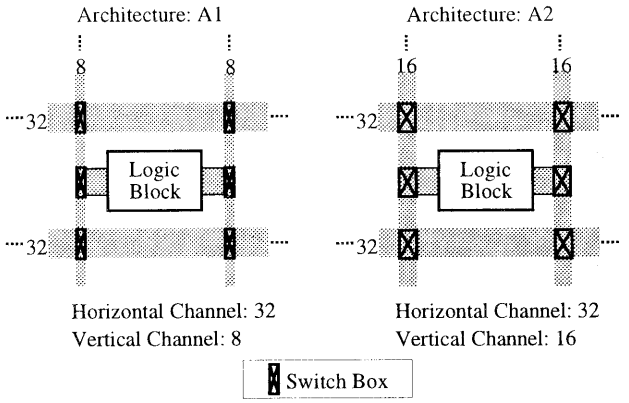


図 6: 適用例 1 (配線要素の評価)

表 1: 適用例 1 の結果

Circuit	#Nets	Architecture: A1			Architecture: A2		
		Unrouted	Ex. Channel		Unrouted	Ex. Channel	
			vertical	horizontal		vertical	horizontal
hpa_t_core1	1189	39	24	1	9	0	0
rst_t_core1	1162	29	26	0	8	0	0
sa_t_core1	1182	20	19	0	14	0	0
random_core1	918	98	76	0	2	1	0

rical array 構造であり、10 行 28 列を仮定している。ただし、各々の論理ブロックの入出力の方向が互いに異なっている。(A) では、入力はは左と上から入り、出力は右と上に出ている。(B) では、入出力とも上のチャネルに接続され、(C) では、(B) に加え下のチャネルにも接続されている。

図 7 のアーキテクチャの各々に対して縦横のトラック数とスイッチボックス内のスイッチ配列パタンを変更しながら、6 つの回路データを使用して評価した。

結果を表 2 に示す。表中、"#V" および "#H" は、各々縦チャネル、横チャネルのトラック数を表す。

"#SW" は、図 7 に示した 1 繰り返し単位ごとに管理されているスイッチボックス内の総スイッチ数を表す。"#LB(%)" は、対応する回路を実現するのに消費した論理ブロックの割合を示している。"Input" および "Output" 欄の "U", "L", "U&R" などの記述は、論理ブロックの入出力の方向を示している。アーキテクチャ "v6" - "v12", "v13" - "v18", "v19" - "v23" は、それぞれ図 7 に示したアーキテクチャ・タイプ (A), (B), (C) に属している。さらに、欄 "hpa_t" から "rand" に示された値は未配線のネット数を示している。

アーキテクチャ "v12" では、全回路について 100% 配線を実現できている。"v12" は、各スイッチボックスを完全スイッチにしている。すなわち、スイッチボックスに接続されている任意の 2 つのトラックが接続可能な構成となっている。そのため、スイッチボックスの影響を除外して、チャネル幅のみに着目できる。"v12" は、与えられた回路を配線するのに十分なチャネル幅を持っていると判断できる。しかし、本アーキテクチャを実現するには、多量のスイッチを必要とし、実現が難しいこともわかる。

"v13" から "v18" は、図 7(B) のアーキテクチャ・タイプであり、各論理ブロックの入出力が上方の横チャネルに接続されている。そのため、スタンダードセルタイプの ASIC のような行ベース (row-based) の配線要素を持っている FPGA とみなすことができる。表 2 に示すように配線結果はあまり良くない。しかし、評価に用いた迷路法を基本としたルータを、チャネルルータで用いられる手法に変更すれば、改善の余地は残されている。

"v19" から "v23" は、アーキテクチャ・タイプ (C)

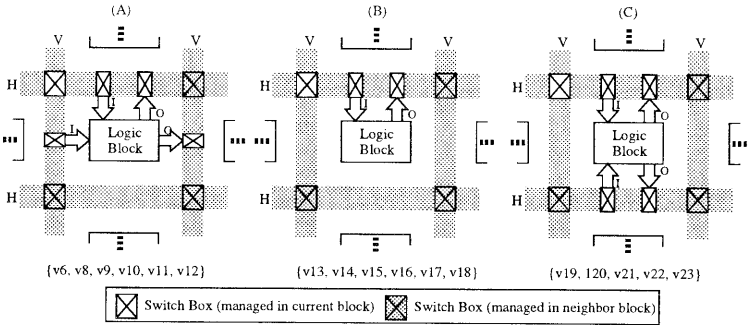


図 7: 評価例 2 (配線チャネルとスイッチパタンの評価)

に属し、アーキテクチャ・タイプ(B)を改善したものとなっている。配線結果は、他の2つのタイプ(A),(B)に比べ良好であり、しかも、スイッチ数も少なく抑えられている。

6 おわりに

FPGA とその専用 CAD の同時評価システムを述べた。本システムは、評価環境を提供するだけでなく、CAD ツールを容易に構築できる環境も提供している。本システムを使用することにより、ユーザは、バランスのとれた FPGA と CAD システムを構築することができる。本システムの適用例を幾つか示し、FPGA アーキテクチャを定量的に評価できることを示した。

我々は、新たな FPGA 設計を本システムを用いて行い、専用 CAD ツールを FPGA 製造の前に完成させた。

謝辞

本システムを構築するにあたり、有意義な議論および援助を頂いた NTT システムエレクトロニクス研究所の高原厚氏、室岡孝宏氏、深見健之助氏に深謝いたします。また、日頃、議論頂く NTT 光ネットワークシステム研究所 伝達プロセッシング研究グループの皆様に感謝いたします。

参考文献

[1] Tsutsui A., Miyazaki T., Yamada K., and Ohta N., " Special Purpose FPGA for High-speed Digital Telecommunication Systems," Proc. ICCD'95, pp. 486-491, October 1995.

[2] Amerson R., Carter R.J., Culbertson W.B., Kuekes P., and Snider G., "Teramac — Configurable Custom Computing," Proc. FCCM'95, pp. 32-38, Napa CA, April 1995.

[3] Brown S.D., Francis R.J., Rose J. and Vranesic Z.G., "Field-ProgrammableGate Arrays," Kluwer Academic Publishers, 1992.

[4] Takashima Y., Takahashi A., and Kajitani Y., "Detailed-Routability of FPGAs with Extremal Switch-Block Structures," Proc. ED&TC'96, Paris, March 1996.

[5] Albaharna O.T., Cheung P.Y.K., and Clarke T.J., "Area & Time Limitations of FPGA-based Virtual Hardware," Proc. ICCD'94, pp. 184-189, October 1994.

[6] Xilinx Inc., "XACT Reference Guide, Volume III," 1994.

[7] Zuken Inc., "CR-5000 Brochure," 1994.

[8] Xilinx Inc., "The Field Programmable Gate Array Data Book," 1994.

[9] Tsutsui A., and Miyazaki T., "An Efficient Design Environment and Algorithms for Transport Processing FPGA," Proc. VLSI'95, pp. 791-798, August 30-September 1 1995.

表 2: 適用例 2 の結果

Architecture							Circuit Data							
Type	Name	#SW	#V	#H	Input	Output	Data	hpa_t	mst_t	rst_t	sa_gs	sa_t	rand	Total
-	-	-	-	-	-	-	#Nets	1189	918	1162	160	1182	918	5529
-	-	-	-	-	-	-	#LB(%)	67.5	71.1	78.6	15.4	69.6	71.1	-
	v6	410	20	20	L&U	U&R	-	42	0	12	0	39	18	111
	v8	390	20	20	L&U	U&R	-	21	0	0	0	19	15	55
(A)	v9	390	20	20	L&U	U&R	-	24	0	0	0	20	15	59
	v10	374	20	20	L&U	U&R	-	16	0	3	0	17	11	47
	v11	374	20	20	L&U	U&R	-	14	0	4	0	14	16	48
	v12	1170	20	20	L&U	U&R	-	0	0	0	0	0	0	0
	v13	384	8	32	U	U	-	39	1	29	0	20	98	187
	v14	384	16	32	U	U	-	9	0	8	0	14	2	33
(B)	v15	797	16	32	U	U	-	3	0	3	0	11	0	17
	v16	406	16	32	U	U	-	13	0	3	0	11	5	32
	v17	399	12	30	U	U	-	25	0	14	0	17	14	70
	v18	609	12	30	U	U	-	20	0	10	0	8	24	62
	v19	399	12	30	U&L	U&L	-	18	1	15	0	17	9	60
	v20	399	12	30	U&L	U&L	-	20	0	14	0	12	5	51
(C)	v21	405	10	30	U&L	U&L	-	43	4	26	0	34	41	148
	v22	406	12	32	U&L	U&L	-	8	0	2	0	3	0	13
	v23	410	16	32	U&L	U&L	-	1	0	1	0	5	0	7