ISSN 2186-5647





# ケアビット分布制御ドントケア抽出法

日本生産工(院)　○山崎 紘史　日大生産工　細川利典
九大　吉村正義

## 1. はじめに

近年，半導体集積技術の進歩に伴い超大規模集積回路(Very Large Scale Integrated circuits : VLSI)が大規模化しており，スキャン設計[1]された VLSI のテスト時消費電力の増大が，現在の回路設計において大きな問題となっている．

テスト時消費電力の増大が引き起こす問題として，発熱と IR ドロップ[2]の 2 つの問題が考えられる．テスト時の過度の電力消費が，VLSI の発熱の原因につながり，回路に致命傷なダメージを与え，場合によっては回路を破損させるおそれがある[3]．また，過度の電力消費が IR ドロップを引き起す可能性が高まる[4]．IR ドロップは遅延を増大させ，正常回路を不良と判定する誤テストを引き起こし，歩留り低下の原因となる[2][5]．したがって，歩留り損失を抑制するため，テスト時消費電力を削減することが重要な課題になっている．テスト時消費電力を削減するための様々な手法が提案されている．電力管理技術の利用，設計変更およびテストデータ変更などの手法が挙げられる[4][6-11]．特に，キャプチャ時のフリップフロップ(Flip-Flop : FF)遷移回数を削減する手法としては LCP(Low capture power) X-Filling 手法[6-8]や，LCP ATPG (Automatic Test Pattern Generation)手法[4][10][11]などが挙げられる．

一般に生成されたテストパターンの入力値は全て 0,1 に設定される．しかしながら，生成されたテストパターンの中には，逆の論理値に変更しても故障検出率が低下しない入力値が存在し，それらの論理値を抽出するドントケア抽出技術[12]が提案されている．LCP X-Filling 手法ではドントケアを含むテストパターン集合に対し，ドントケアに 0, 1 の値を再割当てすることにより，FF 遷移回数を削減し，低消費電力向けのテストパターン集合を生成する．そのため，ドントケア抽出技術の効率化が重要となる．

しかしながら，文献[12]で提案されたドントケア抽出技術では特定のテストパターンにドントケアビットが偏る傾向があり，LCP X-Filling による FF 遷移回数削減の効果が小さい場合がある[13]．特定のテストパターンにドントケアビットが偏る問題を解決する手法として，各テストパターンで検出する故障数を均一化する手法[13]が提案されている．この手法では，各テストパターンで検出する故障数を均一化することにより，低消費電力化に対し効果が高いテストパターンを生成できることが報告されている．しかしながら，この手法ではテストパターンに対してはドントケアビットの均一化を考慮しているが，(擬似)外部入力に対してはケアビットの均一化を考慮していない．

本論文では，入力されたテストパターン集合に対しテストパターン，及び(擬似)外部入力に対してケアビットの分散を制御するドントケア抽出法を提案し，このテストパターン集合に対して LCP X-Filling を行った結果を考察する．

## 2. ケアビット分布制御ドントケア抽出

本章では，ケアビット分布制御のためのドントケア抽出法について説明する．従来手法[12][13]では特定のテストパターンにドントケアビットが偏る問題や，(疑似)外部入力に対してはケアビットの均一化を考慮していない．そのため特定の(疑似)外部入力にケアビットが偏り，スキャンテストにおけるキャプチャ消費電力の削減やチップ温度均一化などの適用分野に効果的でない可能性がある．本手法ではテストパターン，(疑似)外部入力に関してケアビット分布を制御するドントケア抽出法を提案する．

### 2．1．ドントケア抽出アルゴリズム

基本的なドントケア抽出アルゴリズム[12]について説明する．図 2 にアルゴリズムを示す．まず，Step1 で各テストパターン $t_i$ に対して，故障シミュレーションを行う．次に Step2 で，テストパターン $t_i$ に対し，$t_i$ の必須故障[14]集合を求める．Step3 で，それらの必須故障を検出するように $t_i$ での故障シミュレーション結果を基に，外部入力の値を計算し，$t_i$'のケアビットを決定する．t'は必須故障以外の故障を検出する可能性があるので，Step4 で $t_i$'に対する故障シミュレーションを行う．Step2 から Step4 の処理により，初期のテストパターン集合 T'が得られる．T'は必須故障を全て検出するテストパターン集合である．

T'では未検出故障があるので，全ての故障が検出されるように，$t_i$'のドントケアのいくつかを元のテストパターン $t_i$ の値に戻す．Step5 で，$t_i$'では未検出であるが $t_i$ では検出できる故障を算出する．そして Step6 で算出された故障を検出するための論理値を計算し，$t_i$'を最終的に決定する．Step7 で $t_i$'に対し故障シミュレーションを行い，故障リストを更新する．

```
Procedure X-search(C,T)
  Circuit C;  Test_set T;
{
    for each testpattern t_i in T{
        fault_simulation(t_i);                  Step1
    }
    for each testpattern t_i in T{
        F=collect_essential_fault(t_i);         Step2
        t_i'=find_value(F);                     Step3
        fault_simulation(t_i');                 Step4
    }
    for each testpattern t_i in T{
        G=collect_undetected_fault(t_i);        Step5
        t_i'=find_value(G);                     Step6
        fault_simulation(t_i');                 Step7
    }
    rerutn T' composed of t_i';
}
```

図 2．従来手法全体アルゴリズム

---

**A don't care identification method with care bit distribution control**

Hiroshi YAMAZAKI, Toshinori HOSOKAWA, and Masayoshi YOSHIMURA

## 2．2．提案手法全体アルゴリズム

本手法では，ケアビット分布を制御したドントケア抽出を行う．図 3 に提案手法の処理手順を記す．Step1 から Step5，Step7 は 2.1 と同様の処理を行い，Step6 で提案手法であるケアビット分布を制御したドントケア抽出を行う．本節では Step6 に対してのみ説明を行う．

(Step6)
Step5 で算出された故障に対して，ケアビット分布を考慮したドントケア抽出を行う．詳細は 3.4 節で説明する．

## 2．3．評価関数

本節では，本手法でケアビット分布制御に用いる評価関数について説明する．式(1)に各テストパターンに対する評価関数，式(2)に各(疑似)外部入力に対する評価関数を示す．そして式(3)に故障 $f_i$ をどのテストパターンで検出するかを選択するための評価関数を示す．

$$W(t_i)=\sum_{j=1}^{N} b(t_i,p_j) \quad \cdots\cdots (1)$$

$$W(p_j)=\sum_{i=1}^{M} b(t_i,p_j) \quad \cdots\cdots (2)$$

式(1),(2),(3)において，(疑似)外部入力数は N，テストパターン数は M である．また $b(t_i,p_j)$はテストパターン $t_i$ の(疑似)外部入力 $p_j$ の値がドントケアビットだった場合 0，ケアビットだった場合 1 とする．

表 2 は，図 3 の Step4 の後，ドントケアを含むテストパターン集合 T'に対して各テストパターン，各(疑似)外部入力に含まれるケアビット数を式(1)，(2)を用いて計算した例を表している．ここで x はドントケア，c はケアビット，$t_1$~$t_5$ はテストパターン，$p_1$~$p_7$ は(疑似)外部入力を表す．

```
Procedure X-search(C,T)
  Circuit C;  Test_set T;
{
    for each testpattern t_i in T{
        fault_simulation(t_i);                    Step1
    }
    for each testpattern t_i in T{
        F=collect_essential_fault(t_i);           Step2
        t_i'=find_value(F);                       Step3
        fault_simulation(t_i');                   Step4
    }
    for each testpattern t_i in T{
        G=collect_undetected_fault(t_i);          Step5
        t_i'=carebit_distribution_find_value(G);  Step6
        fault_simulation(t_i');                   Step7
    }
    rerutn T' composed of t_i';
}
```

**図 3．提案手法全体アルゴリズム**

**表 2．各テストパターン中のケアビットと $W(t_i),W(p_j)$**

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | $W(t_i)$ |
|---|---|---|---|---|---|---|---|---|
| $t_1$ | x | x | c | c | x | c | x | 3 |
| $t_2$ | c | x | x | c | c | x | c | 4 |
| $t_3$ | x | x | c | c | x | x | x | 2 |
| $t_4$ | c | x | c | c | c | x | x | 4 |
| $t_5$ | x | x | c | c | x | c | x | 3 |
| $W(p_j)$ | 2 | 0 | 4 | 5 | 2 | 2 | 1 | |

**表 3．各ビットに対するコスト $W(t_i,p_j)$**

| | $W(t_i)$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ |
|---|---|---|---|---|---|---|---|---|
| $W(p_j)$ | | 2 | 0 | 4 | 5 | 2 | 2 | 1 |
| $t_1$ | 3 | 5 | 3 | 0 | 0 | 5 | 0 | 4 |
| $t_2$ | 4 | 0 | 4 | 8 | 0 | 0 | 6 | 0 |
| $t_3$ | 2 | 4 | 2 | 0 | 0 | 4 | 4 | 3 |
| $t_4$ | 4 | 0 | 4 | 0 | 0 | 0 | 6 | 5 |
| $t_5$ | 3 | 5 | 3 | 0 | 0 | 5 | 0 | 4 |

次に各ビットに対する評価関数について説明する．各ビットに対するコストは式(1)，(2)で定義した $W(t_i),W(p_j)$を用いて以下の式(3)を用いて計算する．ここで $W(t_i, p_j)$はテストパターン $t_i$ の(疑似)外部入力 $p_j$ の評価値を表す．

$$W(t_i,p_j)=W(t_i)+W(p_j) \quad \cdots\cdots (3)$$

表 3 は各ビットに対するコストを表す．例えば $W(t_1,p_1)$のコストを計算する場合 $w(t_1)$=3，$W(p_1)$=2 なので $W(t_1,p_1)$のコストは式(3)より 5 となる．ただし，$t_1$ の $p_3$ の値がケアビットである場合は $W(t_1,p_3)$=0 とする．この計算を全てのビットに対して行う．

各ビットに対するコストは，対象とするビットをケアビットにした場合どれだけケアビット分散が小さくなるかを示している．例えば $W(t_2,p_3)$のコストは 8 であり，一番コストが高く設定されている．このビットをケアビットにすると，分散が大きくなる．

## 2．4．故障とテストパターンのマッチング

3.2 節では各ビットに対するコストの計算について説明した，本節では 3.2 節で算出したコストを用いて，故障を検出するテストパターンの選択方法について説明する．必須故障ではない故障 f に対しドントケア抽出を行う場合，故障 f を検出するために必要なケアビットを，故障 f を検出する全てのパターンに対して算出する．しかしながら，故障 f は一つのパターンで検出すれば十分であるため，どのテストパターンで検出するかを選択する必要がある．本提案手法ではケアビットの偏りを削減するようなドントケア抽出を行うことが目的である．下記の式(4)で算出したコストを用いて故障 f を検出するテストパターンを選択する．式(4)において N は(疑似)外部入力数である．

$$MW(t_i)=\sum_{j=1}^{N} W(t_i,p_j) \quad \cdots\cdots (4)$$

表 4 は，故障 f はテストパターン $t_1$,$t_3$,$t_5$ で検出可能であり，そのときの各テストパターンで故障 f を検出するために必要なコストを表している．

例として，テストパターン $t_1$ で故障 f を検出するために必要なコストを算出する．テストパターン $t_1$ で故障 f を検出する場合，(疑似)外部入力 $p_1$,$p_2$,$p_3$,$p_6$,$p_7$ をケアビットにする必要がある．ここで表 3 の各ビットに対するコストを基に，式(4)より $t_1$ で故障 f を検出するときのコストを求めると

$$MW(t_1) = W(t_1,p_1)+W(t_1,p_2)+W(t_1,p_3)+W(t_1,p_6)+W(t_1,p_7)$$
$$= 5+3+0+0+4 = 12$$

となり，$t_1$ で故障 f を検出するときにかかるコストは 12 となる．同様に故障 f を $t_3$,$t_5$ で検出するときのコストを求めるとそれぞれ 5, 8 となり，故障 f は $t_3$ で検出するときのコストが最小になるため，故障 f はテストパターン $t_3$ で検出するようにケアビットを決定する．

表 4. 故障 f を検出するために必要なコスト

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | MW($t_i$) |
|---|---|---|---|---|---|---|---|---|
| $t_1$ | c | c | c | x | x | c | c | 12 |
| $t_3$ | x | c | c | c | x | x | c | 5 |
| $t_5$ | c | c | x | x | x | c | x | 8 |

## 2. 5. ケアビット分布を制御したドントケア抽出アルゴリズム

図 3 の step6 で示したケアビット分布を制御したドントケア抽出について説明する．図 4 にケアビット分布を制御したドントケア抽出のアルゴリズムを記す．ここで C は回路，T はテストパターン集合，F は未検出故障リストである．

(step6-1)
未検出故障リスト F より未検出故障 fi を選択する．

(step6-2)
T 中のすべてのテストパターンで fi を故障シミュレーション実行する．fi を検出する T 中のテストパターン集合を DT とする．

(step6-3)
テストパターン集合 DT の各テストパターンに対し，故障 fi を検出するために必要なケアビット $C_{DF,fi}$ を算出する．

(step6-4)
テストパターン集合 DT の各テストパターンについて，故障 fi を検出するために必要なケアビット $C_{DF,fi}$ に対する，コスト V を計算する．

(step6-5)
Step6-4 よりコスト V が最小となるテストパターン ti を選択し，ケアビットを決定する．

(step6-6)
ti に対し故障シミュレーションを行い，未検出故障リスト F を更新する．

## 3. 実験結果

本章では提案手法の性能を評価するために，提案手法を実装し，ドントケア分散とドントケア抽出率ついて評価する．またドントケア抽出後のテストパターン集合に対し LCP X-Filling を実行し，キャプチャセーフ判定結果を評価した．対象回路は ITC'99 ベンチマーク回路の b14,b15,b17,b20,b21,b22 である．初期テストパターンは Synopsys 社の TetraMAX(打ち切り制限 1000) によって生成された遷移故障用のテストパターン集合を用いた．図 5 に実験の全体フローを示す．

```
Procedure distribution Xidentification(C, T,  F)
Circuit C, First Test Set T, No Detect Fault List F;
{
    for each fault in F{
        fi=collect_no_detect_fault(F);                Step1
        DT=collect_detect_test_pattern(T, fi);        Step2
        C_DT,fi=find_calue(DT, fi);                   Step3
        V=caluclation_cost(C_DT,fi);                  Step4
        ti=sellection_low_cost_pattern(V);            Step5
        F=fault_simulation(F, ti);                    Step6
    }
}
```

図 4．ケアビット分布を制御したドントケア抽出

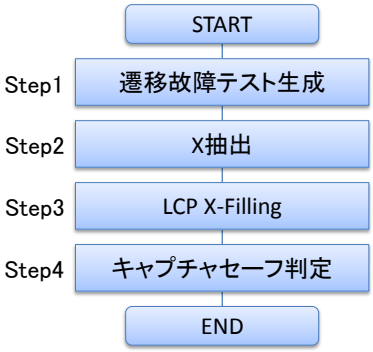


図 5．実験フロー

(step1)
TetraMAX により遷移故障用のテストパターン集合を生成する．なおテスト生成の打ち切り制限はバックトラックリミット 1000 と設定した．

(step2)
step1 で生成した初期テストパターン集合に対して，提案手法と従来法によるドントケア抽出を行い，ドントケア分散を求める．

(step3)
step2 でドントケア抽出を行ったそれぞれのテストパターン集合に対して，LCP X-Filling によるドントケア割当てを行う．

(step4)
step3 で LCP X-Filling を行ったそれぞれのテストパターン集合に対して，キャプチャセーフ判定を行う．キャプチャセーフ判定基準は回路中の FF 遷移回数が全 FF 数の 5,10,15%を上限とし，それを超えるものをアンセーフテストパターンとした．

表5は提案手法と従来手法における疑似外部入力の分散と，テストパターンの分散，ドントケア抽出率を比較したものを表している．PPI分散は疑似外部入力のドントケア分散，TP分散はテストパターン集合のドントケア分散を表している．PPI分散に関しては提案手法のほうが従来手法より，全ての回路において分散値を小さくすることができた．またドントケア抽出率に関しても，ほとんどの回路において提案手法のほうが高いことが分かる．しかしながら，TP分散に関しては，一部回路によっては提案手法のほうが増大していることがわかる．

表6は提案手法と従来手法におけるドントケア抽出後テストパターン集合に対して，LCP X-Fillingを行いキャプチャセーフ判定した結果を表している．ほぼ全ての回路において，提案手法がアンセーフテストパターン数と，アンセーフ故障数を削減することができた．

## 4. おわりに

本論文では，ケアビットの分散を制御するドントケア抽出法を提案し，評価実験を行った．従来のドントケア抽出法と比較して，疑似外部入力に対する分散値は小さくすることができたが，一部回路においてテストパターンに対する分散値は逆に増大した．

キャプチャセーフ判定に関しては，従来手法よりもアンセーフ故障数の削減することができた．今後の課題として，テストパターンに対してドントケア分散値を小さくするようなコストの提案や，キャプチャ消費電力削減のためのドントケア抽出の提案が挙げられる．

## 文 献

[1] 古川寛，"JTAG を使った LSI テスト回路の組み込み手法"，Design Wave Magazine，2000，pp30-31.
[2]Takaki Yoshida, Masahmi Watati, "A New Approach for Low Power Scan Testing", International, Test Conference, 2003,

pp480-487
[3]Jaehoon Song, Hyunbean Yi, Doochan Hwang, Sungju Park "A Compression Improvement Technique for Low-Power Scan Test Data" IEEE Region 10 Conference, 2006, pp12-13
[4] X. Wen, K. Miyase, S. Kajihara, H. Furukawa, Y. Yamato, A. Takashima, K. Noda, H. Ito, K. Hatayama, T. Aikyo, and K. K. Saluja, "A Capture-Safe Test Generation Scheme for At-Speed Scan Testing" IEEE, 2008, pp55-60
[5] Fukuzawa Tomoaki, Miyase Kohei, Yamato Yuta, Furukawa Hiroshi, Wen Xiaoqing, Kajihara Seiji. "A Transition Delay Test Generation Method for Capture Power Reduction during At-Speed Scan Testing" IEICE technical report. Dependable computing 107(337), 2007, pp7-12
[6] Santiago Remersaro, Xijiang Lin, Zhuo Zhang, Sudhakar M. Reddy,Irith Pomeranz and Janusz Rajsk, "Preferred Fill: A Scalable Method to Reduce Capture Power for Scan Based Designs" Test Conference, 2006. ITC'06. IEEE International, pp32.2, Oct. 2006.
[7] Chao-Wen Tzeng Shi-Yu Huang, "QC-Fill: An X-Fill Method for Quick-and-Cool Scan Test" Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09. pp1756-1766 , April 2009.
[8] Jia LI, Qiang XU, Yu HU, and Xiaowei LI, "iFill: An Impact-Oriented X-Filling Method for Shift- and Capture-Power Reduction in At-Speed Scan-Based Testing", Design, Automation and Test in Europe, 2008. DATE '08, pp1184-1189 , March 2008.
[9] Xiaoqing WEN, Yoshiyuki YAMASHITA, Seiji KAJIHARA, Laung-Terng WANG, Kewal K. SALUJA, "A New Method for Low-Capture-Power Test Generation for Scan Testing", IEICE TRANS. INE & SYST. VOLE89-D, NO.5, pp1679-1686, 2006.
[10] V.R. Devanathan, C.P. Ravikumar, V. Kamakoti, "Glitch-Aware Pattern Generation and Optimizatio Framework for Power-Safe Scan Test", 25th IEEE VLSI Test Symmposium, pp167-172, 2007.
[11] Xiaoqing WEN, Seiji KAJIHARA, "A Novel ATPG Method for Capture Power Reduction during Scan Testing" IEICE TRANS. INE&SYST, VOLoE90-D, NO.9 SEPTENIBER, pp1398-1405, 2007.
[12] K. Miyase, S. Kajihara ,"XID: Don't Care Identification of Test Patterns for Combinational Circuits," IEEE Trans. Comuter-Aided Design of Integrated Circuits and Systems,Vol. 23, No. 2, pp. 321-326,Fed. 2004.
[13] Kohei Miyase, Kenji Noda, Hideaki Ito, Kazumi Hatayama,Takashi Aikyo, Yuta Yamato, Hiroshi Furukawa, Xiaoqing Wen,Seiji Kajihara "Effective IR-Drop Reduction in At-Speed ScanTesting Using Distribution-Controlling X-Identification" IEEE/ACM International Conference on Computer-AidedDesign pp52-58.2008
[14] Seiji Kajihara, Irith Pomeranz, Kozo Kinoshita andSudhakar M.Reddy "Cost-Effective Generation of MinimalTest Sets for Stuck-at Faults in Combinational Logic Circuits", 30th ACMIEEE Design Automation Conference,pp102-106,1993.

表 5. 各手法におけるドントケアビットの分散とドントケア抽出率

| 回路名 | 初期テストパターン数 | 疑似外部入力数 | 故障検出率(%) | 従来手法 | | | | | 提案手法 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 最小X数 | 最大X数 | PPI分散 | TP分散 | X抽出率 | 最小X数 | 最大X数 | PPI分散 | TP分散 | X抽出率 |
| b14 | 1163 | 245 | 92.72 | 25 | 243 | 106738 | 5484 | 45.8 | 2 | 243 | 51704 | 4244 | 59.3 |
| b15 | 1020 | 449 | 82.15 | 43 | 441 | 71464 | 12840 | 65.2 | 0 | 442 | 32580 | 21504 | 65.0 |
| b17 | 2153 | 1415 | 80.97 | 126 | 1406 | 191834 | 106261 | 68.4 | 0 | 1407 | 120263 | 244878 | 62.1 |
| b20 | 1383 | 490 | 92.14 | 15 | 471 | 87684 | 21739 | 33.6 | 2 | 474 | 74272 | 18973 | 50.4 |
| b21 | 1480 | 490 | 91.98 | 18 | 478 | 94906 | 22596 | 34.9 | 2 | 478 | 84286 | 19409 | 51.0 |
| b22 | 1509 | 735 | 91.88 | 50 | 719 | 162726 | 36328 | 37.3 | 4 | 721 | 118645 | 40752 | 51.9 |

表 6. 各手法におけるキャプチャセーフ判定結果

| 回路名 | FF数 | FF遷移回数上限(%) | 従来手法 | | | | 提案手法 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | セーフTP数 | アンセーフTP数 | セーフ故障数 | アンセーフ故障数 | セーフTP数 | アンセーフTP数 | セーフ故障数 | アンセーフ故障数 |
| b14 | 245 | 15 | 1121 | 25 | 39695 | 344 | 1149 | 13 | 39892 | 213 |
| b14 | 245 | 10 | 1086 | 60 | 39059 | 983 | 1114 | 48 | 39425 | 682 |
| b14 | 245 | 5 | 456 | 690 | 23466 | 16575 | 745 | 417 | 32277 | 7829 |
| b15 | 449 | 15 | 1000 | 4 | 33962 | 33 | 1011 | 8 | 33989 | 91 |
| b15 | 499 | 10 | 954 | 50 | 33435 | 560 | 979 | 40 | 33412 | 668 |
| b15 | 499 | 5 | 702 | 302 | 29825 | 4170 | 802 | 217 | 30564 | 3516 |
| b17 | 1415 | 15 | 2125 | 0 | 117622 | 0 | 2125 | 0 | 117920 | 0 |
| b17 | 1415 | 10 | 2094 | 31 | 116537 | 1085 | 2126 | 25 | 117326 | 594 |
| b17 | 1415 | 5 | 1572 | 553 | 103420 | 14201 | 1710 | 441 | 106573 | 11347 |
| b20 | 490 | 15 | 1315 | 64 | 79389 | 420 | 1362 | 20 | 79731 | 190 |
| b20 | 490 | 10 | 1073 | 306 | 77111 | 2698 | 1179 | 203 | 78078 | 1843 |
| b20 | 490 | 5 | 215 | 1164 | 34196 | 45613 | 371 | 1011 | 49012 | 30909 |
| b21 | 490 | 15 | 1417 | 47 | 80686 | 325 | 1457 | 22 | 80917 | 247 |
| b21 | 490 | 10 | 1199 | 265 | 78385 | 2626 | 1337 | 142 | 79507 | 1657 |
| b21 | 490 | 5 | 359 | 1105 | 35911 | 45100 | 602 | 877 | 50613 | 30552 |
| b22 | 735 | 15 | 1451 | 48 | 117891 | 323 | 1490 | 15 | 128788 | 154 |
| b22 | 735 | 10 | 1076 | 423 | 112745 | 5469 | 1282 | 223 | 128788 | 2987 |
| b22 | 735 | 5 | 160 | 1339 | 40898 | 77316 | 337 | 1168 | 63385 | 54940 |