# B. TECH. PROJECT REPORT

**On**

# Double Line of Defense for Securing DSP Hardware IP


BY

**Somesh Patil, 160001056**

**Gaurav Naukudkar, 160001040**




**DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY INDORE**


**November 2019**

# Double Line of Defense for Securing DSP Hardware IP

**A PROJECT REPORT**

*Submitted in partial fulfillment of the*
*requirements for the award of the degrees*

***of***
**BACHELOR OF TECHNOLOGY**
**in**

**COMPUTER SCIENCE AND ENGINEERING**

*Submitted by:*
**Somesh Patil, 160001056**
**Gaurav Naukudkar, 160001040**

*Guided by:*
**Dr. Anirban Sengupta**


**INDIAN INSTITUTE OF TECHNOLOGY INDORE**
**November 2019**

## CANDIDATE'S DECLARATION

We hereby declare that the project entitled "**Double Line of Defense for Securing DSP Hardware IP**" submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Anirban Sengupta, Associate Professor, Computer Science and Engineering,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

**Somesh Patil (160001056)** **Gaurav Naukudkar(160001040)**

---

## CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

**Dr. Anirban Sengupta**

**Associate Professor**

**Computer Science and Engineering**

**IIT Indore**

# Preface

This report on "**Double Line of Defense for Securing DSP Hardware IP**" is prepared under the guidance of Dr. Anirban Sengupta.

We have tried to present the detailed concept of multiphase key based structural obfuscation and physical level watermarking. Through this report the explanation and all the transformation for structural obfuscation and watermark embedding is shown diagrammatically. As well as the algorithm for making the design secure is present. For better understanding of our concept one complete structurally obfuscated and watermarked design is added with all the steps from base case design i.e. design which is neither obfuscated nor secure. To conclude the report comparison of our obfuscation with previous work as well as comparison of cost of obfuscated and watermarked design with related work is included.

**Somesh Patil (160001056) and Gaurav Naukudkar (160001040)**
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

# Acknowledgements

We wish to thank Dr. Anirban Sengupta for his kind support and valuable guidance.

It is his help and support, due to which we became able to complete the design and technical report.

Without his support this report would not have been possible. It was only possible because of his enthusiasm, beforehand schedule, knowledge and sincerity towards us and our work to produce better results. We wouldn't have achieved our goals without his encouragement at each step.

**Somesh Patil (160001056) and Gaurav Naukudkar (160001040)**
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

# Abstract

Digital signal processing (DSP) cores are an integral part of consumer electronics devices. A DSP circuit is considered to be secure, if its functionality is designed to be hidden from an adversary. In other words to make a DSP design secured, its hardware architecture should not look obvious in terms of its functionality. Structural obfuscation plays a critical role in realizing this objective. Structural obfuscation offers a means to effectively secure through obfuscation the contents of any Digital Signal Processing (DSP) cores used in consumer electronics devices. However, due to increasing globalization of design supply chain, possibility of intervention and typical attacks is on the rise, which therefore mandates protection of Digital Signal Processing (DSP) cores from piracy/counterfeiting. Embedding a strong watermark is sufficient to prove IP core ownership during conflict resolution process. In this work a novel multiphase key based structural obfuscation methodology along with embedding a watermark at the physical level for protecting a Digital Signal Processing (DSP) core. The proposed approach specifically targets protecting Digital Signal Processing (DSP) cores against both the threat models i.e. reverse engineering and IP piracy/counterfeiting attacks. This problem has never been addressed before. The multiple phases involved in key based structural obfuscation are Loop unrolling, Partitioning, Redundant Operation Elimination (ROE), Tree Height Transformation (THT) and Folding which can yield camouflaged functionally equivalent designs, making it extremely difficult for an adversary to reverse engineer it. Followed by embedding a watermark at the physical level which will protect the Digital Signal Processing (DSP) core from piracy/counterfeiting attacks. Results indicate that without incurring any design cost overhead, the proposed approach is able to protect the DSP core against both the threat models.

# Table of Contents

# List of Figures

# List of Tables

# I. Introduction

Hardware security and Intellectual Property (IP) core protection is an emerging area of research for semiconductor community that focusses on protecting designs against standard threats such as reverse engineering, counterfeit, forgery, malicious hardware modification etc. Hardware security is broadly classified into two types: (a) authentication based approaches (b) obfuscation based approaches. Some of the approaches that fall under the first type are digital watermarking, IP metering, physical unclonable functions etc. The second type of hardware security approach i.e. obfuscation can again be further sub-divided into two types: (i) structural obfuscation (ii) functional obfuscation. Structural obfuscation transforms a design into one that is functionally equivalent to the original but is significantly more difficult to reverse engineer (RE), while the second one is active protection type that locks the design through a secret key. Obfuscation may include altering human readability of hardware description language or encrypting the source code. Converting a design into a form that makes it harder for an adversary to discover its functionality is difficult to reverse engineer i.e. when an IP core functionality is designed to be hidden for an adversary, it is difficult to reverse engineer.

Almost every smart CE device is heavily dependent on DSP and multimedia processor (MP) IP cores. Because of the globalization of the design supply chain, these IP cores are not safe from predatory attacks. Thus, the protection of reusable IP cores has become a serious concern for the CE industry in the current era. However, the existing state of the art focuses on protecting IP cores against single threat i.e. either reverse engineering or IP piracy. But our proposed approach has the ability to tackle both the threat models i.e. reverse engineering and IP piracy simultaneously thereby adding a double line of defense against such threat models, without incurring any overhead cost. Our approach combines structural obfuscation with watermarking which provides double line of defense against IP piracy and reverse engineering attacks. The structural obfuscation technique used in our approach is multiphase key based which makes it very difficult to reverse engineer and also adds novelty to our approach.

## II. Prior Work

Consumer electronics literature has several works dealing with various aspects of digital signal processor. They focus on different aspects of consumer electronics system characteristics such as energy efficiency, high performance, and area efficiency. However, they miss the critical axis of Consumer Electronic system (i.e. the "security and IP protection"); thus, do not comprehensively address the critical Consumer Electronic design issues in the current social network driven era in which the cyber-security is a critical design axis.

Logic obfuscation uses additional XOR/XNOR gates in circuits to protect the IP core [7] [8] [9]. However, this incurs overhead in design due to insertion of additional logic components/circuitry. Further, to effectively implement this approach, determining correct location of key gates is essential. Further, in [10], [11] structural obfuscation is executed on DSP cores. Further, [11] has not handled loop-based CDFG applications for DSP. Therefore, no loop-based HLT techniques are applicable to obfuscate the design. Moreover, no equivalent DSP circuit of obfuscated design is generated during synthesis. These approaches also do not perform multiple-phase obfuscations using Loop Unrolling, Partitioning, Redundant Operation Elimination, Tree Height Transformation and Folding. Also, none of the above approaches perform key-based Structural Obfuscation.

Embedding watermarking at the physical level for IP protection has been tackled only in few works so far. For example, in [5] and [6], authors use only a combination of 0 and 1 to encode their signature in the form of adding additional edges in the colored interval graph during High Level Synthesis. However, in such cases the signature is susceptible to attacks/compromise, if encoding rule of both the variable is known somehow. Moreover [5] and [6] are not capable to produce watermark with low embedding cost or less storage overhead. However, no approach exists in literature that tackles both the attacks on IP cores i.e. Reverse Engineering and IP piracy. Moreover, the proposed approach handles both the attacks and does not incur any design overhead.

# III. Proposed Methodology

## III.1 Overview

For protecting Digital Signal Processing Hardware IP cores we have combined multiphase key based structural obfuscation with watermarking at the physical level which ensures double line of defense against reverse engineering and piracy/counterfeiting attacks. The multiple high-level transformations involved in key based structural obfuscation are Loop Unrolling, Partitioning, Redundant Operation Elimination (ROE), Tree Height Transformation (THT) and Folding. Loop Unrolling is a high-level transformation applicable to only loop-based Control Data Flow Graphs (CDFGs) which is applied to obfuscate it by unwinding the loop. Partitioning is a high-level transformation technique which is applied to obfuscate the loop unrolled CDFG by dividing it into a number of symmetric partitions, such that every partition must contain at least two nodes having a dependency between them. Redundant Operation Elimination (ROE) is a high-level transformation technique which is applied to obfuscate the input CDFG by removing redundant nodes from the graph. Tree Height Transformation (THT) is a high-level transformation technique which is responsible for obfuscating the CDFG by increasing or decreasing the height of the graph. Folding is a high-level transformation technique in which a common resource is shared among a set of nodes that have the same operation type but are present in successive control steps. Then we create a floorplan which denotes the orientation of the resources on the chip. Watermark consists of three variables α, β and γ which is then finally embedded into the floorplan. Alpha (α) component of the signature is embedded by placing an odd functional unit module above an even functional unit module of the same type. Beta (β) component of the signature is embedded by placing an odd multiplexer unit above an even multiplexer unit of the same type. Gamma (γ) component of the signature is embedded by placing an odd demultiplexer unit to the right of an even demultiplexer unit of the same type.

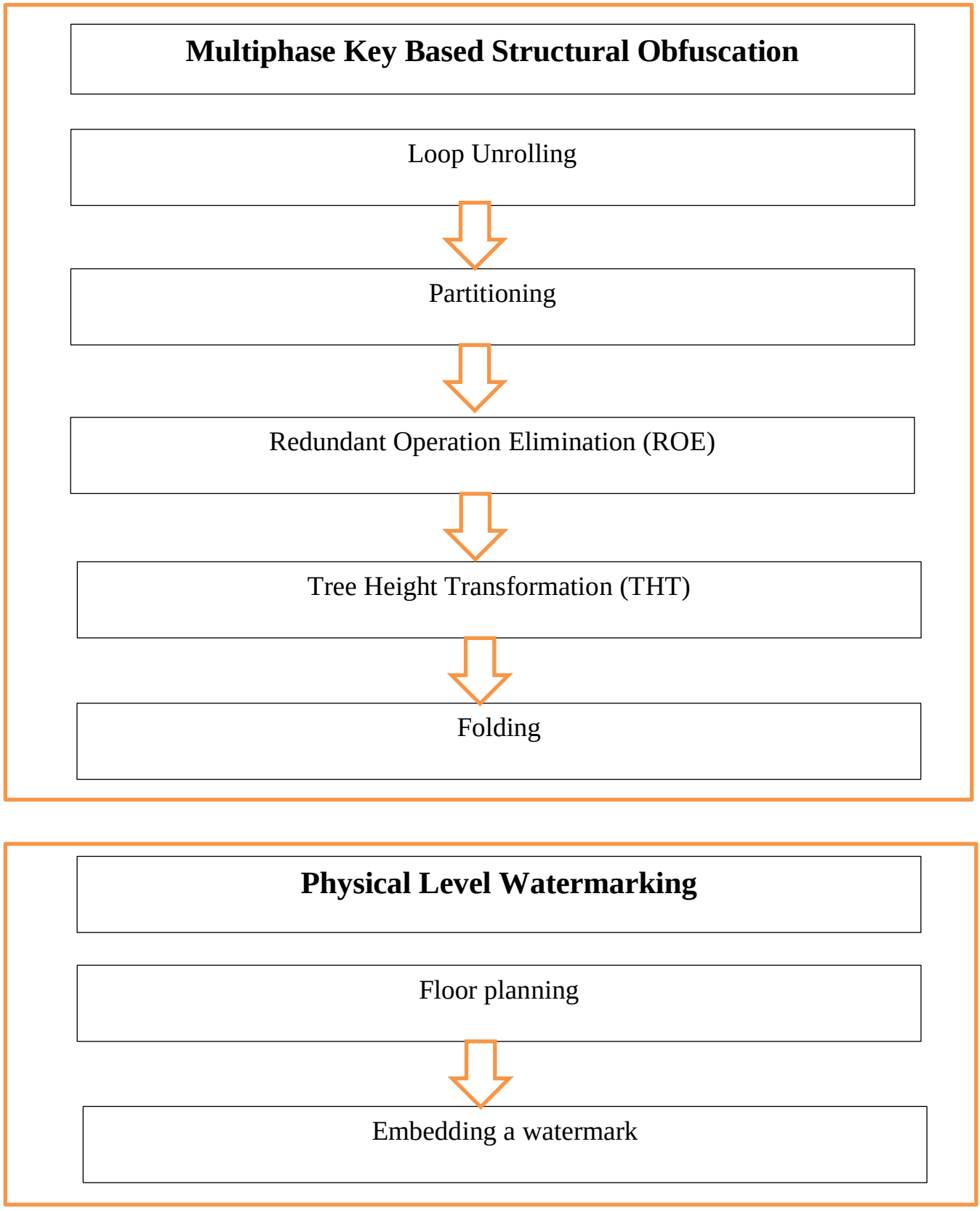


Fig. III.1 Overview of Proposed Algorithm

## III.2 Details of Proposed Algorithm

### III.2.1 Multiphase Key Based Structural Obfuscation

### Loop Unrolling

Loop transformation based high level transformation technique which is applied to obfuscate the input CDFG by unwinding the loop is loop-unrolling. Unrolling of Loops can be achieved by repeating the same loop body in multiple sequences to improve execution delay. Finally, loop unroll based structurally obfuscated graph is produced as output. In the proposed approach the unrolling factor is provided as an input by the user. Loop unrolling minimize the execution time simultaneously obfuscate the design. For example, the figure below unrolls the input loop based CDFG on the left 8 times resulting in loop unrolled CDFG on the right.

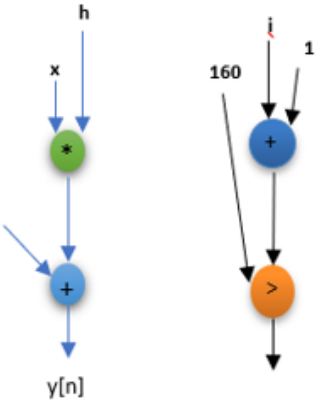


Fig. III.2 FIR filter DFG.

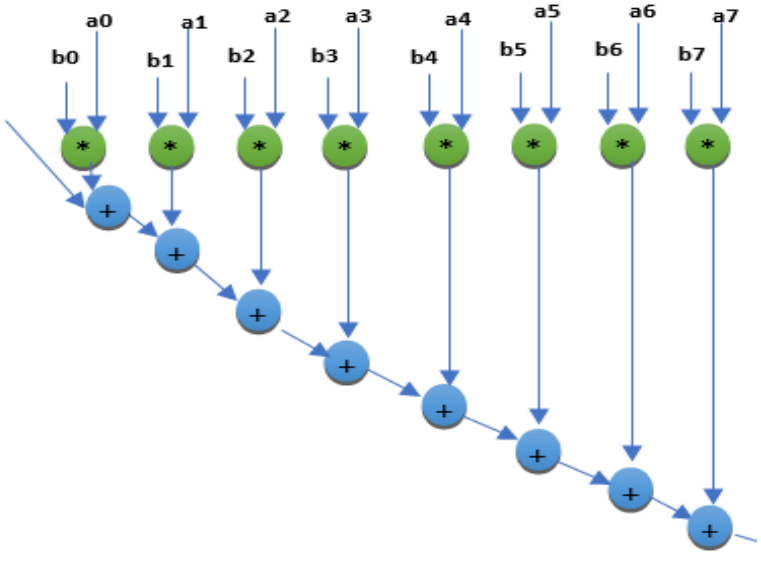


Fig. III.3 Loop Unrolled DFG with unrolling factor 8.

## Partitioning

It is a high-level transformation technique which is applied to obfuscate the loop unrolled CDFG by dividing it into a number of symmetric partitions, such that every partition must contain at least two nodes having a dependency between them. Finally, partition based structurally obfuscated graph is produced as output. Partitioning plays a significant role in obfuscation because finally after applying all the high-level transformations when the partitions are recombined the number of multiplexers and demultiplexers in the final RTL increases thereby obfuscating the IP. In the proposed approach the number of partitions is provided as an input by the user. For example, the figure below shows the CDFG divided into 4 partitions.

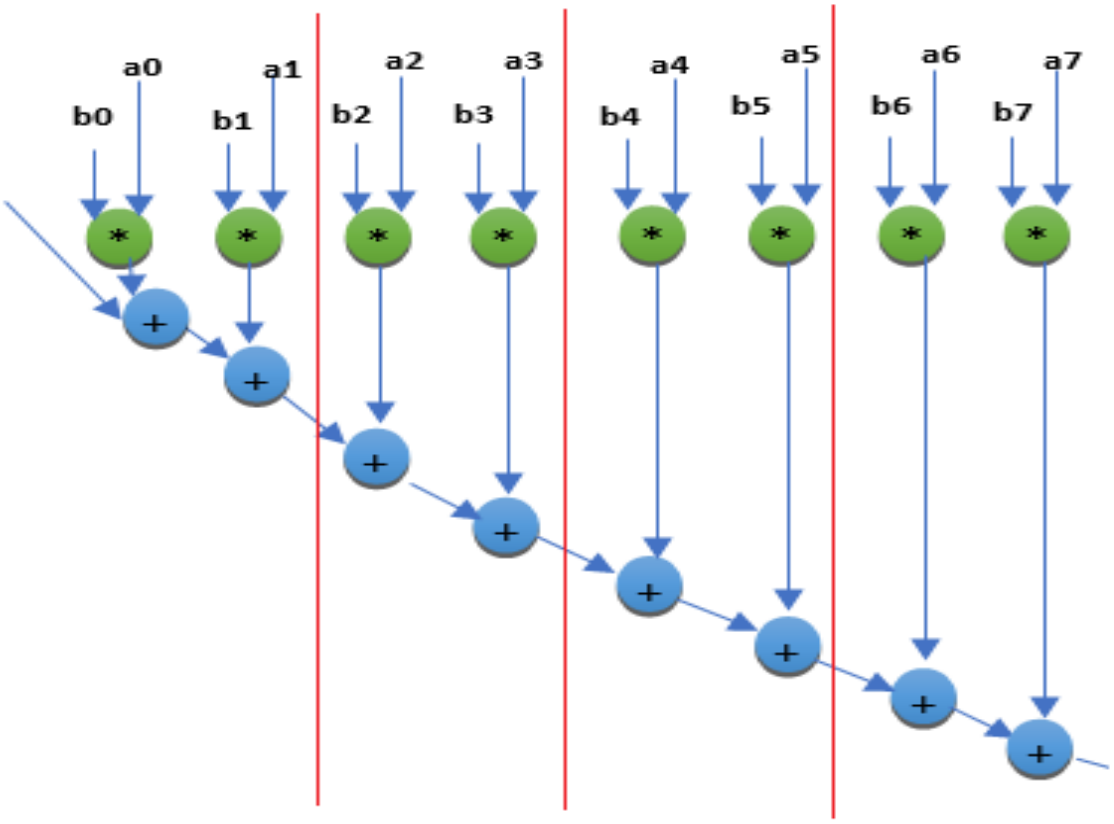


Fig. III.4 Partitioned DFG with number of partitions = 4

## Redundant Operation Elimination (ROE)

A high-level transformation technique which is applied to obfuscate the input CDFG by removing redundant nodes from the graph is redundant operation elimination. A node in the input graph is identified as a redundant node if there exist another node which has exactly same parents/inputs and same operation type. In our proposed approach we scan each node based on the node numbers in ascending order. If a pair of nodes is found which have same inputs and operation type then the node having higher node number is identified as a redundant node. These nodes are deleted from the graph and necessary adjustment is performed to maintain the correct functionality of the graph. Finally, ROE based structurally obfuscated graph is produced as output. For example, after applying ROE on figure below (including all the nodes shaded and non-shaded) only shaded nodes remain in the CDFG (non-shaded nodes are redundant thus excluded). According to our proposed analysis nodes 6, 8, 9, 10, 12 are eliminated. Nodes 5 and 6 are redundant operations hence 6 is eliminates, as a result the input of 9 and 10 gets changed from 6 to 5. Similarly, 7, 8, 9, 10 are also satisfying aforementioned conditions, therefore, 8, 9, 10 are deleted and simultaneously inputs of their children is also changed. After applying the ROE transformation, obfuscated design is obtained.

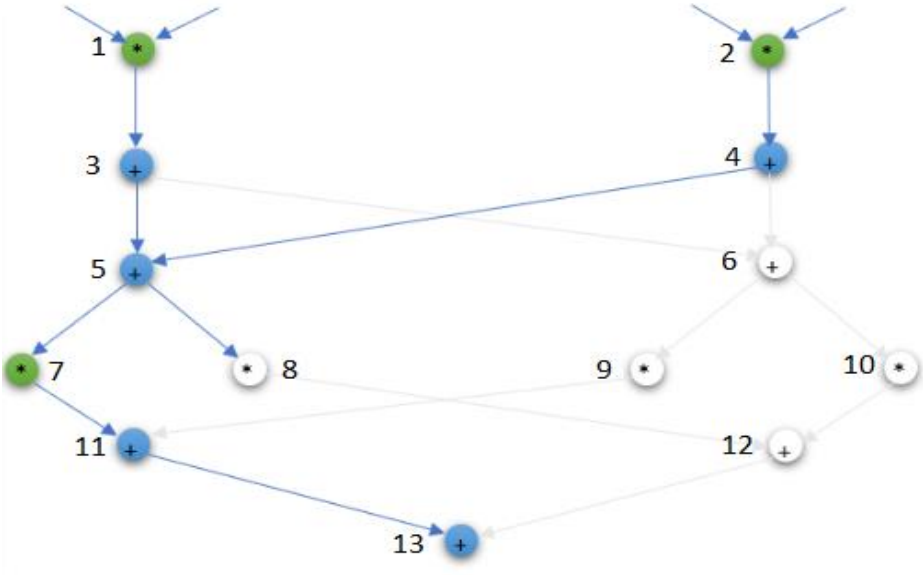


Fig. III.5 DFG in which ROE is applied

## Tree Height Transformation (THT)

A high-level transformation technique which is responsible for obfuscating the input CDFG by increasing or decreasing the height of the graph is tree height transformation. It divides the critical path dependency into temporary sub-computations and evaluates in parallel, thereby generates structurally dissimilar yet functionally equivalent graph. Finally, THT based structurally obfuscated graph is produced as output. For example, in the figure below on right, THT-based structurally obfuscated form of the input graph (shown in figure below on the left) is shown. It reduces the height of the obfuscated graph from 5 to 4 compared to the original design (figure on the left). The computation of node 6 and 8 in obfuscated design is executed prior to the input design. The dependencies of the obfuscated graph are adjusted to maintain the correct functionality. The modified dependencies are marked with red lines.

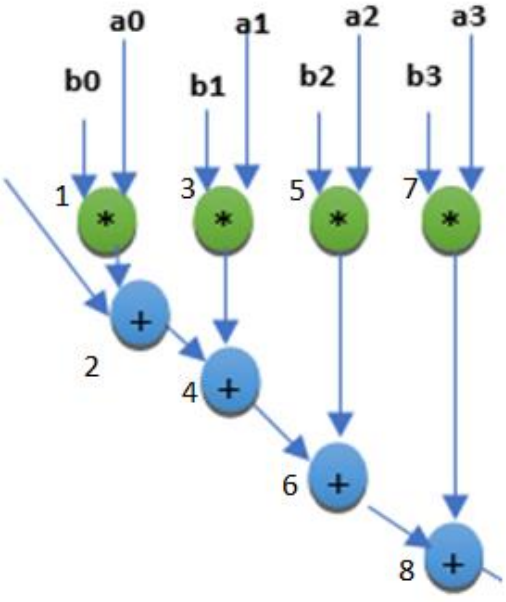


Fig. III.6 DFG before THT

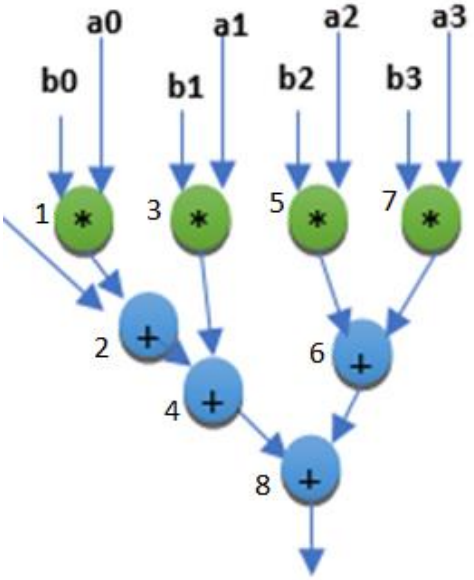


Fig. III.7 DFG after THT

## Folding

A high-level transformation technique in which a common resource is shared among a set of nodes that have the same operation type but are present in successive control steps. The number of nodes in a set sharing the resource is called the folding factor. The value of the key specified by the user denotes the number of folding transformations to be applied which is known as the folding knob. The impact that folding transformation has on the final RTL design is that it increases the number of delay elements (also known as registers) thereby structurally obfuscating the design without causing any change to the functionality. For example, in the figure below nodes 6 and 8 have the same operation type and are present in successive control steps and hence, folded. In, the example value of folding factor is 2 and the value of folding knob is 2.

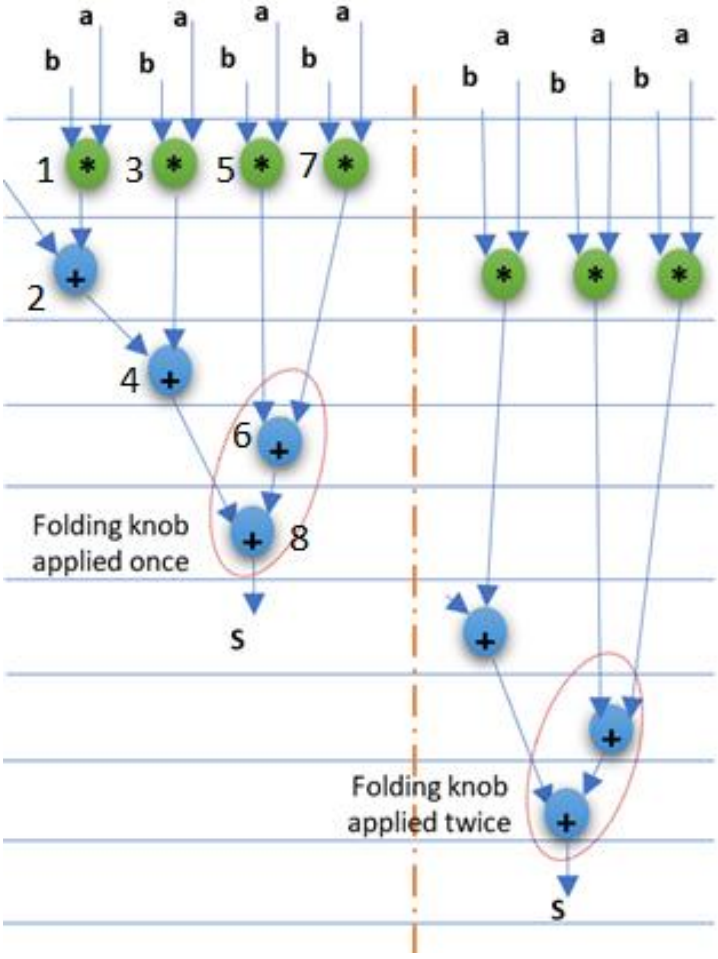


Fig. III.8 CDFG showing Folding

## III.2.2 Physical Level Watermarking

## Floorplanning

After, applying the above-mentioned transformations RTL design is created. Using the RTL design a list of resources comprising of all the resources used in the RTL design is generated which sorted by area of resource. Using the resource list, the dimensions of a hypothetical block are computed, whose width equals maximum width of all the resources present in the resource list and height equals maximum height of all the resources present in the resource list. These blocks are then positioned one after the other in x-direction and one above the other y-direction. Whenever, any block is added it is ensured that all the blocks placed so far forms a rectangle and incase if they don't then the voids of the smallest rectangle that can be formed are filled with empty blocks. The resources are then positioned inside these blocks. Following two rules must be satisfied by the resources positioned inside the blocks: -

i. Only resources of the same type can be positioned one above the other within a block.

ii. Positioning of resources must be done from the leftmost corner of the block in y-direction.

Once, all resources in the resource list get placed inside some block floorplan is generated.

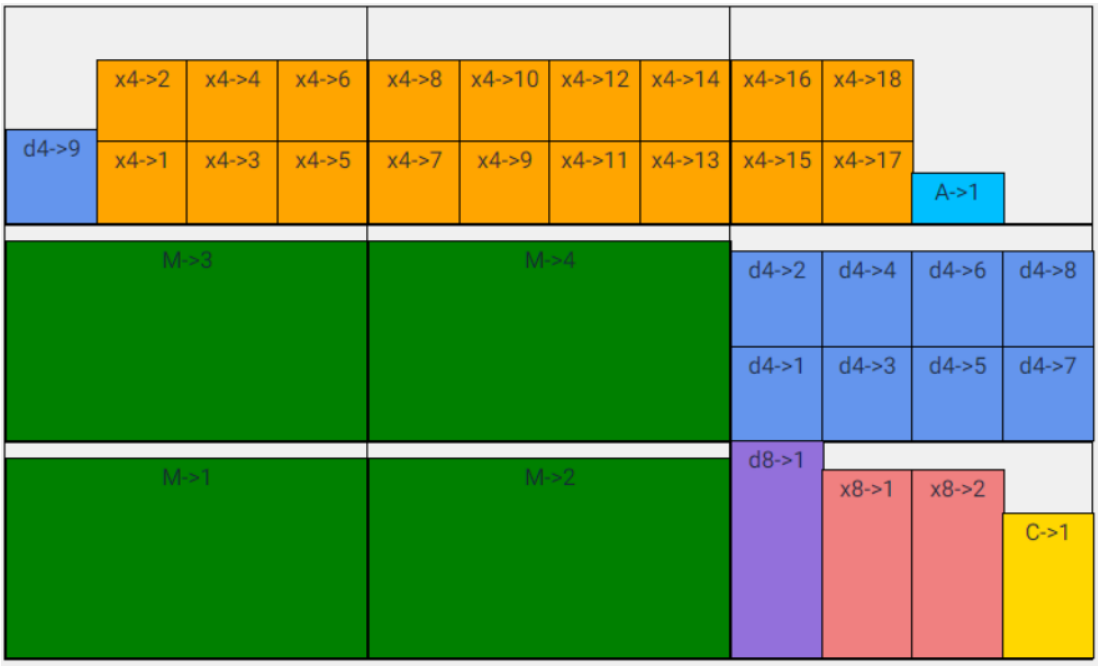


Fig. III.9 Floor plan

## Embed Watermark

After generating the floorplan watermark is embedded in it. The user will give a signature comprising of combination of α, β and γ to be embedded in the floorplan. To embed the watermark, we create three different lists namely functional units list consisting of all the functional units used in the resource list, multiplexer list comprising of all the multiplexers used in the resource list and demultiplexer list comprising of all the demultiplexers used in the resource list. The three variables used in watermark are α, β, γ.

α: odd functional unit module will be placed on top of even functional unit module by swapping between two modules of same type.

β: odd multiplexer will be placed on top of even multiplexer of same size by swapping between two multiplexers of same size.

γ: odd demultiplexer will be placed to the right of even demultiplexer of the same size by swapping between two demultiplexers of same size.

By satisfying the conditions for α, β and γ the watermark is finally embedded in the floorplan.

The figure below shows final floorplan after embedding watermark αβαβββγγββγγββ

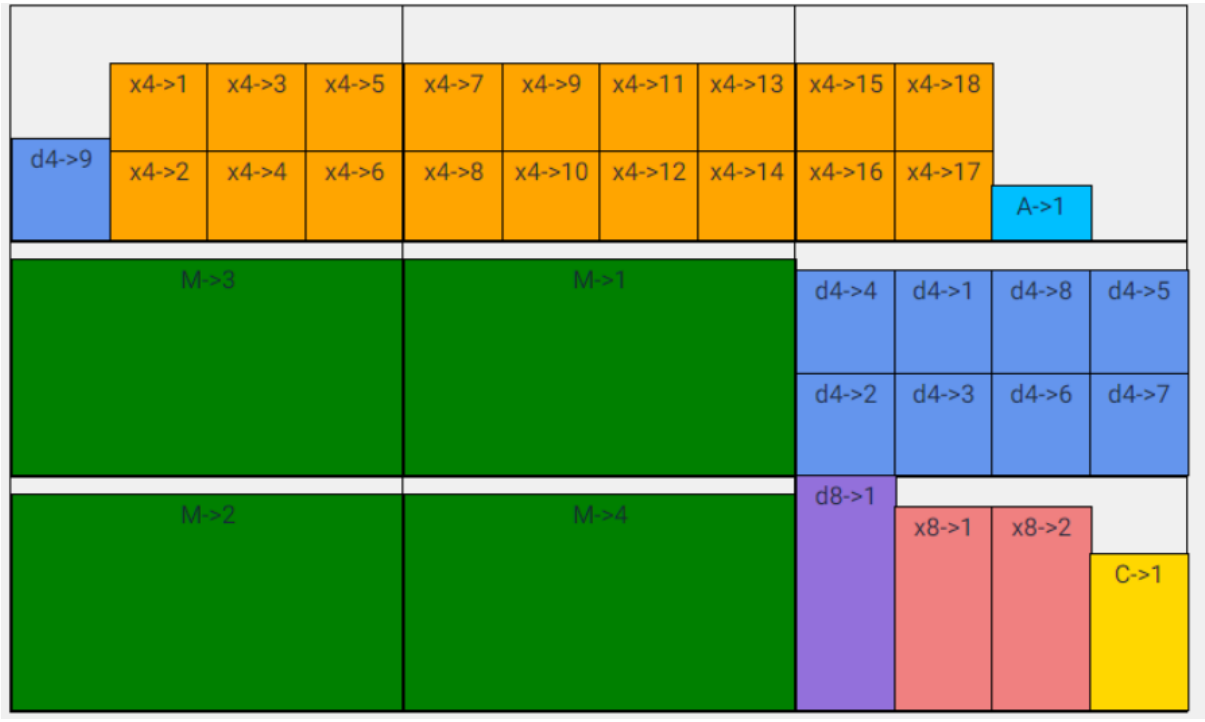


Fig. III.10 Final floor plan after embedding watermark

# IV. Demonstration on a Standard Application

We are going to follow an example of 160 TAP FIR filter with obfuscation key 16-5-5-4 and watermark signature ααβββββββββγγγγγ.

1.We have a DFG from equation of 160 TAP FIR filter.

$$y[n] = \sum_{i=1}^{160} h[i] * x[n-i]$$

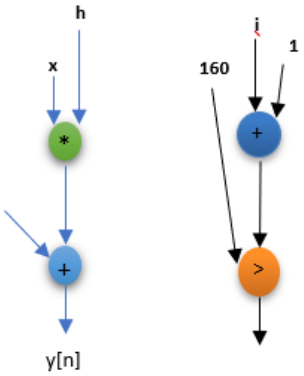


Fig IV.1. DFG representing 160-tap FIR application

2. We choose a key value of 16 for the loop unrolling and get the following DFG.

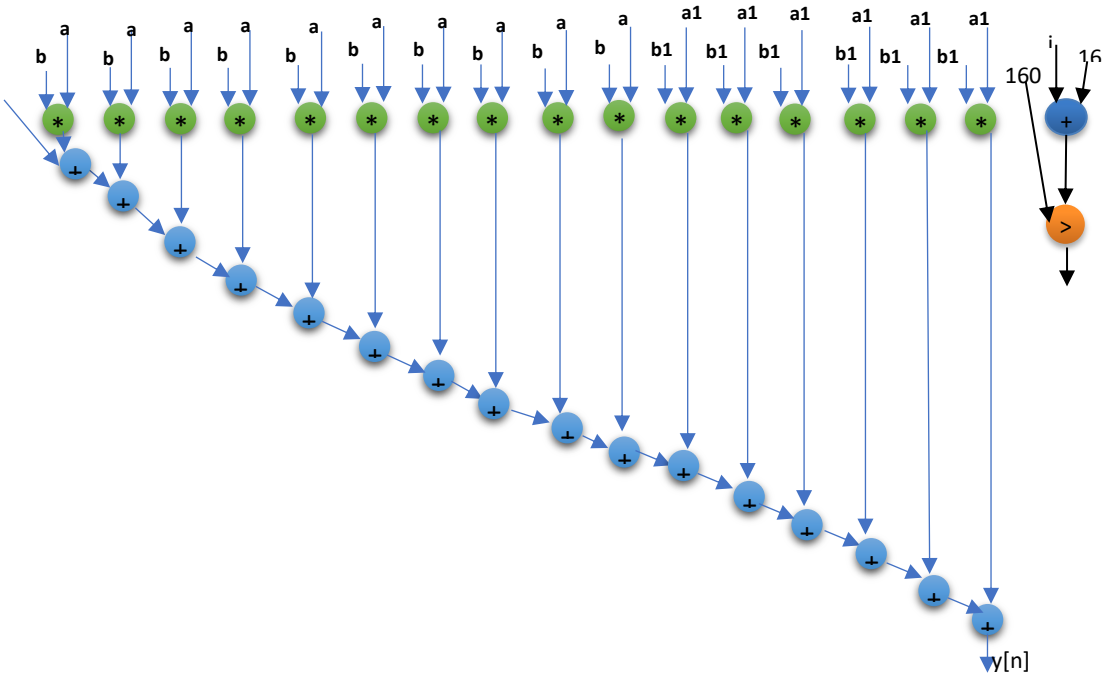


Fig. IV.2 Loop unrolled FIR application with UF=16

3. We choose key value of 5 for the partitioning which gives us the following the partitioned DFG.

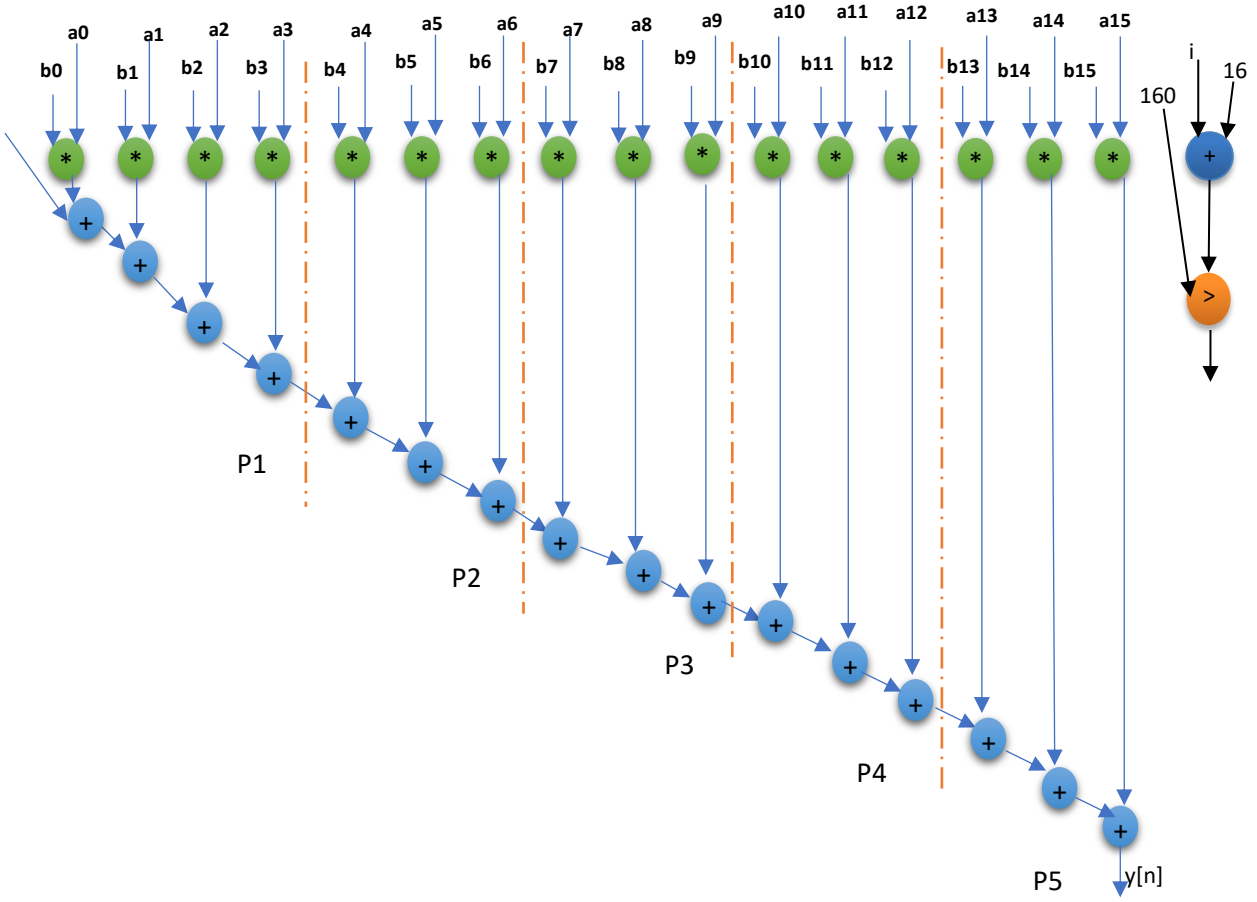


Fig. IV.3 5-Partitioned Loop unrolled FIR application with UF=16

4. Since redundant operation elimination is not applicable on this DFG we move on to next phase.

5. Apply Tree Height Transformation for the key value of 5.

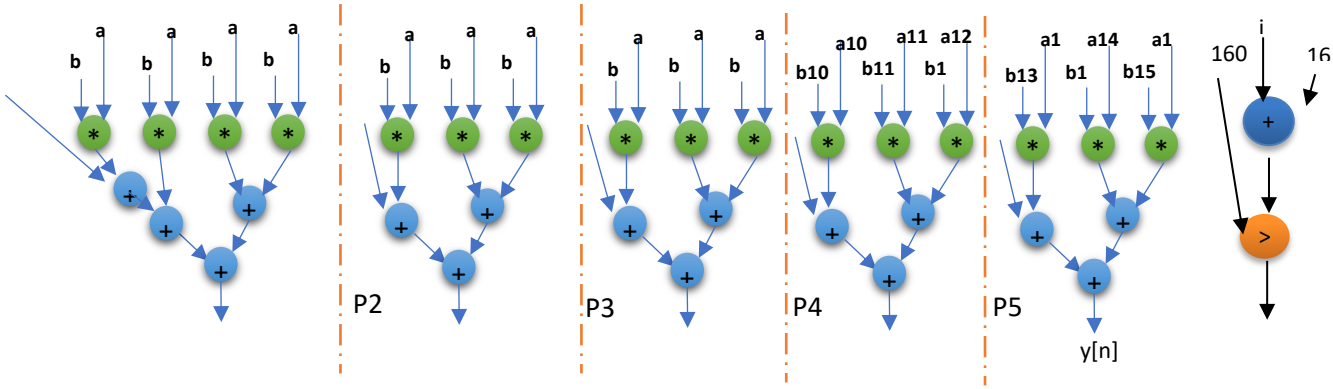


Fig. IV.4 After applying THT based structural obfuscation according to key-5

6. After scheduling and resource allocation (4M,1A), we applying folding according to value of 4.

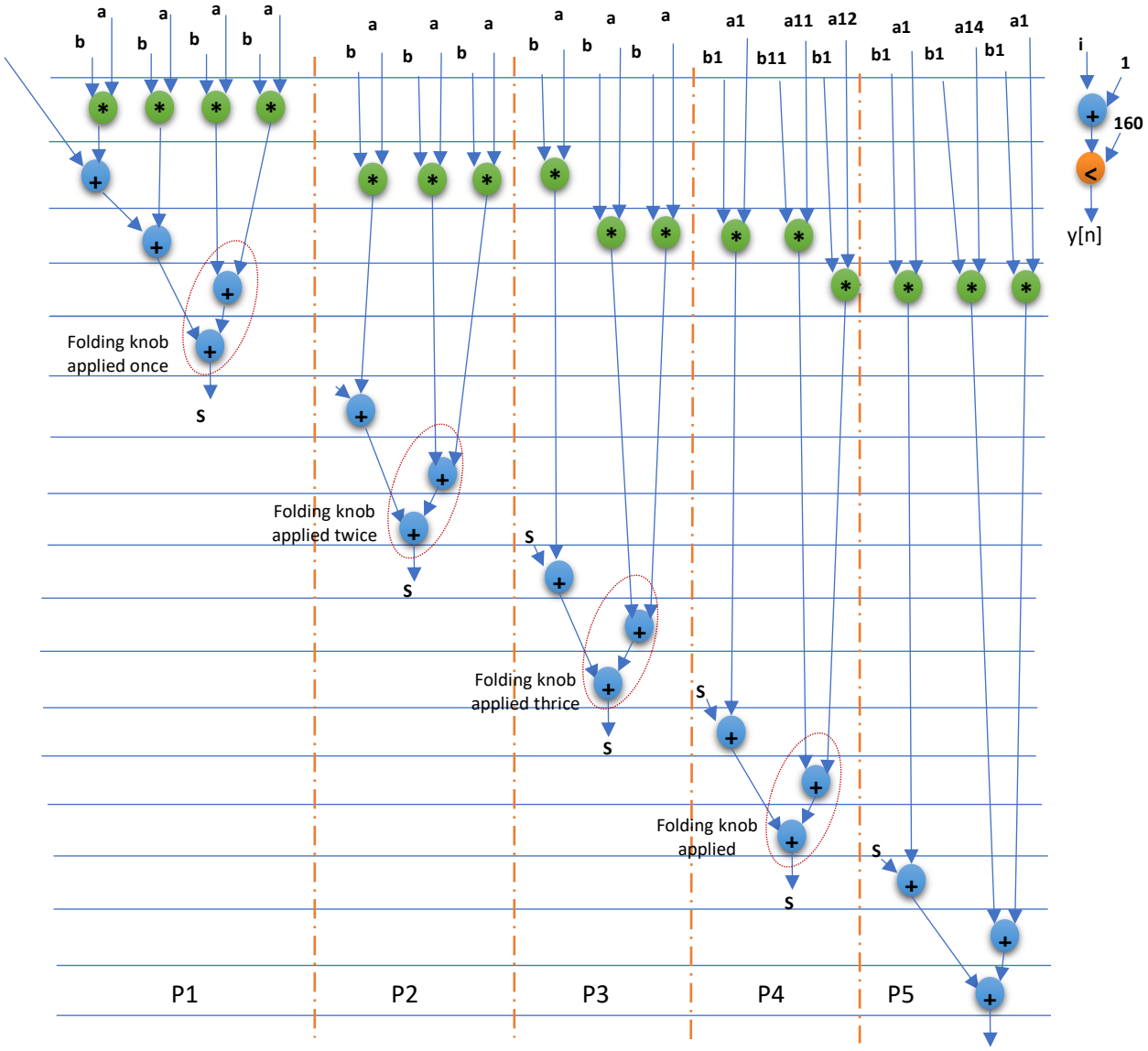


Fig. IV.5 Scheduled FIR application with applied folding four times.

7. After this we generate the RTL for each obfuscated partition.

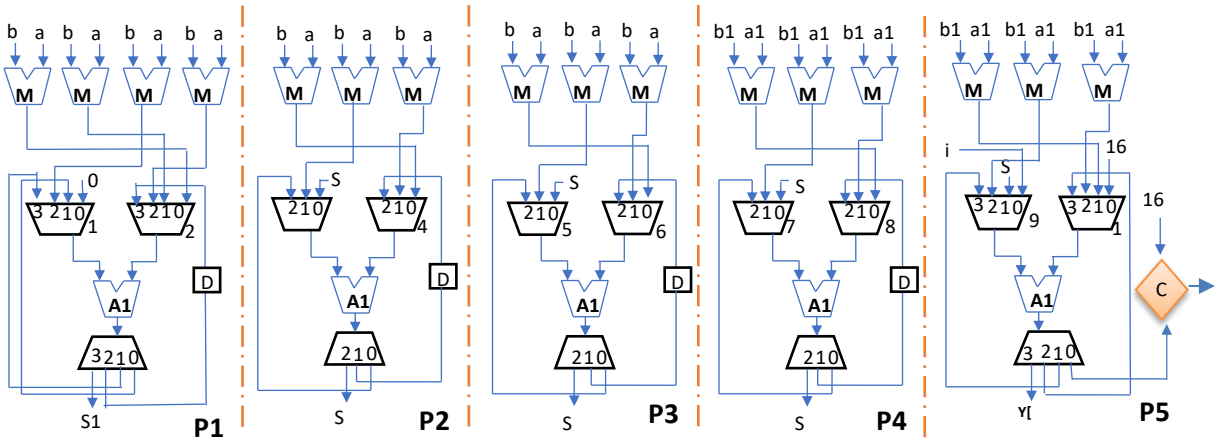


Fig. IV.6 Fig Obfuscated RTL data path of each partition

8. Now combine the partitions to obtain combined RTL data path.

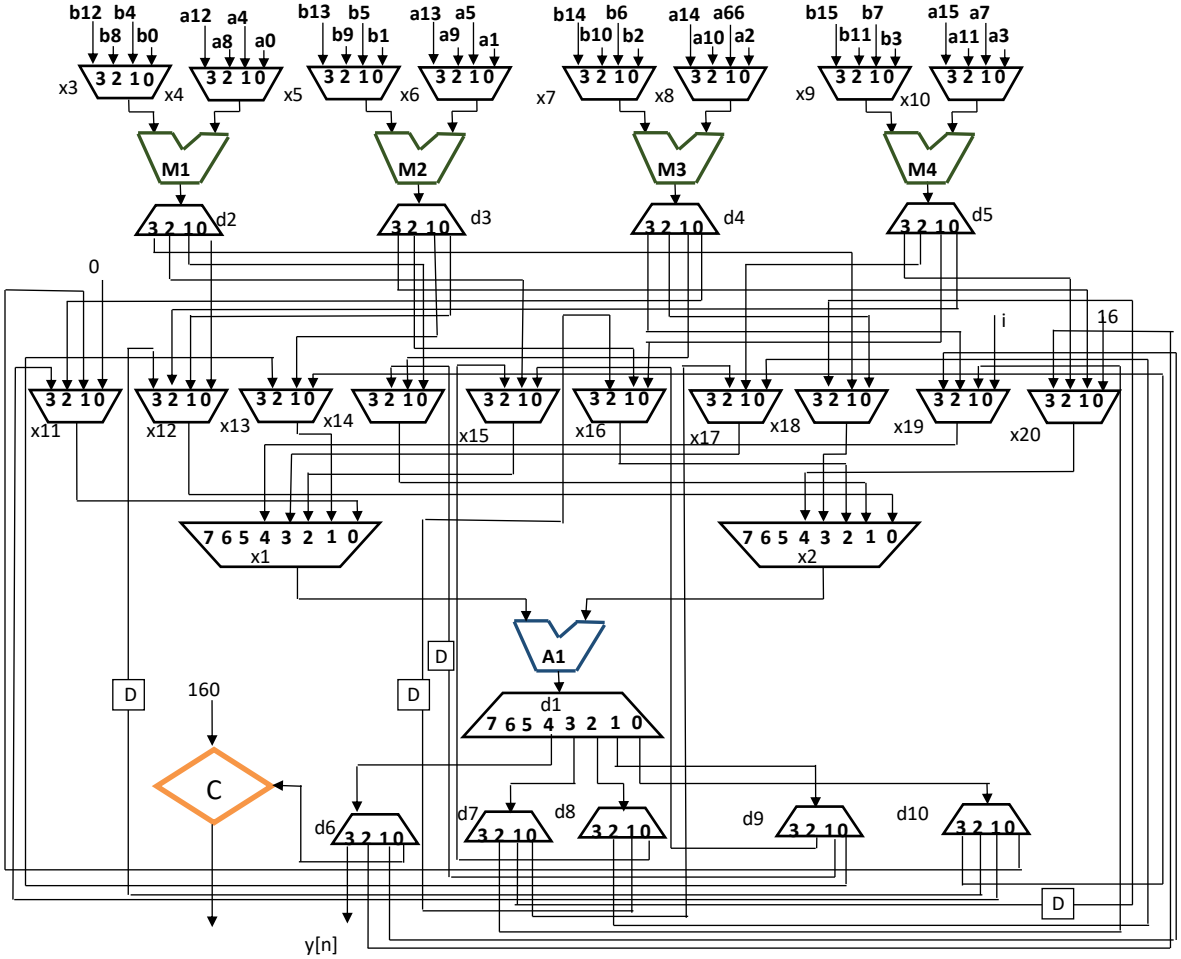


Fig. IV.7 Key-based structurally obfuscated RTL Datapath of FIR application

We get the resource list sorted by area from the above as following:

Multiplier (M): 1,2,3,4.

Demultiplexer 1:8 (d8): 1.

Multiplexer 8:1 (x8): 1,2.

Comparator (C): 1.

Demultiplexer 1:4 (d4): 1,2,3,4,5,6,7,8,9.

Multiplexer 4:1 (x4): 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18.

Adder (A): 1.

9. Now we place the units according the mentioned algorithm and we have the following floorplan.

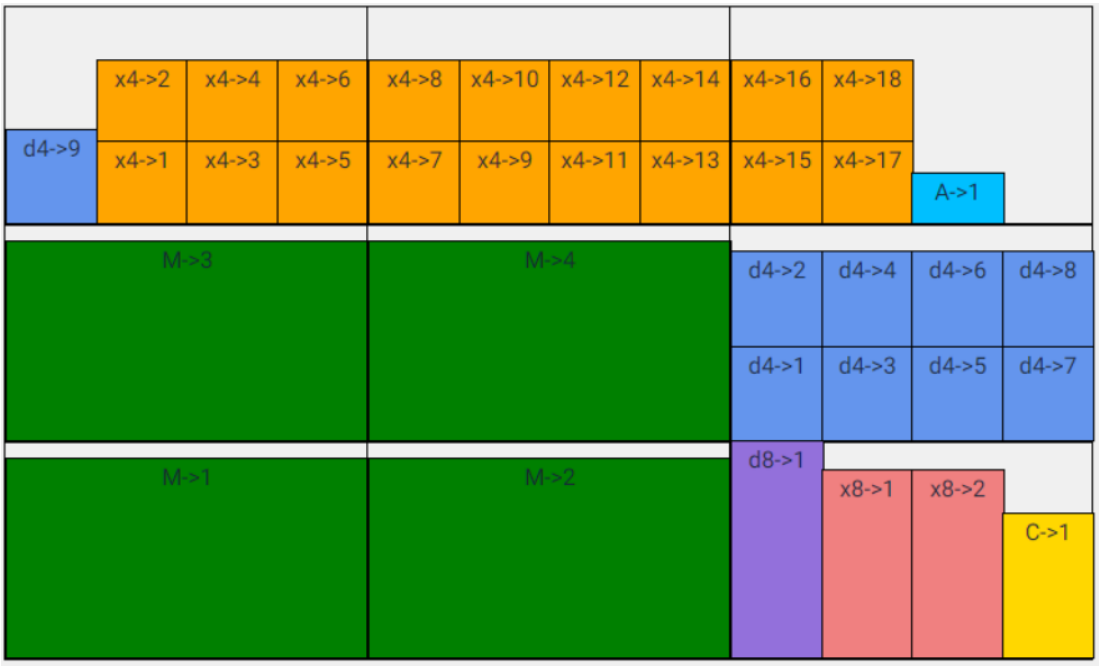


Fig. IV.8 Initial Floorplan

10. After embedding the alphas we get the floorplan as

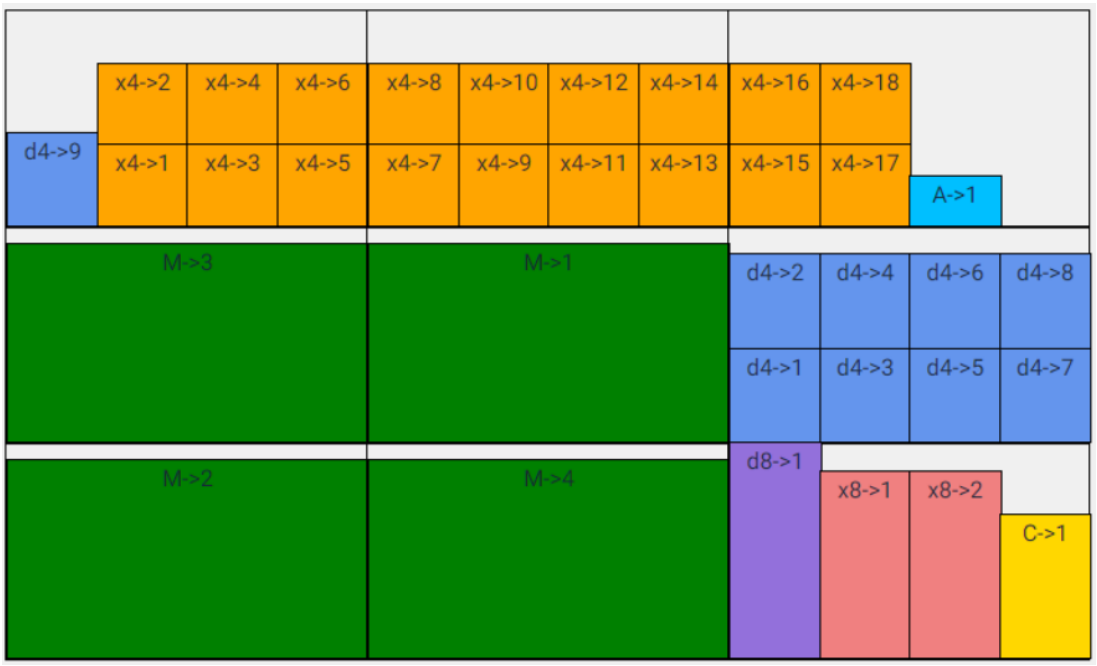


Fig. IV.9 Floorplan after embedding 2 α.

11. After all the betas are embedded, we have floorplan as

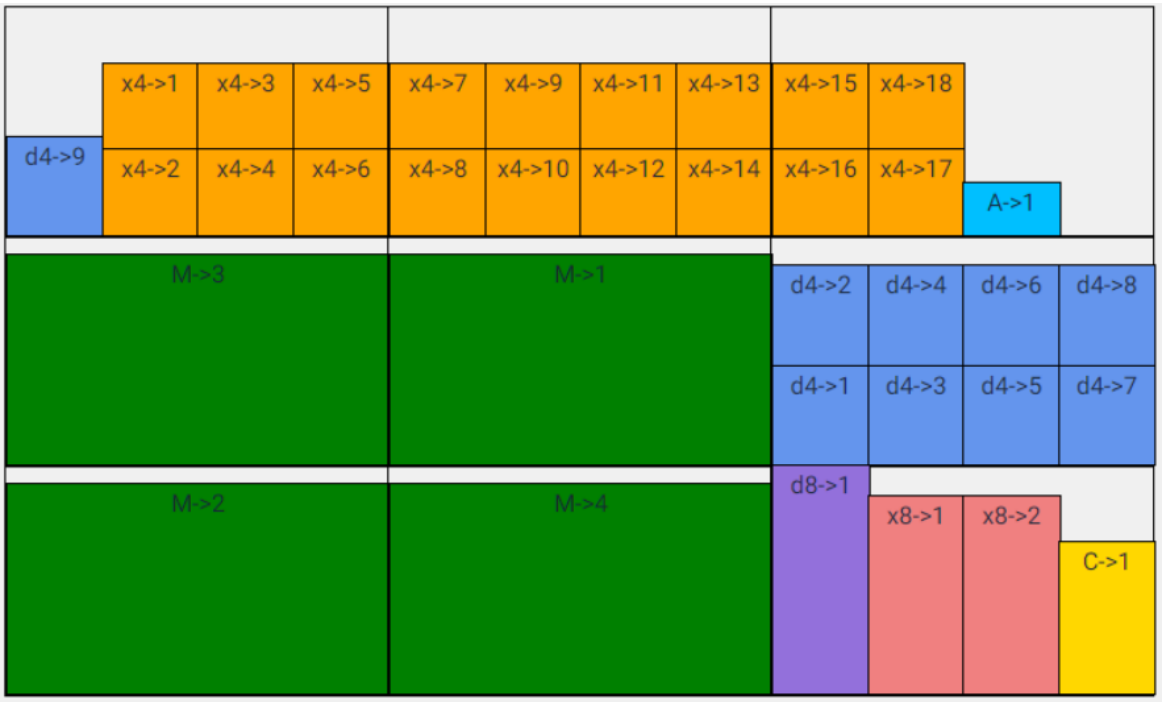


Fig. IV.10 Floorplan after embedding 2 α and 8 β

12. After embedding the gammas, we get the final floorplan as following.

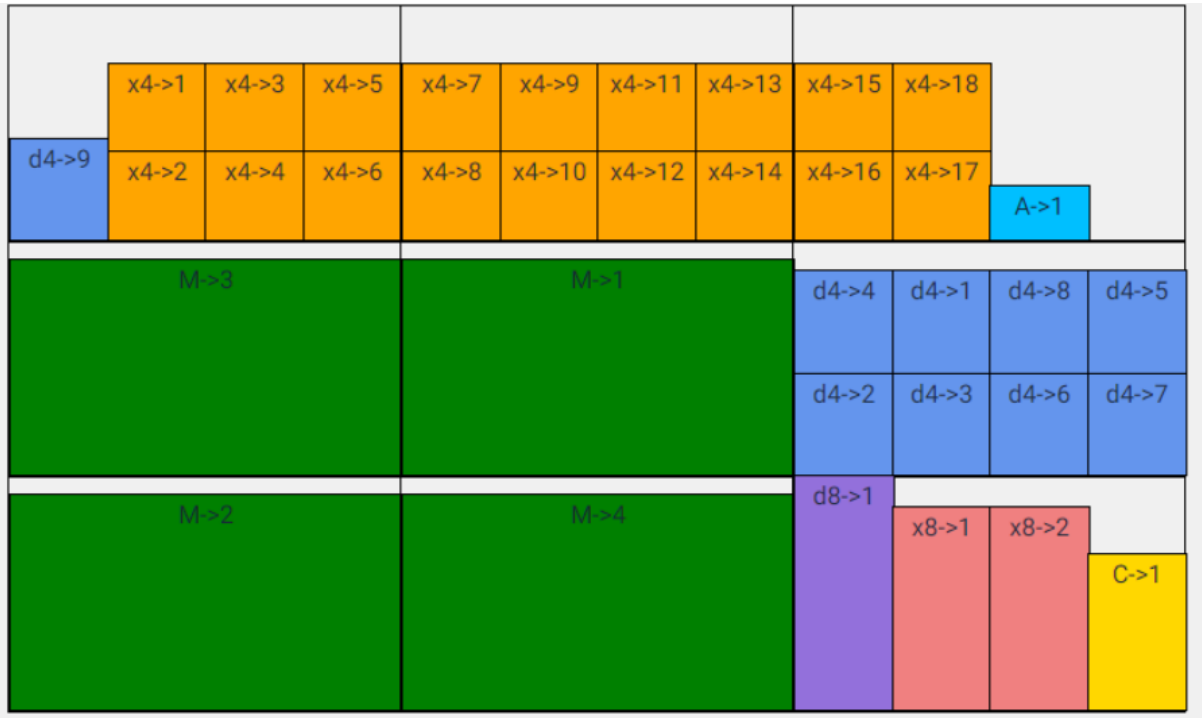


Fig IV.11 Floorplan after embedding 2 α, 8 β, 4 γ

# V. Results

## V.1 Metrics Calculation

### Design Cost

Design Cost for any phase is calculated by the following formula:

$$Cost = 0.5 \times \frac{A}{Amax} + 0.5 \times \frac{L}{Lmax}$$

Where,

A = Area of the current design

$A_{max}$ = Area for the current design using the maximum possible resources

L = Latency for the current design

$L_{max}$ = Latency for the current design using 1 instance of each type of operational resource.

Area is calculated by adding the areas of all the units which will be used in the RTL.

Latency is calculated by adding the max latency for each of the control steps and take cares of the delays of operations, multiplexers and demultiplexers.

For the cost after floor-planning the latency remains the same but area calculation changes, it is measured as the area of the rectangular envelope required to cover the all modules.

### Watermark metrics

i.) Probability of coincidence $P_c$:

$$Pc = \left( \prod_{p=1}^{\alpha} \frac{1}{(\sum_{n1 \in R_m} \frac{n1(n1-1)}{2}) - (p-1)} \right) * \left( \prod_{q=1}^{\beta} \frac{1}{(\sum_{n2 \in X_u} \frac{n2(n2-1)}{2}) - (q-1)} \right) * \left( \prod_{r=1}^{\gamma} \frac{1}{(\sum_{n3 \in D_v} \frac{n3(n3-1)}{2}) - (r-1)} \right)$$

Where, ‘n1’ belongs to FU of type $R_m$, where m is the total types of FUs; ‘n2’ belongs to Multiplexer of size $X_u$, where u indicates different sizes of Multiplexer in the design; ‘n3’ belongs to Demultiplexer of size $D_v$, where v indicates different sizes of Demultiplexer in the design.

ii.) The tamper tolerance ability T:

$$T = V^w$$

Where, V indicates the number of variables corresponding to the watermark signature W indicates the size of the signature (watermark strength).

iii.) Probability of finding watermark signature through brute force attack $P_b$

$$Pb = \frac{1}{T}$$

## V.2 Design Cost and Watermark Metrics

**Table V.1** gives design costs for the various phases of key based structural obfuscation for a particular obfuscation key value for various DSP applications.

Table V.1: Costs for various phases for a given key value

| **DSP Application** | **Key Value** | **Baseline Cost** | **Cost** | | | | |
|---|---|---|---|---|---|---|---|
| | | | Loop Unroll | Partitioning | ROE | THT | Folding |
| **FIR** | 16-5-4-4 | 1 | 0.4141 | 0.3987 | NA | 0.3700 | 0.3715 |
| **DE** | 2-2-2 | 0.7528 | 0.5989 | 0.5787 | NA | NA | 0.5787 |
| **DCT** | 3-1-3 | 0.5001 | NA | 0.5197 | NA | 0.5057 | 0.5094 |
| **IIR** | 2-1-4 | 0.5465 | NA | 0.5333 | NA | 0.5186 | 0.5186 |
| **ARF** | 1-4-10 | 0.4629 | NA | 0.4629 | 0.5195 | NA | 0.5242 |

**Table V.2** Gives us a comparison of how final design cost after obfuscation varies with different obfuscation keys for various DSP applications and how they compare to the baseline design cost.

Table V.2: Baseline vs Final Costs for various keys

| DSP Application | Baseline Cost | Key and Final Cost | | | | | |
|---|---|---|---|---|---|---|---|
| | | Key 1 | Cost | Key 2 | Cost | Key 3 | Cost |
| **FIR** | 1 | 16-5-4-4 | 0. 3715 | 20-4-4-4 | 0.3698 | 32-8-8-8 | 0.3023 |
| **DE** | 0.7528 | 2-2-2 | 0.5787 | 4-2-12 | 0.4591 | 8-4-16 | 0.3853 |
| **DCT** | 0.5001 | 3-1-3 | 0.5094 | 2-2-2 | 0.4799 | 4-4 | 0.5276 |
| **IIR** | 0.5465 | 2-1-4 | 0.5186 | 3-1-4 | 0.5567 | 4-5 | 0.5459 |
| **ARF** | 0.4629 | 1-4-10 | 0.5242 | 2-4-10 | 0.4891 | 3-10 | 0.4799 |

**Table V.3** gives us the final design costs after taking consideration of floorplanning effects in the area calculation for different obfuscation keys for various DSP applications.

Table V.3: Baseline vs Final Costs for various keys (after Floorplanning)

| DSP Application | Baseline Cost | Key and Final Cost (after Floorplanning) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Key 1 | Cost | Key 2 | Cost | Key 3 | Cost |
| **FIR** | 1 | 16-5-4-4 | 0.3649 | 20-4-4-4 | 0.4028 | 32-8-8-8 | 0.3023 |
| **DE** | 0.7528 | 2-2-2 | 0.6682 | 4-2-12 | 0.4743 | 8-4-16 | 0.4377 |
| **DCT** | 0.5001 | 3-1-3 | 0.5060 | 2-2-2 | 0.5057 | 4-4 | 0.4790 |
| **IIR** | 0.5465 | 2-1-4 | 0.4901 | 3-1-4 | 0.5642 | 4-5 | 0.4971 |
| **ARF** | 0.4629 | 1-4-10 | 0.5691 | 2-4-10 | 0.5441 | 3-10 | 0.5028 |

**Table V.4** shows how the strength of watermark varies with the signature for each DSP application with their keys.

Table V.4: Watermark Metrics

| DSP Application (Key) | Watermark Signature | Probability of coincidence | Tamper Tolerance | Probability of Random Brute Attack Success |
|---|---|---|---|---|
| FIR (16-5-5-4) | ααββββββββγγγγ | 6.1414E-28 | 14348907 | 6.9691E-8 |
| | αααββββββββγγγγ | 8.9665E-26 | 4782969 | 2.0907E-7 |
| | αααββββββββγγγ | 2.9589E-24 | 1594323 | 6.2722E-7 |
| DE (2-2-2) | αααβββββββββγγ | 8.3883E-27 | 14348907 | 6.9691E-8 |
| | αααββββββββγγ | 4.9071E-24 | 1594323 | 6.2722E-7 |
| | ααββββββββγ | 1.3740E-22 | 531441 | 1.8816E-6 |
| DCT (3-1-3) | βββββββγγ | 1.3227E-16 | 59049 | 1.6935E-5 |
| | ββββββγγ | 3.2406E-13 | 6561 | 1.5241E-4 |
| | βββββββγ | 7.1294E-14 | 6561 | 1.5241E-4 |
| IIR (2-1-4) | βββββββγ | 3.9519E-14 | 6561 | 1.5241E-4 |
| | ββββββγ | 2.1340E-12 | 2187 | 4.5724E-4 |
| | βββββγ | 1.1737E-10 | 729 | 0.0014 |
| ARF (1-4-10) | αααββββγγ | 2.2603E-9 | 19683 | 5.0805E-5 |
| | αααβββγγ | 2.1473E-7 | 2187 | 4.5724E-4 |
| | αααβββγ | 1.2883E-7 | 2187 | 4.572E-4 |

We observed an average runtime of 68.8 ms on Intel i3-5005U processor and 4GB of DDR3 ram.

# VI. Conclusion

Through our work we tried to bring new innovation in this field. We proposed a first ever key based structural obfuscation algorithm providing the IP designer to have control over extent of obfuscation over each of the phases of the obfuscation. This algorithm is also first of its kind which uses partitioning based obfuscation approach.

We also proposed first ever physical level watermarking algorithm which allows us to combine the benefits of structural obfuscation and watermarking.

Thus, we have a one of its kind algorithm which combines two types of DSP securing techniques, with negligible to no extra overhead in most of the cases we tested upon.

# VII. References

1. A. Sengupta, S. Bhadauria and S. P. Mohanty, "Embedding low cost optimal watermark during high level synthesis for reusable IP core protection," 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, 2016, pp. 974-977.

2. A. Sengupta and D. Roy, "Multi-phase watermark for IP core protection," 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2018, pp. 1-3.

3. A. Sengupta, D. Roy, S. P. Mohanty and P. Corcoran, "DSP design protection in CE through algorithmic transformation based structural obfuscation," in IEEE Transactions on Consumer Electronics, vol. 63, no. 4, pp. 467-476, November 2017.

4. A. Sengupta, S. Neema, P. Sarkar, S. Harsha P, S. P. Mohanty and M. K. Naskar, "Obfuscation of Fault Secured DSP Design Through Hybrid Transformation," 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, 2018, pp. 732-737.

5. F. Koushanfar, I. Hong, M. Potkonjak, "Behavioral Synthesis Techniques for Intellectual Property Protection", ACM Trans. Des. Autom. Electron. Syst., vol. 10, no. 3, pp. 523-545, July 2005.

6. I. Hong, M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection", Proc. of the 36th annual ACM/IEEE Design Automation Conference, pp. 849-854, 1999.

7. J. Zhang, "A Practical Logic Obfuscation Technique for Hardware Security", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 3, pp. 1193-1197, March 2016.

8. X. Wang, X. Jia, Q. Zhou, Y. Cai, J. Yang, M. Gao, G. Qu, "Secure and low-overhead circuit obfuscation technique with multiplexers", 2016 International Great Lakes Symposium on VLSI, pp. 133-136, May 2016.

9. J.A. Roy, F. Koushanfar, I.L. Markov, "EPIC: Ending Piracy of Integrated Circuits", 2008 Design Automation and Test in Europe, pp. 1069-1074, March 2008.

10. Y. Lao, K.K. Parhi, "Obfuscating DSP Circuits via High-Level Transformations", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 5, pp. 819-830, May 2015.

11. A. Sengupta, D. Roy, "Protecting an intellectual property core during architectural synthesis using high-level transformation based obfuscation", Electronics Letters, May 2017