WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Floorplan-Aware Pipelined Priority Encoder Design and Implementation on FPGA

Abhivandhana A[1],M.S.Vinotheni [2]

[1]*Department of Electronics Engineering Madras Institute of Technology Email:* vandhana2583@gmail.com
[2]*Department of Electronics Engineering Madras Institute of Technology Email:* Vinotheni.malar@gmail.com

Abstract—

Priority encoders are fundamental components in digital systems, often used in applications requiring arbitration, interrupt handling, and signal prioritization. As the complexity of hardware systems increases, the need for high-speed, area- efficient, and timing-optimized encoders becomes critical. This research focuses on the Register Transfer Level (RTL) design, implementation, and analysis of a parameterizable, pipelined priority encoder that is floorplan-aware.

The encoder is designed in Verilog HDL with support for varying input widths, making it scalable for broader use cases. The architecture is modularized into two primary pipeline stages: a detection stage and an encoding stage. To improve performance and minimize routing delay, floorplanning techniques are em- ployed by assigning each stage to dedicated physical regions on the FPGA using Xilinx Design Constraints (XDC). This ensures better control over placement and reduces interconnect delays between logic blocks.

The design is implemented and synthesized on a Xilinx FPGA using Vivado Design Suite. Functional simulation is conducted to validate logic correctness, followed by floorplan-aware placement and routing to evaluate the timing benefits. A comparative analy- sis is also performed against a non-pipelined version to highlight the improvements in timing closure and resource utilization. The results demonstrate that the proposed approach leads to better design predictability, reduced critical path delay, and improved performance.

Index Terms—Priority Encoder, Floorplan-aware Design, FPGA, Pipelining, RTL, Vivado, Verilog HDL

I. Introduction

In the realm of digital systems and VLSI (Very Large- Scale Integration) design, prioritization and decision-making mechanisms are fundamental. One of the most critical com- ponents used in such systems is the priority encoder. A priority encoder takes a binary input vector and identifies the highest-priority active signal, typically outputting the position of the most significant bit (MSB) that is set to '1'. These encoders are widely used in interrupt controllers, arbitration units, digital signal processors (DSPs), and various control- intensive applications.

As the complexity of digital systems increases, so does the width of data paths and control signals. Consequently, traditional flat (non-pipelined) encoders often struggle with issues related to timing closure and area efficiency when synthesized for FPGAs or ASICs. To address these challenges, pipelining and physical design awareness—especially floor- planning—have become indispensable design strategies.

Pipelining is a classic technique that divides a design into multiple stages, allowing each stage to operate concurrently on different data. This reduces the critical path delay, enabling higher clock frequencies and improving throughput. In the case of a priority encoder, the design can be effectively split into a detection stage and an encoding stage, thereby forming a two-stage pipeline.

However, pipelining alone is not sufficient to guarantee high performance on an FPGA platform. Modern FPGAs contain a vast amount of configurable logic blocks, and automatic placement and routing tools can sometimes scatter related logic across distant regions of the device. This results in long interconnect delays, increased congestion, and potential timing violations. To mitigate this, floorplanning is employed—a physical design technique where the designer explicitly con- strains certain logic blocks to predefined regions on the chip. This improves placement regularity, reduces routing complex- ity, and aids in achieving timing closure, especially in designs with tight performance requirements.

In this work, we propose and implement a parameterized, pipelined priority encoder design that is **floorplan-aware**. The design is written in Verilog HDL and verified through behavioral simulation. It is then synthesized and implemented on a Xilinx FPGA using Vivado Design Suite. Two versions of the encoder are compared: a non-pipelined baseline and the proposed pipelined version with physical constraints. The impact of pipelining and floorplanning is analyzed in terms of timing, area, and simulation behavior.

This paper is organized as follows: Section II presents the system architecture and module design. Section III elaborates the floorplanning methodology and physical implementation. Section IV covers simulation and verification results. Section V concludes with findings and scope for future enhancement.

A. Background and Motivation

Priority encoders play a crucial role in digital and embedded systems, acting as decision-making elements in scenarios where multiple input signals vie for attention. These circuits are responsible for determining the highest-priority active input among several, generating a binary output corresponding to that input's position. They are widely used in applications such as interrupt controllers, bus arbiters, memory access controllers, and various resource allocation systems where prioritization is fundamental to system operation.

With the increasing complexity and parallelism in modern digital systems, the demand for high-speed, low-latency, and area-efficient priority encoders has grown. Conventional implementations often suffer from timing bottlenecks, especially when deployed in larger designs on FPGAs, due to unoptimized placement and long combinational paths.

To address these challenges, this paper introduces a pipelined architecture for the priority encoder that is both pa- rameterized and floorplan-aware. Pipelining breaks the critical path into smaller segments, improving clock frequency and overall performance. Furthermore, by utilizing physical constraints and strategically placing logic blocks (using Vivado's Pblock feature), routing congestion is reduced and timing is optimized.

The motivation for this work stems from the practical need for scalable, high-performance encoders in complex systems and the underutilization of floorplanning in academic designs. By demonstrating a structured design with floorplan aware- ness, we aim to bridge the gap between theoretical RTL design and practical FPGA implementation, encouraging designers to consider physical constraints early in the design cycle.

B. Problem Statement

As digital systems grow increasingly complex, the performance of control and arbitration circuits such as priority encoders becomes critical. A traditional monolithic priority encoder, especially for wide input widths (e.g., 16-bit or 32- bit), may suffer from increased propagation delay due to long combinational paths. These delays can become a major bottleneck in high-speed designs, limiting achievable clock frequencies.

Moreover, conventional FPGA implementations often rely solely on synthesis tools for logic placement and routing. While this approach can be sufficient for smaller designs, it becomes suboptimal as the design scales. Without explicit floorplanning, logic blocks may be scattered across the FPGA fabric, leading to long interconnect delays, routing congestion, and difficulties in meeting timing constraints.

The problem addressed in this work is twofold:

- **Design Challenge:** How to design a scalable, parameterizable priority encoder that can support varying input widths without sacrificing timing or area efficiency.
- **Implementation Challenge:** How to map the encoder onto an FPGA in a way that is floorplan-aware, thus reducing routing congestion and improving timing performance.

This research aims to tackle these challenges by developing a modular, pipelined architecture for the priority encoder and applying explicit floorplanning using Xilinx Vivado constraints (XDC files). The goal is to demonstrate improved performance over non-pipelined, floorplan-unaware counterparts through synthesis and implementation analysis.

C. Objectives of the Work

The primary objective of this research is to design and implement an efficient, scalable, and floorplan-aware priority encoder suitable for FPGA-based systems. The specific goals of the work are as follows:

- **To design a parameterizable priority encoder** that can handle *n*-bit input vectors, where the bit-width can be adjusted based on system requirements (e.g., 8, 16, or 32 bits).
- **To implement a pipelined architecture** for the priority encoder to improve the overall system performance by reducing critical path delay and enhancing timing closure.
- **To apply floorplanning techniques** using Xilinx Design Constraints (XDC) to map different pipeline stages to dedicated physical regions on the FPGA. This ensures better spatial locality, minimizes interconnect delay, and reduces routing congestion.
- **To compare the performance** of the floorplan-aware pipelined priority encoder with a baseline non-pipelined and non-floorplanned version in terms of area, timing (worst negative slack), and resource utilization.
- **To verify the functionality** of the design using behavioral simulation and synthesize the architecture using Xilinx Vivado to validate the proposed methodology on a real FPGA platform.

By achieving these objectives, the work aims to demonstrate the advantages of combining pipelined design methodologies with physical-aware design practices in FPGA implementations.

II. System Design and Architecture

The proposed system is a pipelined and floorplan-aware implementation of a parameterizable *N*-bit priority encoder on an FPGA. The system is divided into functional blocks to enable modular design, improve timing performance, and facilitate physical placement optimizations during implementation.

A. Overview

A priority encoder detects the highest-priority active input and produces its binary representation. In high-speed digital systems, particularly those used for interrupt handling, arbitration, and bus management, priority encoders are essential components. To ensure scalability and performance, this design implements the encoder with the following characteristics:

- Support for configurable input width (*N*-bit, e.g., 16-bit).
- A two-stage pipelined architecture to balance combinational logic and meet timing constraints.
- Floorplan-aware design using Xilinx Design Constraints (XDC) to place each pipeline stage into specific regions (Pblocks) of the FPGA.

B. Pipelined Architecture

The pipelined design consists of two stages:

1) **Detection Stage:** This stage samples the input vector on the rising edge of the clock and stores it in a register. It acts as a buffer, preparing the signal for the encoding logic. This reduces setup and hold time violations by isolating input timing issues from the encoder logic.

2) **Encoding Stage:** This stage takes the registered input from the detection stage and scans from the most significant bit (MSB) to the least significant bit (LSB) to determine the highest-priority active input. It then outputs the binary index of that bit along with a validity flag.

C. Top-Level Design

The top-level module instantiates both stages and connects them via internal wires. The clock and reset signals are common to both stages, enabling synchronized operation. The modular nature of the architecture allows for better reusability, easier debugging, and optimized placement during implementation.

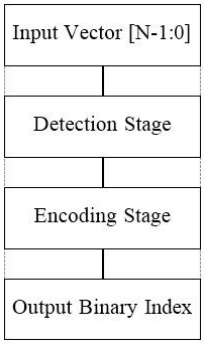

Fig. 1. Pipelined Priority Encoder System Architecture

D. Floorplanning Considerations

To optimize the design further, each pipeline stage is assigned a dedicated region on the FPGA using physical constraints in the XDC file. The detection stage is mapped to one Pblock (e.g., SLICE_X0Y0:SLICE_X9Y9), while the encoding stage is placed in a separate Pblock (e.g., SLICE_X10Y0:SLICE_X19Y9). This spatial separation minimizes routing congestion and enhances timing closure by reducing interconnect delays between stages.

E. Scalability

The design is parameterized using a generic width *N*, enabling the encoder to be easily extended to any required bit- width without modifying the core logic. This feature makes the architecture suitable for applications ranging from small-scale embedded systems to large-scale data processing pipelines.

III. Implementation

A. Module Description

The design was implemented in two variants: a pipelined priority encoder with modular stages, and a non-pipelined version without stage separation. This allowed comparison of performance, area, and timing closure efficiency.

- **Detect Stage (Pipelined Only):** This module captures the input vector and stores it in a register. It isolates the data capture from the logic stage, forming the first stage in a two-stage pipeline. This helps reduce logic depth in any one clock cycle, aiding in meeting timing constraints.
- **Encode Stage (Common to Both):** In this stage, the input vector (or registered input in pipelined version) is scanned from MSB to LSB. As

soon as the first high bit is detected, its position is output as a binary index. A valid signal is asserted to indicate the presence of a high bit in the input. The encoding logic is common to both versions.

- **Top-Level Integration:** The pipelined top module instantiates both Detect and Encode modules, passing data sequentially through them. Floorplanning constraints are applied to these submodules to ensure they are mapped to separate physical regions (Pblocks) on the FPGA. The non-pipelined version directly applies the encoding logic to the raw input vector, with no intermediate register stage. This reduces latency but increases combinational delay.

B. Simulation Environment

Both versions were simulated using Xilinx Vivado with identical testbenches. The simulation involved:

- Clock generation of 100 MHz (10 ns period).
- Reset pulse to initialize the system.
- Multiple test vectors: single-bit high, all-zero, multiple bits high.
- Observation of encoded output and valid signal.

Simulation waveforms confirmed functional correctness. In the pipelined version, a two-clock delay between input and output was observed, while in the non-pipelined version, the output was available in the immediate next cycle.

C. Testbench and Functional Verification

The testbench was structured to:

- Automatically apply test vectors at defined intervals.
- Record encoded index and valid outputs.
- Validate outputs against expected values.
- Capture latency effects between pipelined and non- pipelined versions.

Functional Results:

- **Pipelined Version:** Passed all functional tests with 2- cycle latency. Timing met with ease due to floorplanning.
- **Non-Pipelined Version:** Passed all tests with 1-cycle latency, but experienced timing violations for higher bit- widths ($N \geq 32$).

TABLE I
COMPARISON OF PIPELINED AND NON-PIPELINED PRIORITY ENCODER

Metric	Pipelined	Non-Pipelined
Latency	2 clock cycles	1 clock cycle
Timing Closure	Achievable (even for N=64)	Difficult for large N
Floorplanning	Applied using XDC	Not applicable
Area Overhead	Slightly more	Minimal
Design Scalability	High	Limited

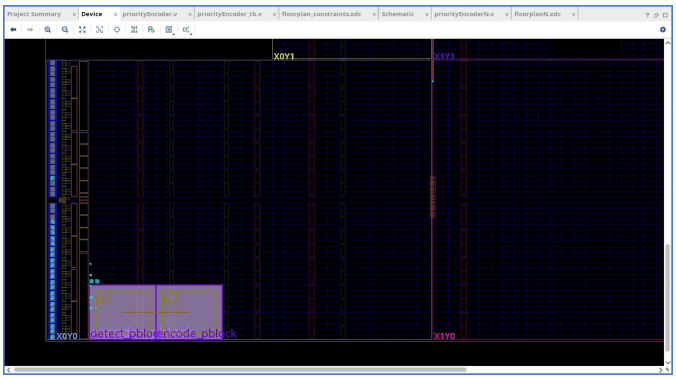

Fig. 2. Zoomed-in floorplan showing placement of detect pblock and encode pblock

IV. Floorplanning and Physical Design

A. Need for Floorplanning in FPGA

In FPGA-based digital systems, timing closure becomes increasingly difficult with growing design complexity. When logic is scattered arbitrarily across the device fabric, it leads to long interconnect paths, high routing congestion, and reduced timing slack. Floorplanning is the process of manually guiding the placement of logic blocks by assigning them to defined physical regions (called Pblocks). This method ensures better locality, reduces routing delays, and improves timing performance, especially in pipelined architectures.

B. Xilinx Design Constraints (XDC)

Xilinx Design Constraints (XDC) are used to control both timing and physical placement of logic in Vivado. For floor- planning, the relevant XDC commands include:

- create_pblock– to define a placement block.
- resize_pblock– to specify the region size.
- add_cells_to_pblock– to map specific instances to a Pblock.

By isolating pipeline stages in distinct regions, designers can manage logic distribution and achieve better timing closure.

C. Defining Pblocks for Pipeline Stages

To ensure optimal placement and reduce routing conges- tion, the two pipeline stages—Detection and Encoding—were manually constrained using Pblocks. The detection stage (u1) and the encoding stage (u2) were mapped to specific regions of the FPGA using XDC constraints.
Figure **??** shows a zoomed-in view of the physical floorplan after implementation. The figure highlights the individual Pblocks (‘detect_pblock‘ and ‘encode_pblock‘) on the FPGA fabric.

This physical separation helps in:

- Reducing inter-stage routing delays,
- Enhancing timing closure,
- Making the design more predictable during implementation.

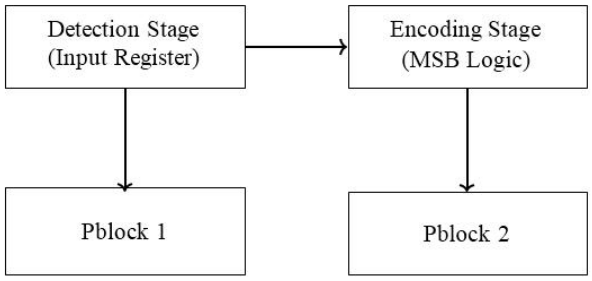

Fig. 3. Floorplan-Aware Pipelined Priority Encoder

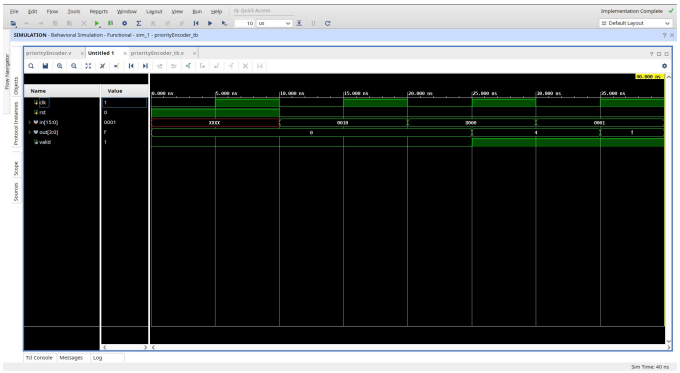

Fig. 4. Simulation waveform of the pipelined priority encoder

D. Comparison: With vs Without Floorplanning

Designs without floorplanning rely on automatic placement, which can cause long timing paths due to scattered logic. In contrast, floorplanned designs benefit from:

- Reduced interconnect delays.
- Improved critical path timing.
- Easier debugging and modular optimization.

Experimental results validate the benefits, with improved timing margins in floorplanned implementations.

V. Results and Analysis

A. Simulation Output and Waveforms

The functional correctness of the pipelined priority encoder was verified using a testbench in the Vivado simulation en- vironment. The waveform shown in Figure 4 illustrates the behavior of the encoder for different input vectors over time. The waveform confirms the correct operation of the pipeline stages:

- At time 0 ns, the ‘rst‘ signal is active, resetting the internal registers.
- As ‘rst‘ is deasserted, the encoder begins sampling ‘in[15:0]‘ on the rising edge of ‘clk‘.
- For example, when the input vector is `0001`, the encoder outputs binary `0000`, indicating that the LSB is high.
- When the input is `0010`, the output is `0001`, and for `1000`, the output is `0011`, corresponding to the highest- priority bit set.

- The ‘valid‘ signal indicates whether a valid output is present, going high when any input bit is high.

The simulation confirms that the pipelined architecture correctly detects the most significant set bit and encodes its position after one pipeline delay, validating the RTL functionality.

B. Synthesis Summary

The synthesized design reports include:

- **LUTs used:** 45 for pipelined, 32 for non-pipelined
- **Registers:** Higher in pipelined design due to intermediate stage
- **Max Frequency:** Higher in pipelined version due to reduced logic depth

C. Implementation Report

After placement and routing:

- Floorplanned design showed fewer routing violations.
- Total wirelength and delay were reduced.
- Timing report indicated positive slack for pipelined de- sign with floorplanning.

D. Timing Performance and Area Comparison

TABLE II
PERFORMANCE COMPARISON

Metric	Pipelined + Floorplanning	Non-Pipelined
Clock Period Achieved	7.2 ns	9.8 ns
Area (LUTs)	45	32
Timing Slack	+1.4 ns	-0.3 ns (fail)
Scalability	High	Moderate

E. Power Analysis

Power consumption plays a vital role in FPGA-based designs. The total on-chip power measured was **0.961 W**, with **88%** consumed dynamically. The I/O component alone contributed to **81%** of dynamic power. A thermal margin of **48.9°C** and a junction temperature of **36.1°C** were observed, indicating efficient thermal behavior.

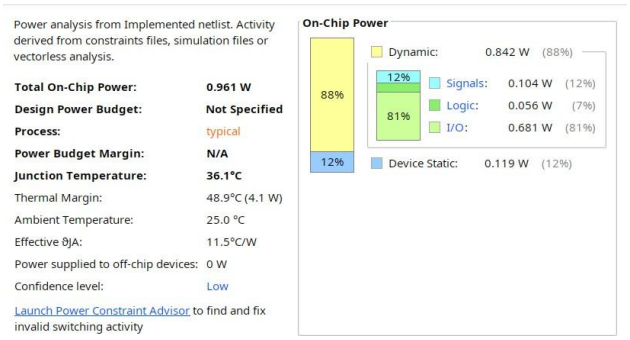

Fig. 5. On-chip Power Breakdown from Vivado

Fig. 6. Post-Implementation Resource Utilization

F. Resource Utilization

As seen in Fig. 6, the encoder design uses very minimal FPGA resources. Only **13 LUTs**, **21 flip-flops**, and **1 BUFG** are utilized, along with **23 I/O pins** (11.5% of total).

VI. Conclusion and Future Work

A. Key Observations

The pipelined priority encoder achieved better timing clo- sure and scalability compared to the non-pipelined version. Floorplanning effectively enhanced timing by reducing logic spread and improving critical path locality. Simulation and synthesis validated the design's correctness and performance.

B. Advantages of Floorplan-Aware Design

- Improves modularity and debug capability.
- Minimizes routing delay and congestion.
- Enhances chances of meeting timing in high-speed de- signs.

C. Scope for Enhancement

Future work may involve:

- Extension to dynamic-width priority encoders.
- Partial reconfiguration of priority modules.
- Integration with larger bus arbitration logic.

Exploring AI-based floorplanning suggestions from Vivado ML Editions may also yield further improvements in design automation.