DESIGN OF 12-BIT SAR ADC USING LOW POWER DYNAMIC DOUBLE TAIL COMPARATOR

Mani Ratnam T S, Thirumurugan , Siva Prakash S , D Sri Varshini ,Sneha Majumdar

***Abstract*—This paper outlines the design and execution of a 12-bit Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) that employs low-power dynamic double-tail comparators. The main aim of this design is to deliver high-resolution performance while minimizing power consumption, which is essential for applications with energy constraints, such as portable and battery-operated devices. The proposed SAR ADC capitalizes on the benefits of dynamic double-tail comparators, which facilitate lower power dissipation during the decision-making process, all while ensuring the necessary speed and accuracy for achieving 12-bit resolution. We present a comprehensive design of the comparator architecture, the SAR control logic, and the overall output of the ADC. Simulation results validate the design's capability to operate with accuracy. The design is realized in a 180nm CMOS process, and the outcomes indicate promising performance metrics, rendering it suitable for applications requiring high resolution and low power consumption.**

Keywords—SAR ADC, dynamic double-tail comparator, low power, 12-bit resolution, CMOS, analog-to-digital conversion

I. INTRODUCTION

In modern electronics, the need to accurately translate analog signals from the real world into digital data is essential for a wide variety of applications. These applications encompass wearable devices, such as health monitors and IoT sensors, as well as more conventional data acquisition systems. This process is made possible thanks to an Essential component—Analog to Digital Converter (ADC), which allows devices to recognize and interpret sounds, temperature or light as digital data that can be processed/almost anything. Without ADCs the functionality would be impossible for any device that is intended to perceive or react with the surrounding physical environment in a precise digital manner.

Analog-to-Digital Converters represent critical elements in digital systems where connectivity to the analog domain is achieved through the conversion of real-world signals into digital data. In recent years, there has been a growing demand for high-resolution, energy-efficient ADCs with the increased demand for IoT applications, new portable medical devices, and other electronic items running on batteries. Among various architectures of ADCs, the SAR is so valuable in providing moderate to high resolution with acceptable low power. Thus, this makes it very appropriate for the applications wherein accuracy is demanded with least energy utilization. The SAR-type ADC achieves this balance via the binary search algorithm. It iterates and converges to the digital output through a series of comparisons between the input signal and a reference voltage, where each bit is resolved in the process. This process allows SAR ADCs to maintain high accuracy and moderate sampling rates while using less power than other architectures, including flash or delta-sigma ADCs. A critical component in the SAR ADC design is the comparator that determines the accuracy and rate at which successive bit decisions are made in the conversion operation.

Out of numerous ADC designs, applications requiring efficiency along with moderate to high accuracy have started to select SAR ADCs for their needs. There are low-power SAR ADCs, which are characterized by low energy consumption and are often used in battery-operated or low-power applications, where every milliwatt is valuable. The analog-to-digital converters operate on the principle of quantization by employing a binary tree to determine the bits of the output signal in succession through the analog input and a standard signal. This process is enhanced by a digital-to-analog converter (DAC) that is used to provide the standard of the signal. The same general components that are common with a number of SAR ADC's, namely, the successive approximation register, the DAC, and the comparator. This process is governed by the SAR controller who determines each bit's temperature by specifying whether the analogue is greater or less than the point of reference provided by the DAC. It is the comparator that is always generating these comparisons, which is an integral part of the accuracy and the rate of the entire ADC interpretation.

SAR ADC makes use of the comparator considering it is the most critical component among all. It is required there that the input analog voltage is either above or below the reference voltage from the DAC which will respect the computation step so that the SAR ADC can determine the corresponding binary encoding for the input. Such a large number of decision making processes requires the comparator to work with relatively high speed and very low input offset so that there is no chance of error that might spoil the final digital output. As expected, sometimes it happens that the SAR ADC conversion speed is not less than the response time of the comparator. In that case the particular design of the comparator affects the completing characteristics of the SAR ADC. It is worth mentioning that for SAR ADCs designed for high resolution applications acquiring fast, accurate, and highly sensitive low power comparator is essential. Low power consumption is vital, especially in portable and battery-powered devices, where

ADCs are often operated at low sampling rates but still require high accuracy. Furthermore, the comparator must exhibit strong noise immunity to ensure that the final digital output is unaffected by electrical interference or signal fluctuations.
The comparator is responsible for deciding, at each step of binary search, if the input analog voltage is higher or lower than the reference voltage set by the digital-to-analog converter (DAC). The comparator's mistake, therefore, affects right at the source, the final accuracy of the ADC output as a pure digital quantity. Comparator design becomes very much a development focus in high resolution SAR ADCs. Traditional designs of comparators of this type, which include the single-tail dynamic comparators, have already gained extensive use in SAR ADCs for simple and power-hungry processes. In reality, however, with the rapid increase of numbers of elevated resolutions and higher sampling rates, traditional designs of these comparators are actually limited to speed, precision, and immunity against noise.

In this regard, the dual-tail comparator has surfaced as a promising candidate for high-resolution Successive Approximation Register Analog-to-Digital Converter applications. The dual tail architecture associates an auxiliary tail transistor with the comparator, which improves its speed and noise performance by allowing input and output stage isolation. This results in switching more speedily due to decreased sensitivity of the comparator's input stage to those fluctuations in the output node during switching.

The dual-tail structure further enhances the power efficiency of the comparator since the current drawn on each comparison is now as low as possible, especially since the devices are typically small enough, portable, and even battery-powered. All these factors make the dual-tail comparator particularly suitable for SAR ADCs in high-precision, low-power applications. Although this architecture offers many advantages, including a high conversion rate with minimum power consumption, implementing the dual-tail comparator in SAR ADCs is a complicated process that requires careful consideration of design parameters to optimize performance without increasing power. Optimizing the balance between speed, precision, and energy efficiency in a SAR ADC using a dual-tail comparator requires careful analysis and proper adjustment of the threshold voltage in the comparator, transistor sizing, and noise tolerance. Therefore, the performance of the comparator needs to be examined in the SAR ADC as a whole because design trade-offs for the comparator are going to impact the speed, resolution, and overall power characteristics of the ADC.

II. LITERATURE REVIEW:

Comparator is essential in SAR ADC architectures where comparisons are made accurately at each step of the binary search. In early designs, static and single-tail comparators are used because they are simple and consume less power. However, with the advancement of resolutions in successive approximation register ADCs, limitations on single-tail comparators become apparent. They tend to show slower responses and increased noise sensitivity. Both of these factors could pose problems for high-resolution SAR ADCs where precision is paramount. According to Razavi (2001), since these error signals directly passed on to the bit resolution in the SAR ADCs, such ADCs require the comparator design to be noise-resistant for fast operation in order to ensure performance in high-resolution applications.

The double-tail comparator topology appears promising for overcoming the effectiveness limitations of single-tail comparators in high resolution SAR ADCs. It was until 2007 that Verma and Chandrakasan introduced the design of a dual-tail comparator. The second tail transistor decouples the input and the output in a very effective manner, hence faster switching times and noise immunity through isolation of the input stage from the fluctuations likely to transpire in the output. Furthermore, dual-tail architecture low power is realized due to its reduction of current during each switching event and hence highly suited for low-power applications (Verma & Chandrakasan, 2007).

In the ensuing years, researchers have illustrated multiple optimizations of the dual-tail comparator to further improve its performance in Successive Approximation Register Analog-to-Digital Converter applications. For instance, Babayan-Mashhadi and Lotfi (2013) introduced a modified dual-tail comparator that incorporates regenerative feedback, thereby reducing the response time without leading to an increase in power consumption. This design enhancement was demonstrated to be advantageous for SAR ADCs by facilitating higher sampling rates while preserving accuracy in noisy environments, thus addressing a substantial limitation of earlier comparator designs (Babayan-Mashhadi & Lotfi, 2013). By the work of the authors, a fact is pointed out that, by proper modifications, the dual-tail comparator can be made to satisfy stringent requirements of high-resolution ADCs with power efficiency.More recent research is focused on lowering offset and power consumption for dual-tail comparators applied in ultra-low-power applications. Chen et al. (2018) further reduced offset voltage, a key parameter in the SAR ADC's accuracy, using a calibration technique designed for the dual-tail comparator. This calibration technique proved, in low-power IoT devices that offset voltage could be minimized without the need of increasing the power requirement of the comparator. Thus, this research just hints that the dual-tail comparator could meet the requirements of the IoT devices that require low power operation without compromising ADC accuracy according to Chen et al. 2018. Apart from the improvements in the comparator itself, numerous works have researched the overall impact of comparator topologies in complete SAR ADC architecture. Lin and Wu (2020) conducted a system-level analysis of a 12-bit SAR ADC equipped with a dual-tail comparator and demonstrated how the improved speed and noise immunity of the comparator will significantly impact the overall performance of the ADC. Their work establishes the significance of holistic design strategies that consider the inter-dependencies in the fundamental building blocks of a SAR ADC, such as the DAC, control logic, and the comparator (Lin & Wu, 2020).Overall, by summary, the dual-tail comparator has been established as one of the most

prominent advantages for high-resolution SAR ADCs with many studies featuring real speed, noise immunity, and efficiency compared to their single variants.

III. DUAL TAIL COMPARATOR:

Contrasted with traditional comparator architectures, which probably lack the prerequisite speed and noise resilience for accurate applications, the tail end-configuration of the two distinct "tail" transistors in the dual-tail comparator makes it achieve faster response times and better noise immunity functionality at operation with reduced power consumption, an indispensable requirement for high-performance SAR ADCs.Functionally, the dual tail comparator works in two different phases: the pre-charge phase and the evaluation phase. In the pre-charge phase, all internal nodes of the comparator are set up. Thus, the tail transistors are conditioned for the comparison process. Then, during the evaluation phase, these nodes switch according to the input signal, and a digital output is generated that may signify whether the input voltage is above or below the reference level. It preserves faster switching due to independently managing input and output stages, thus supporting the high sampling rates necessary for SAR ADCs for quick and accurate binary searches within each successive approximation cycle.

IV. METHODOLOGY:

In the paper, we employed Cadence software as the major design and simulation tool to develop a high resolution SAR ADC with a dual-tail comparator. A dual-tail comparator is a very important component in achieving accurate and power-efficient analog-to-digital conversion. The following are the major phases of the design methodology: setting up a CMOS cell library followed by design of both analogue and digital blocks until final verification of the output. Below is an outline of the methodology followed with a focus on the function and implementation of the dual-tail comparator in the SAR ADC structure.

1. Cadence Software as a Design Platform

The project settled to use Cadence software specifically for its broad ranges of design, simulation, and verification tools targeting CMOS devices. It supports a rich degree of customization as well, thereby allowing optimal variations of parameters with simulations carried out under different environmental conditions. Its simulation environment allowed performing high-level research into the performance of the dual-tail comparator in the context of the SAR ADC especially in terms of speed, power efficiency, and noise immunity.

The digital and analog design capabilities of Cadence made it an ideal platform to use in handling the complicated requirements of a mixed-signal design project like this.

2. Development of a CMOS Cell Library

A customized CMOS cell library was developed within Cadence for the construction of a powerful SAR ADC and comparator. The cell library contained elementary CMOS blocks that are fundamental for the building up of analog and digital designs. It included logic gates, switches, and current mirrors, all accurately modeled to achieve optimal performance within the target operational parameters of low power consumption and high switching speed. The cell library also comprised bespoke designs tailored just for the dual-tail comparator, optimized for minimum offset and maximum response time. This library served as the foundation for the design of both the SAR ADC and the dual-tail comparator, ensuring consistency and functionality across all parts of the design.

3.Analog Design Development

The analog design phase focused on the comparator's analog circuitry, DAC, and other analog components of the SAR ADC. In this stage, we implemented the dual-tail comparator's two-phase operation—pre-charge and evaluation phases—using CMOS technology. Cadence's analog simulation tools allowed for fine-tuning of the comparator's transistors and capacitors, optimizing the response time and minimizing power consumption while maintaining accuracy.

The layout of the analog design has been paid special attention to minimize parasitic effects, which is critical in high-resolution ADCs: it may introduce noise and adverse impacts upon the comparator performance. The analog components were simulated and refined through iterative procedures so as to ensure that the comparator met the stringent requirements of the SAR ADC with respect to accuracy of comparisons provided, with minimal delay.

4. Netlist for Digital Design

The digital circuitry part of the SAR ADC was conceptualized using a netlist design technique in Cadence. Thus, the resultant netlist consisted of the design parameters of digital elements controlling the binary search algorithm during the process of analog-to-digital conversion. It rendered an intricate illustration of the digital logic circuit, which in turn easily combined with its corresponding analog counterpart. We checked the digital elements under timing and functionality with the netlist and ensured that the SAR logic was synchronized with the dual-tail comparator. This is because locking the digital element to the analog components was necessary, as any mismatch in the system would have resulted in flawed accuracy and efficiency of the SAR ADC's conversion operations.

5. Verification and validation of outputs

After defining and integrating both analog and digital components, we performed rigorous verification of the SAR ADC's output. This was done by simulating various input signals and environmental conditions so as to evaluate the performance of the dual-tail comparator in various conditions. Cadence's verification environment was used to test some of the key metrics of interest in the ADC, such as ADC sampling rate, consumption power, and noise immunity. Special attention was paid to examination of comparison time and robustness of comparator, especially under noisy input and varying reference voltages conditions. Verification results confirmed our designs, that the dual-tail comparator meets our requirements: it produces

comparisons with a sufficient accuracy and efficiency, thus improving also the SAR ADC performance.

V.IMPLEMENTATION:

1.Hardware Description Language (HDL) Coding of SAR Logic using RTL Verilog:

To develop the digital logic for a 12-bit SAR ADC in Register Transfer Level (RTL) Verilog.

Successive Approximation Logic: Define the logic responsible for bitwise approximation, starting from the Most Significant Bit (MSB) to the Least Significant Bit (LSB).

RTL Verilog code for SAR logic was coded , implemented and verified using necessary testbench using Intel ModelSim.

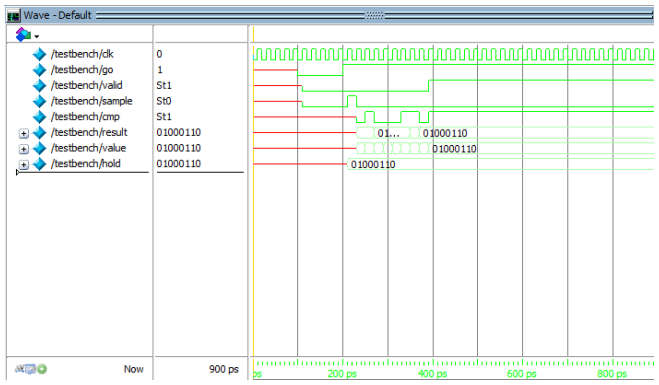

Figure 1: ModelSim Wave Window

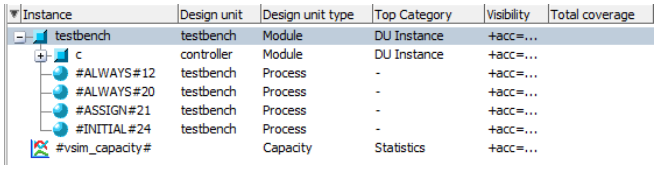

Figure 2: ModelSim Instances Window

Then the OSS CAD suite environment was set up for the gate level synthesis.

About Yosys:Yosys is a framework for Verilog RTL synthesis.Yosys is controlled using synthesis scripts. For example, the following Yosys synthesis script reads a design (with the top module mytop) from theVerilog file mydesign.v , synthesizes it to a gate-level netlist using the cell library in the Liberty file mycells.lib and writes the synthesized results as Verilog netlist to synth.v.

Then the standard cmos cell libraries contains the synthesized basic gates. Usingit , the RTL code of the SAR logic was synthesized.

```
C:\Users\Lenovo>cd C:\Users\Lenovo\Downloads\oss-cad-suite

C:\Users\Lenovo\Downloads\oss-cad-suite>environment.bat

[OSS CAD Suite] C:\Users\Lenovo\Downloads\oss-cad-suite>cd C:\Users\Lenovo\Downloads\yosys-main\examples\cmos

[OSS CAD Suite] C:\Users\Lenovo\Downloads\yosys-main\examples\cmos>yosys
```

Figure 3 : OSS CAD SUITE Environment

```
 /----------------------------------------------------------------------------\
 |  yosys -- Yosys Open SYnthesis Suite                                       |
 |  Copyright (C) 2012 - 2024  Claire Xenia Wolf <claire@yosyshq.com>         |
 |  Distributed under an ISC-like license, type "license" to see terms        |
 \----------------------------------------------------------------------------/
 Yosys 0.45+139 (git sha1 4d581a97d, x86_64-w64-mingw32-g++ 13.2.1 -O3)

yosys> read_verilog SAR_LOGIC.v

1. Executing Verilog-2005 frontend: SAR_LOGIC.v
Parsing Verilog input from `SAR_LOGIC.v' to AST representation.
Generating RTLIL representation for module `\controller'.
Successfully finished Verilog frontend.

yosys> hierarchy -check -top controller

2. Executing HIERARCHY pass (managing design hierarchy).

2.1. Analyzing design hierarchy..
Top module:  \controller

2.2. Analyzing design hierarchy..
Top module:  \controller
Removed 0 unused modules.

yosys> read_verilog -lib cmos_cells.v

3. Executing Verilog-2005 frontend: cmos_cells.v
Parsing Verilog input from `cmos_cells.v' to AST representation.
Generating RTLIL representation for module `\BUF'.
Generating RTLIL representation for module `\NOT'.
Generating RTLIL representation for module `\NAND'.
Generating RTLIL representation for module `\NOR'.
Generating RTLIL representation for module `\DFF'.
Generating RTLIL representation for module `\DFFSR'.
Successfully finished Verilog frontend.
```

Figure 4 : Vosys reading Verilog

USING SUPERMACRO SYNTH

yosys> synth

4. Executing SYNTH pass.

4.1. Executing HIERARCHY pass (managing design hierarchy).

4.1.1. Analyzing design hierarchy..
Top module: \controller

4.1.2. Analyzing design hierarchy..
Top module: \controller
Removed 0 unused modules.

4.2. Executing PROC pass (convert processes to netlists).

4.2.1. Executing PROC_CLEAN pass (remove empty switches from decision trees).
Cleaned up 0 empty switches.

4.2.2. Executing PROC_RMDEAD pass (remove dead branches from decision trees).
Marked 1 switch rules as full_case in process $proc$SAR_LOGIC.v:18$1 in module controller.
Removed a total of 0 dead cases.

4.2.3. Executing PROC_PRUNE pass (remove redundant assignments in processes).
Removed 0 redundant assignments.
Promoted 0 assignments to connections.

4.2.4. Executing PROC_INIT pass (extract init attributes).

4.2.5. Executing PROC_ARST pass (detect async resets in processes).

4.2.6. Executing PROC_ROM pass (convert switches to ROMs).
Converted 0 switches.
<suppressed ~4 debug messages>

4.2.7. Executing PROC_MUX pass (convert decision trees to multiplexers).
Creating decoders for process `\controller.$proc$SAR_LOGIC.v:18$1'.
1/3: $0\mask[7:0]
2/3: $0\state[1:0]
3/3: $0\result[7:0]

4.2.8. Executing PROC_DLATCH pass (convert process syncs to latches).

4.2.9. Executing PROC_DFF pass (convert process syncs to FFs).
Creating register for signal `\controller.\result' using process `\controller.$proc$SAR_LOGIC.v:18$1'.
created $dff cell `$procdff$31' with positive edge clock.
Creating register for signal `\controller.\state' using process `\controller.$proc$SAR_LOGIC.v:18$1'.
created $dff cell `$procdff$32' with positive edge clock.
Creating register for signal `\controller.\mask' using process `\controller.$proc$SAR_LOGIC.v:18$1'.
created $dff cell `$procdff$33' with positive edge clock.

4.2.10. Executing PROC_MEMWR pass (convert process memory writes to cells).

4.2.11. Executing PROC_CLEAN pass (remove empty switches from decision trees).
Found and cleaned up 4 empty switches in `\controller.$proc$SAR_LOGIC.v:18$1'.
Removing empty process `controller.$proc$SAR_LOGIC.v:18$1'.
Cleaned up 4 empty switches.

4.2.12. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.
<suppressed ~5 debug messages>

4.3. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.4. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 1 unused cells and 13 unused wires.
<suppressed ~2 debug messages>

4.5. Executing CHECK pass (checking for obvious problems).
Checking module controller...
Found and reported 0 problems.

4.6. Executing OPT pass (performing simple optimizations).

4.6.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.6.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
<suppressed ~18 debug messages>
Removed a total of 6 cells.

4.6.3. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
Creating internal representation of mux trees.
Evaluating internal representation of mux trees.
Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~3 debug messages>

4.6.4. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
Optimizing cells in module \controller.
Performed a total of 0 changes.

4.6.5. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.6.6. Executing OPT_DFF pass (perform DFF optimizations).

4.6.7. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 0 unused cells and 6 unused wires.
<suppressed ~1 debug messages>

4.6.8. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.6.9. Rerunning OPT passes. (Maybe there is more to do..)

4.6.10. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
Creating internal representation of mux trees.
Evaluating internal representation of mux trees.
Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~3 debug messages>

4.6.11. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
Optimizing cells in module \controller.

Performed a total of 0 changes.

4.6.12. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.6.13. Executing OPT_DFF pass (perform DFF optimizations).

4.6.14. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.6.15. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.6.16. Finished OPT passes. (There is nothing left to do.)

4.7. Executing FSM pass (extract and optimize FSM).

4.7.1. Executing FSM_DETECT pass (finding FSMs in design).
Found FSM state register controller.state.

4.7.2. Executing FSM_EXTRACT pass (extracting FSM from design).
Extracting FSM `\state' from module `\controller'.
found $dff cell for state register: $procdff$32
root of input selection tree: $0\state[1:0]
found reset state: 2'00 (guessed from mux tree)
found ctrl input: \go
found state code: 2'00
found ctrl input: $procmux$17_CMP
found ctrl input: \sample
found ctrl input: $procmux$19_CMP
found ctrl input: \mask [0]
found state code: 2'11
found state code: 2'10
found state code: 2'01
found ctrl output: \sample
found ctrl output: \valid
found ctrl output: $procmux$17_CMP
found ctrl output: $procmux$19_CMP
ctrl inputs: { \mask [0] \go }
ctrl outputs: { $procmux$19_CMP $procmux$17_CMP $0\state[1:0] \sample \valid }
transition: 2'00 2'-0 -> 2'00 6'100000
transition: 2'00 2'-1 -> 2'01 6'100100
transition: 2'10 2'-0 -> 2'00 6'010000
transition: 2'10 2'01 -> 2'10 6'011000
transition: 2'10 2'11 -> 2'11 6'011100
transition: 2'01 2'-0 -> 2'00 6'000010
transition: 2'01 2'-1 -> 2'10 6'001010
transition: 2'11 2'-0 -> 2'00 6'000001
transition: 2'11 2'-1 -> 2'11 6'001101

4.7.3. Executing FSM_OPT pass (simple optimizations of FSMs).
Optimizing FSM `$fsm$\state$34' from module `\controller'.

4.7.4. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 8 unused cells and 8 unused wires.
<suppressed ~9 debug messages>

4.7.5. Executing FSM_OPT pass (simple optimizations of FSMs).
Optimizing FSM `$fsm$\state$34' from module `\controller'.
Removing unused output signal $0\state[1:0] [0].
Removing unused output signal $0\state[1:0] [1].
Removing unused output signal $procmux$19_CMP.

4.7.6. Executing FSM_RECODE pass (re-assigning FSM state encoding).
Recoding FSM `$fsm$\state$34' from module `\controller' using `auto' encoding:
mapping auto encoding to `one-hot` for this FSM.
00 -> ---1
10 -> --1-
01 -> -1--
11 -> 1---

4.7.7. Executing FSM_INFO pass (dumping all available information on FSM cells).

FSM `$fsm$\state$34' from module `controller':

Information on FSM fsm\state$34 (\state):

Number of input signals: 2
Number of output signals: 3
Number of state bits: 4

Input signals:
0: \go
1: \mask [0]

Output signals:
0: \valid
1: \sample
2: $procmux$17_CMP

State encoding:
0: 4'---1 <RESET STATE>
1: 4'--1-
2: 4'-1--
3: 4'1---

Transition Table (state_in, ctrl_in, state_out, ctrl_out):
0: 0 2'-0 -> 0 3'000
1: 0 2'-1 -> 2 3'000
2: 1 2'-0 -> 0 3'100
3: 1 2'01 -> 1 3'100
4: 1 2'11 -> 3 3'100
5: 2 2'-0 -> 0 3'010
6: 2 2'-1 -> 1 3'010
7: 3 2'-0 -> 0 3'001
8: 3 2'-1 -> 3 3'001

4.7.8. Executing FSM_MAP pass (mapping FSMs to basic logic).
Mapping FSM `$fsm$\state$34' from module `\controller'.

4.8. Executing OPT pass (performing simple optimizations).

4.8.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.
<suppressed ~4 debug messages>

4.8.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.8.3. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
  Creating internal representation of mux trees.
  Evaluating internal representation of mux trees.
  Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~2 debug messages>

4.8.4. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \controller.
Performed a total of 0 changes.

4.8.5. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.8.6. Executing OPT_DFF pass (perform DFF optimizations).
Adding EN signal on $procdff$33 ($dff) from module controller (D = $procmux$8_Y, Q = \mask).
Adding EN signal on $procdff$31 ($dff) from module controller (D = $procmux$25_Y, Q = \result).

4.8.7. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 2 unused cells and 11 unused wires.
<suppressed ~3 debug messages>

4.8.8. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.
<suppressed ~2 debug messages>

4.8.9. Rerunning OPT passes. (Maybe there is more to do..)

4.8.10. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
  Creating internal representation of mux trees.
  Evaluating internal representation of mux trees.
  Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~2 debug messages>

4.8.11. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \controller.
Performed a total of 0 changes.

4.8.12. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
<suppressed ~3 debug messages>
Removed a total of 1 cells.

4.8.13. Executing OPT_DFF pass (perform DFF optimizations).

4.8.14. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 0 unused cells and 1 unused wires.
<suppressed ~1 debug messages>

4.8.15. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.8.16. Rerunning OPT passes. (Maybe there is more to do..)

4.8.17. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
  Creating internal representation of mux trees.
  Evaluating internal representation of mux trees.
  Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~2 debug messages>

4.8.18. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \controller.
Performed a total of 0 changes.

4.8.19. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.8.20. Executing OPT_DFF pass (perform DFF optimizations).

4.8.21. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.8.22. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.8.23. Finished OPT passes. (There is nothing left to do.)

4.9. Executing WREDUCE pass (reducing word size of cells).
Removed top 1 bits (of 2) from port B of cell controller.$auto$fsm_map.cc:77:implement_pattern_cache$47 ($eq).

4.10. Executing PEEPOPT pass (run peephole optimizers).

4.11. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.12. Executing ALUMACC pass (create $alu and $macc cells).
Extracting $alu and $macc cells in module controller:
  created 0 $alu and 0 $macc cells.

4.13. Executing SHARE pass (SAT-based resource sharing).

4.14. Executing OPT pass (performing simple optimizations).

4.14.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.14.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.14.3. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
  Creating internal representation of mux trees.
  Evaluating internal representation of mux trees.
  Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~2 debug messages>

4.14.4. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \controller.
Performed a total of 0 changes.

4.14.5. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.14.6. Executing OPT_DFF pass (perform DFF optimizations).

4.14.7. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.14.8. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.14.9. Finished OPT passes. (There is nothing left to do.)

4.15. Executing MEMORY pass.

4.15.1. Executing OPT_MEM pass (optimize memories).
Performed a total of 0 transformations.

4.15.2. Executing OPT_MEM_PRIORITY pass (removing unnecessary memory write priority relations).
Performed a total of 0 transformations.

4.15.3. Executing OPT_MEM_FEEDBACK pass (finding memory read-to-write feedback paths).

4.15.4. Executing MEMORY_BMUX2ROM pass (converting muxes to ROMs).

4.15.5. Executing MEMORY_DFF pass (merging $dff cells to $memrd).

4.15.6. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.15.7. Executing MEMORY_SHARE pass (consolidating $memrd/$memwr cells).

4.15.8. Executing OPT_MEM_WIDEN pass (optimize memories where all ports are wide).
Performed a total of 0 transformations.

4.15.9. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.15.10. Executing MEMORY_COLLECT pass (generating $mem cells).

4.16. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.17. Executing OPT pass (performing simple optimizations).

4.17.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.
<suppressed ~3 debug messages>

4.17.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.17.3. Executing OPT_DFF pass (perform DFF optimizations).
Adding SRST signal on $auto$ff.cc:266:slice$68 ($dffe) from module controller (D = \mask [7:1], Q = \mask [6:0], rval = 7'0000000).
Adding SRST signal on $auto$ff.cc:266:slice$73 ($dffe) from module controller (D = $procmux$23_Y, Q = \result, rval = 8'00000000).

4.17.4. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 1 unused cells and 2 unused wires.
<suppressed ~2 debug messages>

4.17.5. Rerunning OPT passes. (Removed registers in this run.)

4.17.6. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.17.7. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.17.8. Executing OPT_DFF pass (perform DFF optimizations).

4.17.9. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.17.10. Finished fast OPT passes.

4.18. Executing MEMORY_MAP pass (converting memories to logic and flip-flops).

4.19. Executing OPT pass (performing simple optimizations).

4.19.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.19.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.19.3. Executing OPT_MUXTREE pass (detect dead branches in mux trees).
Running muxtree optimizer on module \controller..
  Creating internal representation of mux trees.
  Evaluating internal representation of mux trees.
  Analyzing evaluation results.
Removed 0 multiplexer ports.
<suppressed ~1 debug messages>

4.19.4. Executing OPT_REDUCE pass (consolidate $*mux and $reduce_* inputs).
  Optimizing cells in module \controller.
Performed a total of 0 changes.

4.19.5. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.19.6. Executing OPT_SHARE pass.

4.19.7. Executing OPT_DFF pass (perform DFF optimizations).

4.19.8. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..

4.19.9. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.19.10. Finished OPT passes. (There is nothing left to do.)

4.20. Executing TECHMAP pass (map to technology primitives).

4.20.1. Executing Verilog-2005 frontend: C:\Users\Lenovo\DOWNLO~1\OSS-CA~1\bin\../share/yosys/techmap.v
Parsing Verilog input from `C:\Users\Lenovo\DOWNLO~1\OSS-CA~1\bin\../share/yosys/techmap.v' to AST representation.
Generating RTLIL representation for module `_90_simplemap_bool_ops'.
Generating RTLIL representation for module `_90_simplemap_reduce_ops'.
Generating RTLIL representation for module `_90_simplemap_logic_ops'.
Generating RTLIL representation for module `_90_simplemap_compare_ops'.
Generating RTLIL representation for module `_90_simplemap_various'.
Generating RTLIL representation for module `_90_simplemap_registers'.
Generating RTLIL representation for module `_90_shift_ops_shr_shl_sshl_sshr'.
Generating RTLIL representation for module `_90_shift_shiftx'.
Generating RTLIL representation for module `_90_fa'.
Generating RTLIL representation for module `_90_lcu_brent_kung'.
Generating RTLIL representation for module `_90_alu'.
Generating RTLIL representation for module `_90_macc'.
Generating RTLIL representation for module `_90_alumacc'.
Generating RTLIL representation for module `\$__div_mod_u'.
Generating RTLIL representation for module `\$__div_mod_trunc'.
Generating RTLIL representation for module `_90_div'.
Generating RTLIL representation for module `_90_mod'.

Generating RTLIL representation for module `\$__div_mod_floor'.
Generating RTLIL representation for module `_90_divfloor'.
Generating RTLIL representation for module `_90_modfloor'.
Generating RTLIL representation for module `_90_pow'.
Generating RTLIL representation for module `_90_pmux'.
Generating RTLIL representation for module `_90_demux'.
Generating RTLIL representation for module `_90_lut'.
Successfully finished Verilog frontend.

4.20.2. Continuing TECHMAP pass.
Using extmapper simplemap for cells of type $reduce_and.
Using extmapper simplemap for cells of type $or.
Using extmapper simplemap for cells of type $reduce_or.
Using extmapper simplemap for cells of type $dffe.
Using extmapper simplemap for cells of type $eq.
Using extmapper simplemap for cells of type $ne.
Using extmapper simplemap for cells of type $and.
Using extmapper simplemap for cells of type $mux.
Using extmapper simplemap for cells of type $not.
Using extmapper simplemap for cells of type $reduce_bool.
Using extmapper simplemap for cells of type $sdffce.
Using extmapper simplemap for cells of type $dff.
No more expansions possible.
<suppressed ~93 debug messages>

4.21. Executing OPT pass (performing simple optimizations).

4.21.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.
<suppressed ~6 debug messages>

4.21.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
<suppressed ~9 debug messages>
Removed a total of 3 cells.

4.21.3. Executing OPT_DFF pass (perform DFF optimizations).

4.21.4. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 7 unused cells and 14 unused wires.
<suppressed ~8 debug messages>

4.21.5. Finished fast OPT passes.

4.22. Executing ABC pass (technology mapping using ABC).

4.22.1. Extracting gate netlist of module `\controller' to `<abc-temp-dir>/input.blif'..
Extracted 27 gates and 49 wires to a netlist network with 22 inputs and 15 outputs.

4.22.1.1. Executing ABC.
Running ABC command: "<yosys-exe-dir>/yosys-abc" -s -f <abc-temp-dir>/abc.script 2>&1
ABC: ABC command line: "source <abc-temp-dir>/abc.script".
ABC:
ABC: + read_blif <abc-temp-dir>/input.blif
ABC: + read_library <abc-temp-dir>/stdcells.genlib
ABC: Entered genlib library with 13 gates from file "<abc-temp-dir>/stdcells.genlib".
ABC: + strash
ABC: + dretime
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif

4.22.1.2. Re-integrating ABC results.
ABC RESULTS: NOT cells: 2
ABC RESULTS: ANDNOT cells: 1
ABC RESULTS: NAND cells: 3
ABC RESULTS: NOR cells: 5
ABC RESULTS: ORNOT cells: 4
ABC RESULTS: OR cells: 8
ABC RESULTS: AND cells: 1
ABC RESULTS: internal signals: 12
ABC RESULTS: input signals: 22
ABC RESULTS: output signals: 15
Removing temp directory.

4.23. Executing OPT pass (performing simple optimizations).

4.23.1. Executing OPT_EXPR pass (perform const folding).
Optimizing module controller.

4.23.2. Executing OPT_MERGE pass (detect identical cells).
Finding identical cells in module `\controller'.
Removed a total of 0 cells.

4.23.3. Executing OPT_DFF pass (perform DFF optimizations).

4.23.4. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 0 unused cells and 47 unused wires.
<suppressed ~1 debug messages>

4.23.5. Finished fast OPT passes.

4.24. Executing HIERARCHY pass (managing design hierarchy).

4.24.1. Analyzing design hierarchy..
Top module: \controller

4.24.2. Analyzing design hierarchy..
Top module: \controller
Removed 0 unused modules.

```
4.25. Printing statistics.

=== controller ===

   Number of wires:                 25
   Number of wire bits:             49
   Number of public wires:           9
   Number of public wire bits:      33
   Number of ports:                  7
   Number of port bits:             21
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 44
     $_ANDNOT_                       1
     $_AND_                          1
     $_DFFE_PP_                      1
     $_DFF_P_                        4
     $_NAND_                         3
     $_NOR_                          5
     $_NOT_                          2
     $_ORNOT_                        4
     $_OR_                           8
     $_SDFFCE_PN0P_                 15

4.26. Executing CHECK pass (checking for obvious problems).
Checking module controller...
Found and reported 0 problems.
```

Figure 5: statistics

Mapping the components of netlist to the flip flops of cmos cells library.

```
5. Executing DFFLIBMAP pass (mapping DFF cells to sequential cells from liberty file).
  cell DFF (noninv, pins=3, area=18.00) is a direct match for cell type $_DFF_P_.
  cell DFFSR (noninv, pins=5, area=18.00) is a direct match for cell type $_DFFSR_PPP_.
  final dff cell mappings:
    unmapped dff cell: $_DFF_N_
    \DFF _DFF_P_ (.C( C), .D( D), .Q( Q));
    unmapped dff cell: $_DFF_NN0_
    unmapped dff cell: $_DFF_NN1_
    unmapped dff cell: $_DFF_NP0_
    unmapped dff cell: $_DFF_NP1_
    unmapped dff cell: $_DFF_PN0_
    unmapped dff cell: $_DFF_PN1_
    unmapped dff cell: $_DFF_PP0_
    unmapped dff cell: $_DFF_PP1_
    unmapped dff cell: $_DFFSR_NNN_
    unmapped dff cell: $_DFFSR_NNP_
    unmapped dff cell: $_DFFSR_NPN_
    unmapped dff cell: $_DFFSR_NPP_
    unmapped dff cell: $_DFFSR_PNN_
    unmapped dff cell: $_DFFSR_PNP_
    unmapped dff cell: $_DFFSR_PPN_
    \DFFSR _DFFSR_PPP_ (.C( C), .D( D), .Q( Q), .R( R), .S( S));

5.1. Executing DFFLEGALIZE pass (convert FFs to types supported by the target).
Mapping DFF cells in module `\controller':
  mapped 20 $_DFF_P_ cells to \DFF cells.
```

Figure 6: Flip flop mapping

```
6.1.2. Re-integrating ABC results.
ABC RESULTS:               NOT cells:       22
ABC RESULTS:               NOR cells:       11
ABC RESULTS:              NAND cells:       59
ABC RESULTS:        internal signals:       28
ABC RESULTS:           input signals:       22
ABC RESULTS:          output signals:       28
```

Figure 7: Logic cells mapping

USING SUPER MACRO OPT-CLEAN

```
yosys> opt_clean

7. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \controller..
Removed 0 unused cells and 77 unused wires.
<suppressed ~1 debug messages>
```

Figure 8: Cleaning the netlist

Dumping the netlist onto .v file

```
yosys> write_verilog Sar_synth.v

8. Executing Verilog backend.

8.1. Executing BMUXMAP pass.

8.2. Executing DEMUXMAP pass.
Dumping module `\controller'.
```

Figure 9: Writing the Synthesized Netlist

2. Capacitive Digital-to-Analog Converter (C-DAC)
Design of the Capacitor Array and Switches

A. Capacitor Array Design Strategy
In this 12-bit SAR ADC, a two-stage weighted capacitor array is utilized instead of a fully parallel arrangement. The design divides the array into two parts: an 8-bit Least Significant Bit (LSB) section and a 4-bit Most Significant Bit (MSB) section. To achieve a compa
ct design, a split capacitor, denoted as C split is introduced into the array. This split capacitor reduces the overall capacitance size, helping prevent fabrication issues that arise from extremely small LSB capacitors. Without the split capacitor, the LSB capacitance would be too low, challenging the limits of fabrication accuracy and reliability.

B. DAC Switches Logic
Transmission gates are employed in the DAC to minimize charge injection, a phenomenon that can lead to inaccuracies in analog output. Since NMOS and PMOS transistors have complementary polarities, the positive charge in the NMOS transistor can offset the negative charge in the PMOS transistor, reducing unwanted charge accumulation. This complementary structure effectively mitigates charge injection.

When determining the sizing strategy for the DAC switches, the objective is to balance the resistance Ron across all switches. However, setting all switches to the same Ron can introduce transients when switching. To address this, switch sizes are adjusted in proportion to the capacitance values they control. By sizing switches based on their corresponding capacitors, the design maintains stable operation and minimizes the risk of transient effects during switching events.

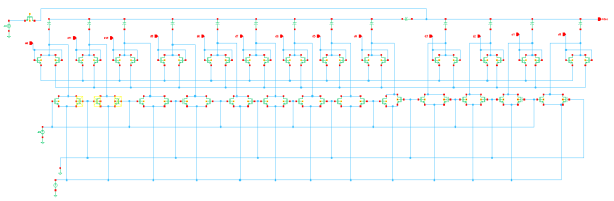

Figure 10: DAC switches

The capacitor values used for the CDAC array:

	Capacitor	Size(pF)
	Cd	1/128
LSB	C12	1/128
LSB	C11	1/64
LSB	C10	1/32
LSB	C9	1/16
LSB	C8	1/8
LSB	C7	1/4
LSB	C6	1/2
LSB	C5	1
	C split	130fF
MSB	C4	1/8
MSB	C3	1/4
MSB	C2	1/2
MSB	C1	1
	Ctotal	2

The sizing used for CDAC array:

	Capacitor	Size
	Bit D	1 unit
LSB	Bit 12	1unit
LSB	Bit 11	2 units
LSB	Bit 10	4 units
LSB	Bit 9	8 units
LSB	Bit 8	16 units
LSB	Bit 7	32 units
LSB	Bit 6	64 units
LSB	Bit 5	128 units
MSB	Bit 4	16 units
MSB	Bit 3	32 units
MSB	Bit 2	64 units
MSB	Bit 1	128 units

After constructing the DAC, the next step is to set up a testbench to verify its functionality before moving on to the SAR logic and comparator stages. This testbench is designed to confirm that the DAC accurately converts digital input signals to the expected analog output range. The approach involves applying a pulse signal to each input pin sequentially, with staggered delays, ensuring that all possible digital input combinations from binary 000000000000 (all bits low) to 111111111111 (all bits high) are tested.

For the test inputs, a "vpulse" signal is applied to each bit input, simulating different digital values over time. The period for each "vpulse" signal decreases progressively from bit 1 (the MSB) to bit 12 (the LSB). For bit 1, the pulse period is 4096 * tCLK, for bit 2 it is 2048 * tCLK, and so on, with each subsequent bit's pulse period halved until bit 12, which has a pulse period of 2 * tCLK. Here, tCLK represents the clock period, which is set to 0.6 seconds in this configuration.

This staged delay approach ensures a full sweep of all possible digital input codes to test the DAC's response across its entire input range, allowing for accurate assessment of the DAC's performance and proper calibration before integrating the SAR ADC's remaining components.

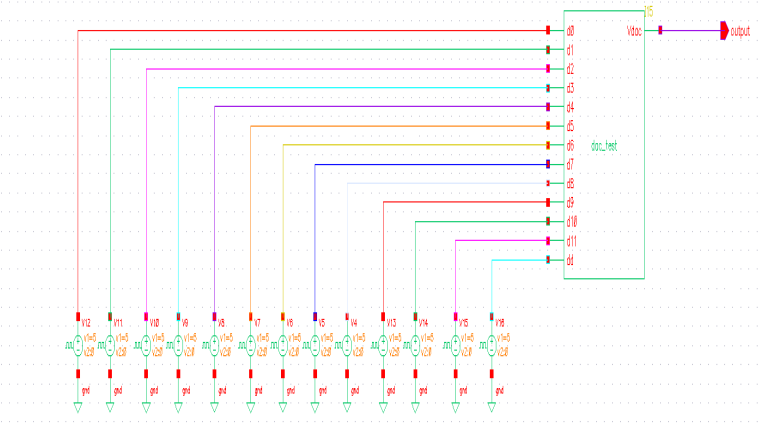

Figure 11: DAC TEST

3.Design of Double Tail Dynamic latched Comparator:

Select Double-Tail Configuration

- Opt for the double-tail design, which facilitates operation at reduced supply voltages and contributes to lower power consumption. This configuration features distinct tails for both the input and latch stages, allowing for optimized current management and rapid performance.

Design the Input Stage

- Create the input differential pair (either NMOS or PMOS) to generate the initial differential voltage. This stage is responsible for pre-charging nodes to a predetermined voltage level during the reset phase. The tail current should be selected based on the desired input impedance and integration duration.

Set Up Intermediate and Latch Stages

- In the intermediate stage, arrange cross-coupled inverters and current sources to achieve high gain for swift regeneration. Choose appropriate transistor sizes in the latch stage to ensure robust positive feedback, facilitating quick amplification of the input difference.

Add Control Transistors

- Integrate cross-coupled control transistors (for instance, Mc1 and Mc2 in a double-tail configuration) to enhance the regeneration speed. These transistors are instrumental in producing a larger differential voltage by channeling more current into one path based on the input difference, thereby improving latch speed.

Utilize Tail Transistors

- Implement two tail transistors (one designated for the input and another for the latch) to enable independent current control in each section. This separation reduces reliance on the common-mode voltage and enhances power efficiency.

Conduct Simulation

- Perform simulations of the comparator to evaluate the output.

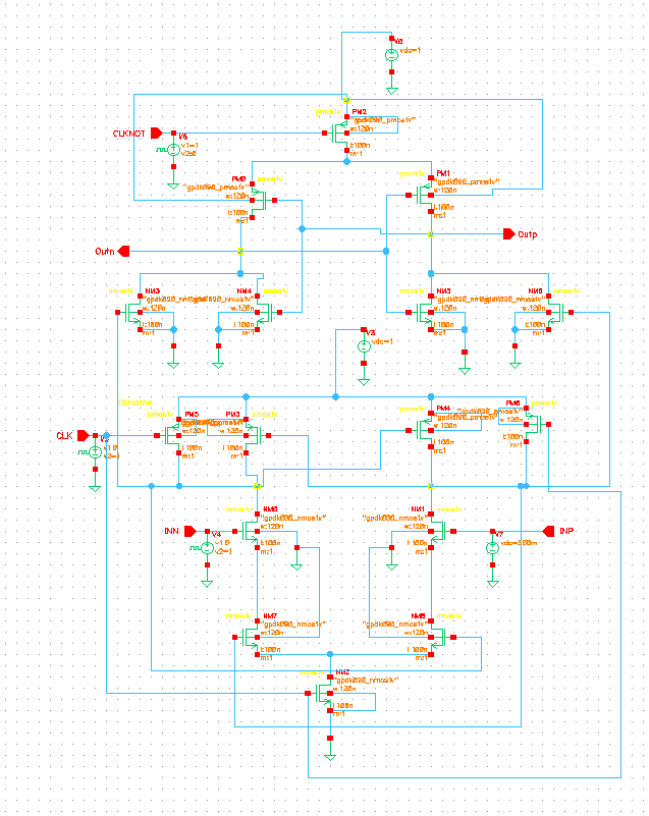

Figure 12: Double Tail Dynamic latched Comparator

The designed double tail dynamic latched comparator circuit was simulated and verified.

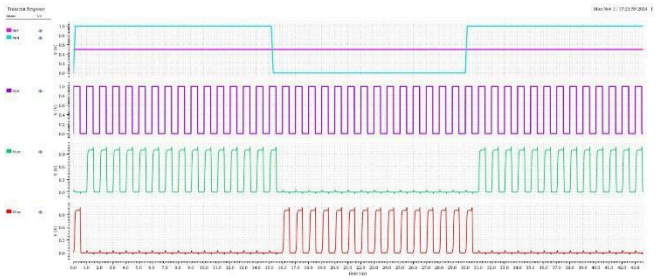

Figure 13: Simulation for double tail dynamic latched comparator

3. Combining Individual blocks:

In this schematic of a 12-bit Successive Approximation Register (SAR) ADC circuit, we see three main blocks: the Comparator, the Controller (SAR logic), and the CDAC.Given below is a detailed breakdown of the connections between each block:

- Comparator:

- The comparator has two input pins labeled `INN` (inverting input) and `INP` (non-inverting input), which receive the differential input signal from an external source (like a sensor or another analog source). `V1` represents this input source, connected to ground as well.

- The comparator output pins `Outn` and `Outp` produce differential digital outputs that represent whether the input signal is above or below a reference threshold.

- These output pins are connected to the `cmp` input on the Controller block. This connection allows the controller to assess the comparator output and proceed with the binary search algorithm for conversion.

- Controller:

- The controller block receives inputs such as `clk`, `go`, and `cmp`.

- `clk` is the clock signal that controls the timing of the conversion process.

- `go` is a control signal to initiate the conversion process in the SAR ADC.

- `cmp` is the comparator output, used by the controller to make successive approximation decisions.

- The controller outputs the `result<7:0>` (digital conversion result), `valid` (indicating the result is ready), and `sample` signals. The `sample` signal is connected to the DAC, allowing the DAC to update its reference voltage based on the controller's feedback.

- `result<7:0>` provides the digital output, likely representing the SAR ADC's conversion results for the connected digital circuit or microcontroller.

3. CDAC (Capacitance Digital-to-Analog Converter):

- The DAC block converts the digital control signals from the Controller into an analog reference voltage.

- It has several input pins (`d0` to `d11`) that connect to the controller's output, representing the bit positions of the digital code generated during the SAR conversion process.

- The DAC also has `AVdd` (analog supply voltage) and `AGnd` (analog ground) connections for stable operation.

- The DAC output is fed back to the `INN` and `INP` inputs of the comparator, enabling it to compare the DAC's output with the input signal iteratively.

- `Vdd` and `Vref` are also connected to the DAC to set the reference voltage range for accurate conversions.

The connection diagram of the whole completed ADC is given below.

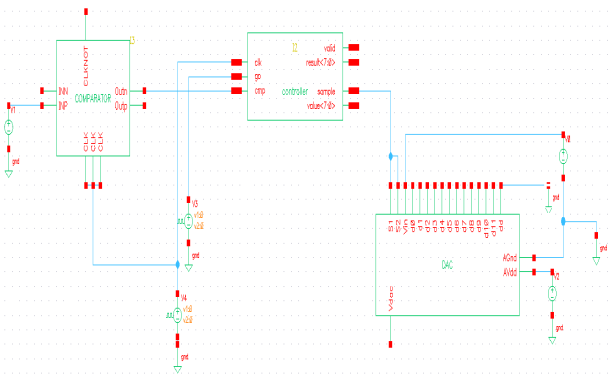

Figure 14: Final ADC circuit

4. Final output of SAR ADC:

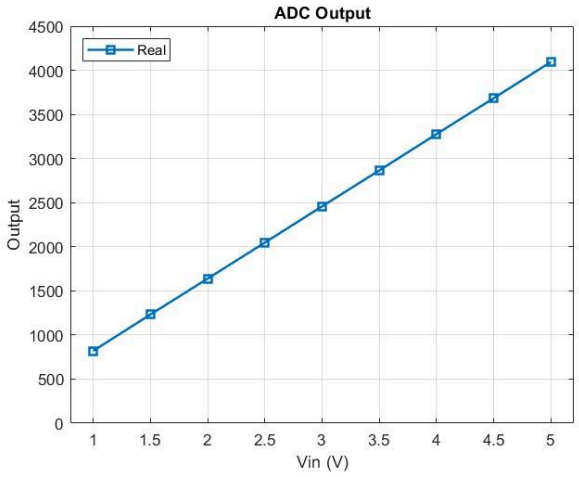

Figure 15 ADC output

Vin (V)	Output
1	820
1.5	1232
2	1639
2.5	2047
3	2458
3.5	2865
4	3277
4.5	3684
5	4095

Figure 16 values of vin and output

The ADC Output values in the table and graph exhibit a linear relationship with the input voltage (Vin). This consistency indicates that the ADC effectively converts analog signals into digital values with minimal deviation, closely matching the expected behavior. The graph visually confirms this trend, showing a smooth and proportional increase in output as the input voltage rises.

Conclusion

The design of the 12-bit Successive Approximation Register (SAR) ADC integrates several key components, each optimized for efficiency, precision, and power consumption.
The Capacitive Digital-to-Analog Converter (C-DAC) utilizes a two-stage weighted capacitor array, divided into an 8-bit Least Significant Bit (LSB) section and a 4-bit Most Significant Bit (MSB) section. The introduction of the split capacitor mitigates fabrication challenges associated with small capacitance values in the LSB section, ensuring both accuracy and reliability.
For the DAC switches, the design incorporates transmission gates, which effectively reduce charge injection and associated inaccuracies. The sizing of the switches is carefully managed to balance resistance and prevent transients during switching events, thus maintaining stable operation.
The double-tail dynamic latched comparator is selected for its ability to operate efficiently at low supply voltages while reducing power consumption. The use of cross-coupled transistors enhances the comparator's regeneration speed, ensuring fast and accurate conversion decisions.
The final SAR ADC integrates the comparator, controller (SAR logic), and C-DAC to complete the analog-to-digital conversion process. The comparator compares the input signal against the DAC's output, and the controller manages the conversion cycle, providing the digital result. The DAC receives feedback from the controller and iteratively adjusts its output during the conversion process.
Through extensive simulation and testing, the system is verified to exhibit a linear relationship between the input voltage and the output digital code, demonstrating the ADC's accuracy and reliability. The design ensures minimal deviation from expected behavior, confirming that the ADC performs its function of converting analog signals to digital outputs with high precision.
In conclusion, the proposed SAR ADC design leverages well-optimized sub-circuits and careful component sizing to deliver a compact, efficient, and accurate solution for analog-to-digital conversion. The design's successful integration and testing pave the way for further advancements in high-resolution ADC systems.

APPENDIX

RTL VERILOG CODE

```
// ADC controller
module controller(clk,go,valid,result, sample,value,cmp);
   input clk; // clock input
   input go; // go=1 to perform conversion
   output valid; // valid=1 when conversion finished
   output [7:0] result; // 8-bit result output
   output sample; // to S&H circuit
   output [7:0] value; // to DAC
   input cmp; // from comparator
   reg [1:0] state; // current state in state machine
   reg [7:0] mask; // bit to test in binary search
   reg [7:0] result; // hold partially converted result

   // state assignment
   parameter sWait=0, sSample=1, sConv=2, sDone=3;

   // synchronous design
   always @(posedge clk) begin
      if (!go)
         state <= sWait; // stop and reset if go=0
      else
         case (state) // choose next state in state machine
            sWait : state <= sSample;
            sSample : begin // start new conversion
               state <= sConv; // enter convert state next
                    mask <= 8'b10000000; // reset mask to MSB
only
               result <= 8'b0; // clear result
            end
            sConv : begin
                  // set bit if comparator indicates input larger
than value currently under consideration
               if (cmp) result <= result | mask;
               // shift mask to try next bit next time
               mask <= mask >> 1;
               // finished once LSB has been done
               if (mask[0]) state <= sDone;
            end
            sDone :;
         endcase
   end

    assign sample = (state == sSample); // drive sample and
hold
    assign value = result | mask; // (result so far) OR (bit to
try)
   assign valid = (state == sDone); // indicate when finished
endmodule
```

RTL VERILOG TESTBENCH

```
module testbench();
    // Registers to hold inputs to circuit under test, wires for
outputs
   reg clk, go;
   wire valid, sample, cmp;
   wire [7:0] result;
   wire [7:0] value;

   // Instantiate the controller circuit
   controller c(clk, go, valid, result, sample, value, cmp);

   // Generate a clock with period of 20 time units
   always begin
      #10;
      clk = ~clk;
   end
   initial clk = 0;

   // Simulate analogue circuit with a digital model
   reg [7:0] hold;
   always @(posedge sample) hold = 8'b01000110;
   assign cmp = (hold >= value);

   // Monitor some signals and provide input stimuli
   initial begin
            $monitor($time, " go=%b valid=%b result=%b
sample=%b value=%b cmp=%b state=%b mask=%b",
                  go, valid, result, sample, value, cmp, c.state,
c.mask);
      #100; go = 0;
      #100; go = 1;
      #5000; go = 0;
      #5000; go = 1;
      #40; go = 0;
      #5000;
      $stop;
   end
endmodule
```

GATE LEVEL NETLIST

```
/* Generated by Yosys 0.45+139 (git sha1 4d581a97d, x86_64-w64-mingw32-g++ 13.2.1 -O3) */

(* top =  1  *)
(* src = "SAR_LOGIC.v:2.1-44.10" *)
module controller(clk, go, valid, result, sample, value, cmp);
  wire _000_;
  wire _001_;
  wire _002_;
  wire _003_;
  wire _004_;
  wire _005_;
  wire _006_;
  wire _007_;
  wire _008_;
  wire _009_;
  wire _010_;
  wire _011_;
  wire _012_;
  wire _013_;
  wire _014_;
  wire _015_;
  wire _016_;
  wire _017_;
  wire _018_;
  wire _019_;
  wire _020_;
  wire _021_;
  wire _022_;
  wire _023_;
  wire _024_;
  wire _025_;
  wire _026_;
  wire _027_;
  wire _028_;
  wire _029_;
  wire _030_;
  wire _031_;
  wire _032_;
  wire _033_;
  wire _034_;
  wire _035_;
  wire _036_;
  wire _037_;
  wire _038_;
  wire _039_;
  wire _040_;
  wire _041_;
  wire _042_;
  wire _043_;
  wire _044_;
  wire _045_;
  wire _046_;
  wire _047_;
  wire _048_;
  wire _049_;
  wire _050_;
  wire _051_;
  wire _052_;
  wire _053_;
  wire _054_;
  wire _055_;
  wire _056_;
  wire _057_;
  wire _058_;
  wire _059_;
  wire _060_;
  wire _061_;
  wire _062_;
  wire _063_;
  wire _064_;
  wire _065_;
  wire _066_;
  wire _067_;
  wire _068_;
  wire _069_;
  wire _070_;
  wire _071_;
  wire _072_;
  wire _073_;
  wire _074_;
  wire _075_;
  wire _076_;
  wire _077_;
  wire _078_;
  wire _079_;
  wire _080_;
  wire _081_;
  wire _082_;
  wire _083_;
  (* src = "SAR_LOGIC.v:3.11-3.14" *)
  input clk;
  wire clk;
  (* src = "SAR_LOGIC.v:9.11-9.14" *)
  input cmp;
  wire cmp;
  (* src = "SAR_LOGIC.v:4.11-4.13" *)
  input go;
  wire go;
  (* src = "SAR_LOGIC.v:11.15-11.19" *)
  wire [7:0] mask;
  (* src = "SAR_LOGIC.v:6.18-6.24" *)
  output [7:0] result;
  wire [7:0] result;
  (* src = "SAR_LOGIC.v:7.12-7.18" *)
  output sample;
  wire sample;
  (* onehot = 32'd1 *)
  wire [3:0] state;
  (* src = "SAR_LOGIC.v:5.12-5.17" *)
  output valid;
  wire valid;
  (* src = "SAR_LOGIC.v:8.18-8.23" *)
  output [7:0] value;
  wire [7:0] value;
  NOT _084_ (
    .A(state[1]),
    .Y(_050_)
  );
  NOT _085_ (
```

```
  .A(mask[7]),
  .Y(_051_)
);
NOT _086_ (
  .A(mask[6]),
  .Y(_052_)
);
NOT _087_ (
  .A(mask[5]),
  .Y(_053_)
);
NOT _088_ (
  .A(mask[4]),
  .Y(_054_)
);
NOT _089_ (
  .A(mask[3]),
  .Y(_055_)
);
NOT _090_ (
  .A(mask[2]),
  .Y(_056_)
);
NOT _091_ (
  .A(mask[1]),
  .Y(_057_)
);
NOT _092_ (
  .A(go),
  .Y(_001_)
);
NOT _093_ (
  .A(cmp),
  .Y(_058_)
);
NOT _094_ (
  .A(mask[0]),
  .Y(_059_)
);
NOT _095_ (
  .A(result[0]),
  .Y(_060_)
);
NOT _096_ (
  .A(result[1]),
  .Y(_061_)
);
NOT _097_ (
  .A(result[2]),
  .Y(_062_)
);
NOT _098_ (
  .A(result[3]),
  .Y(_063_)
);
NOT _099_ (
  .A(result[4]),
  .Y(_064_)
);
NOT _100_ (
  .A(result[5]),
  .Y(_065_)
);
NOT _101_ (
  .A(result[6]),
  .Y(_066_)
);
NOT _102_ (
  .A(result[7]),
  .Y(_067_)
);
NAND _103_ (
  .A(_051_),
  .B(_067_),
  .Y(value[7])
);
NAND _104_ (
  .A(_052_),
  .B(_066_),
  .Y(value[6])
);
NAND _105_ (
  .A(_053_),
  .B(_065_),
  .Y(value[5])
);
NAND _106_ (
  .A(_054_),
  .B(_064_),
  .Y(value[4])
);
NAND _107_ (
  .A(_055_),
  .B(_063_),
  .Y(value[3])
);
NAND _108_ (
  .A(_056_),
  .B(_062_),
  .Y(value[2])
);
NAND _109_ (
  .A(_057_),
  .B(_061_),
  .Y(value[1])
);
NAND _110_ (
  .A(_059_),
  .B(_060_),
  .Y(value[0])
);
NOR _111_ (
  .A(state[1]),
  .B(sample),
  .Y(_068_)
);
NOT _112_ (
  .A(_068_),
  .Y(_069_)
);
NOR _113_ (
  .A(_001_),
  .B(_068_),
  .Y(_070_)
```

```
);
NAND _114_ (
  .A(go),
  .B(_069_),
  .Y(_071_)
);
NOR _115_ (
  .A(sample),
  .B(_059_),
  .Y(_072_)
);
NOR _116_ (
  .A(_071_),
  .B(_072_),
  .Y(_002_)
);
NAND _117_ (
  .A(state[1]),
  .B(mask[0]),
  .Y(_073_)
);
NOT _118_ (
  .A(_073_),
  .Y(_074_)
);
NOR _119_ (
  .A(valid),
  .B(_074_),
  .Y(_075_)
);
NOR _120_ (
  .A(_001_),
  .B(_075_),
  .Y(_003_)
);
NAND _121_ (
  .A(go),
  .B(state[0]),
  .Y(_076_)
);
NOT _122_ (
  .A(_076_),
  .Y(_000_)
);
NOR _123_ (
  .A(_050_),
  .B(_001_),
  .Y(_077_)
);
NAND _124_ (
  .A(state[1]),
  .B(go),
  .Y(_078_)
);
NOR _125_ (
  .A(mask[7]),
  .B(_070_),
  .Y(_079_)
);
NOR _126_ (
  .A(_077_),
  .B(_079_),
  .Y(_004_)
);
NAND _127_ (
  .A(mask[0]),
  .B(_071_),
  .Y(_080_)
);
NAND _128_ (
  .A(mask[1]),
  .B(_077_),
  .Y(_081_)
);
NAND _129_ (
  .A(_080_),
  .B(_081_),
  .Y(_005_)
);
NAND _130_ (
  .A(mask[1]),
  .B(_071_),
  .Y(_082_)
);
NAND _131_ (
  .A(mask[2]),
  .B(_077_),
  .Y(_083_)
);
NAND _132_ (
  .A(_082_),
  .B(_083_),
  .Y(_006_)
);
NAND _133_ (
  .A(mask[2]),
  .B(_071_),
  .Y(_020_)
);
NAND _134_ (
  .A(mask[3]),
  .B(_077_),
  .Y(_021_)
);
NAND _135_ (
  .A(_020_),
  .B(_021_),
  .Y(_007_)
);
NAND _136_ (
  .A(mask[3]),
  .B(_071_),
  .Y(_022_)
);
NAND _137_ (
  .A(mask[4]),
  .B(_077_),
  .Y(_023_)
);
NAND _138_ (
  .A(_022_),
  .B(_023_),
  .Y(_008_)
);
```

```
NAND _139_ (
  .A(mask[4]),
  .B(_071_),
  .Y(_024_)
);
NAND _140_ (
  .A(mask[5]),
  .B(_077_),
  .Y(_025_)
);
NAND _141_ (
  .A(_024_),
  .B(_025_),
  .Y(_009_)
);
NAND _142_ (
  .A(mask[5]),
  .B(_071_),
  .Y(_026_)
);
NAND _143_ (
  .A(mask[6]),
  .B(_077_),
  .Y(_027_)
);
NAND _144_ (
  .A(_026_),
  .B(_027_),
  .Y(_010_)
);
NAND _145_ (
  .A(mask[6]),
  .B(_071_),
  .Y(_028_)
);
NAND _146_ (
  .A(mask[7]),
  .B(_077_),
  .Y(_029_)
);
NAND _147_ (
  .A(_028_),
  .B(_029_),
  .Y(_011_)
);
NOR _148_ (
  .A(_050_),
  .B(cmp),
  .Y(_030_)
);
NAND _149_ (
  .A(state[1]),
  .B(_058_),
  .Y(_031_)
);
NAND _150_ (
  .A(_070_),
  .B(_031_),
  .Y(_032_)
);
NAND _151_ (
  .A(result[0]),
  .B(_032_),
  .Y(_033_)
);
NOR _152_ (
  .A(_078_),
  .B(_030_),
  .Y(_034_)
);
NAND _153_ (
  .A(value[0]),
  .B(_034_),
  .Y(_035_)
);
NAND _154_ (
  .A(_033_),
  .B(_035_),
  .Y(_012_)
);
NAND _155_ (
  .A(result[1]),
  .B(_032_),
  .Y(_036_)
);
NAND _156_ (
  .A(value[1]),
  .B(_034_),
  .Y(_037_)
);
NAND _157_ (
  .A(_036_),
  .B(_037_),
  .Y(_013_)
);
NAND _158_ (
  .A(result[2]),
  .B(_032_),
  .Y(_038_)
);
NAND _159_ (
  .A(value[2]),
  .B(_034_),
  .Y(_039_)
);
NAND _160_ (
  .A(_038_),
  .B(_039_),
  .Y(_014_)
);
NAND _161_ (
  .A(result[3]),
  .B(_032_),
  .Y(_040_)
);
NAND _162_ (
  .A(value[3]),
  .B(_034_),
  .Y(_041_)
);
NAND _163_ (
  .A(_040_),
  .B(_041_),
  .Y(_015_)
```

```
);
NAND _164_ (
  .A(result[4]),
  .B(_032_),
  .Y(_042_)
);
NAND _165_ (
  .A(value[4]),
  .B(_034_),
  .Y(_043_)
);
NAND _166_ (
  .A(_042_),
  .B(_043_),
  .Y(_016_)
);
NAND _167_ (
  .A(result[5]),
  .B(_032_),
  .Y(_044_)
);
NAND _168_ (
  .A(value[5]),
  .B(_034_),
  .Y(_045_)
);
NAND _169_ (
  .A(_044_),
  .B(_045_),
  .Y(_017_)
);
NAND _170_ (
  .A(result[6]),
  .B(_032_),
  .Y(_046_)
);
NAND _171_ (
  .A(value[6]),
  .B(_034_),
  .Y(_047_)
);
NAND _172_ (
  .A(_046_),
  .B(_047_),
  .Y(_018_)
);
NAND _173_ (
  .A(result[7]),
  .B(_032_),
  .Y(_048_)
);
NAND _174_ (
  .A(value[7]),
  .B(_034_),
  .Y(_049_)
);
NAND _175_ (
  .A(_048_),
  .B(_049_),
  .Y(_019_)
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _176_ (
  .C(clk),
  .D(_004_),
  .Q(mask[7])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _177_ (
  .C(clk),
  .D(_005_),
  .Q(mask[0])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _178_ (
  .C(clk),
  .D(_006_),
  .Q(mask[1])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _179_ (
  .C(clk),
  .D(_007_),
  .Q(mask[2])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _180_ (
  .C(clk),
  .D(_008_),
  .Q(mask[3])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _181_ (
  .C(clk),
  .D(_009_),
  .Q(mask[4])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _182_ (
  .C(clk),
  .D(_010_),
  .Q(mask[5])
);
(* src = "SAR_LOGIC.v:18.5-39.8" *)
DFF _183_ (
  .C(clk),
  .D(_011_),
  .Q(mask[6])
);
DFF _184_ (
  .C(clk),
  .D(_001_),
  .Q(state[0])
);
DFF _185_ (
  .C(clk),
  .D(_002_),
  .Q(state[1])
);
DFF _186_ (
  .C(clk),
  .D(_000_),
  .Q(sample)
);
```

```
  DFF _187_ (
    .C(clk),
    .D(_003_),
    .Q(valid)
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _188_ (
    .C(clk),
    .D(_012_),
    .Q(result[0])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _189_ (
    .C(clk),
    .D(_013_),
    .Q(result[1])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _190_ (
    .C(clk),
    .D(_014_),
    .Q(result[2])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _191_ (
    .C(clk),
    .D(_015_),
    .Q(result[3])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _192_ (
    .C(clk),
    .D(_016_),
    .Q(result[4])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _193_ (
    .C(clk),
    .D(_017_),
    .Q(result[5])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _194_ (
    .C(clk),
    .D(_018_),
    .Q(result[6])
  );
  (* src = "SAR_LOGIC.v:18.5-39.8" *)
  DFF _195_ (
    .C(clk),
    .D(_019_),
    .Q(result[7])
);
  assign state[3:2] = { valid, sample };
endmodule
```

DOT FILE FOR VISUALIZATION

```
digraph "controller" {
rankdir="LR";
remincross=true;
n1 [ shape=diamond, label="_000_", color="black",
fontcolor="black"];
n2 [ shape=diamond, label="_001_", color="black",
fontcolor="black"];
n3 [ shape=diamond, label="_002_", color="black",
fontcolor="black"];
n4 [ shape=diamond, label="_003_", color="black",
fontcolor="black"];
n5 [ shape=diamond, label="_004_", color="black",
fontcolor="black"];
n6 [ shape=diamond, label="_005_", color="black",
fontcolor="black"];
n7 [ shape=diamond, label="_006_", color="black",
fontcolor="black"];
n8 [ shape=diamond, label="_007_", color="black",
fontcolor="black"];
n9 [ shape=diamond, label="_008_", color="black",
fontcolor="black"];
n10 [ shape=diamond, label="_009_", color="black",
fontcolor="black"];
n11 [ shape=diamond, label="_010_", color="black",
fontcolor="black"];
n12 [ shape=diamond, label="_011_", color="black",
fontcolor="black"];
n13 [ shape=diamond, label="_012_", color="black",
fontcolor="black"];
n14 [ shape=diamond, label="_013_", color="black",
fontcolor="black"];
n15 [ shape=diamond, label="_014_", color="black",
fontcolor="black"];
n16 [ shape=diamond, label="_015_", color="black",
fontcolor="black"];
n17 [ shape=diamond, label="_016_", color="black",
fontcolor="black"];
n18 [ shape=diamond, label="_017_", color="black",
fontcolor="black"];
n19 [ shape=diamond, label="_018_", color="black",
fontcolor="black"];
n20 [ shape=diamond, label="_019_", color="black",
fontcolor="black"];
n21 [ shape=diamond, label="_020_", color="black",
fontcolor="black"];
n22 [ shape=diamond, label="_021_", color="black",
fontcolor="black"];
n23 [ shape=diamond, label="_022_", color="black",
fontcolor="black"];
n24 [ shape=diamond, label="_023_", color="black",
fontcolor="black"];
n25 [ shape=diamond, label="_024_", color="black",
fontcolor="black"];
n26 [ shape=diamond, label="_025_", color="black",
fontcolor="black"];
n27 [ shape=diamond, label="_026_", color="black",
fontcolor="black"];
n28 [ shape=diamond, label="_027_", color="black",
fontcolor="black"];
```

n29 [shape=diamond, label="_028_", color="black", fontcolor="black"];
n30 [shape=diamond, label="_029_", color="black", fontcolor="black"];
n31 [shape=diamond, label="_030_", color="black", fontcolor="black"];
n32 [shape=diamond, label="_031_", color="black", fontcolor="black"];
n33 [shape=diamond, label="_032_", color="black", fontcolor="black"];
n34 [shape=diamond, label="_033_", color="black", fontcolor="black"];
n35 [shape=diamond, label="_034_", color="black", fontcolor="black"];
n36 [shape=diamond, label="_035_", color="black", fontcolor="black"];
n37 [shape=diamond, label="_036_", color="black", fontcolor="black"];
n38 [shape=diamond, label="_037_", color="black", fontcolor="black"];
n39 [shape=diamond, label="_038_", color="black", fontcolor="black"];
n40 [shape=diamond, label="_039_", color="black", fontcolor="black"];
n41 [shape=diamond, label="_040_", color="black", fontcolor="black"];
n42 [shape=diamond, label="_041_", color="black", fontcolor="black"];
n43 [shape=diamond, label="_042_", color="black", fontcolor="black"];
n44 [shape=diamond, label="_043_", color="black", fontcolor="black"];
n45 [shape=diamond, label="_044_", color="black", fontcolor="black"];
n46 [shape=diamond, label="_045_", color="black", fontcolor="black"];
n47 [shape=diamond, label="_046_", color="black", fontcolor="black"];
n48 [shape=diamond, label="_047_", color="black", fontcolor="black"];
n49 [shape=diamond, label="_048_", color="black", fontcolor="black"];
n50 [shape=diamond, label="_049_", color="black", fontcolor="black"];
n51 [shape=diamond, label="_050_", color="black", fontcolor="black"];
n52 [shape=diamond, label="_051_", color="black", fontcolor="black"];
n53 [shape=diamond, label="_052_", color="black", fontcolor="black"];
n54 [shape=diamond, label="_053_", color="black", fontcolor="black"];
n55 [shape=diamond, label="_054_", color="black", fontcolor="black"];
n56 [shape=diamond, label="_055_", color="black", fontcolor="black"];
n57 [shape=diamond, label="_056_", color="black", fontcolor="black"];
n58 [shape=diamond, label="_057_", color="black", fontcolor="black"];
n59 [shape=diamond, label="_058_", color="black", fontcolor="black"];
n60 [shape=diamond, label="_059_", color="black", fontcolor="black"];
n61 [shape=diamond, label="_060_", color="black", fontcolor="black"];
n62 [shape=diamond, label="_061_", color="black", fontcolor="black"];
n63 [shape=diamond, label="_062_", color="black", fontcolor="black"];
n64 [shape=diamond, label="_063_", color="black", fontcolor="black"];
n65 [shape=diamond, label="_064_", color="black", fontcolor="black"];
n66 [shape=diamond, label="_065_", color="black", fontcolor="black"];
n67 [shape=diamond, label="_066_", color="black", fontcolor="black"];
n68 [shape=diamond, label="_067_", color="black", fontcolor="black"];
n69 [shape=diamond, label="_068_", color="black", fontcolor="black"];
n70 [shape=diamond, label="_069_", color="black", fontcolor="black"];
n71 [shape=diamond, label="_070_", color="black", fontcolor="black"];
n72 [shape=diamond, label="_071_", color="black", fontcolor="black"];
n73 [shape=diamond, label="_072_", color="black", fontcolor="black"];
n74 [shape=diamond, label="_073_", color="black", fontcolor="black"];
n75 [shape=diamond, label="_074_", color="black", fontcolor="black"];
n76 [shape=diamond, label="_075_", color="black", fontcolor="black"];
n77 [shape=diamond, label="_076_", color="black", fontcolor="black"];
n78 [shape=diamond, label="_077_", color="black", fontcolor="black"];
n79 [shape=diamond, label="_078_", color="black", fontcolor="black"];
n80 [shape=diamond, label="_079_", color="black", fontcolor="black"];
n81 [shape=diamond, label="_080_", color="black", fontcolor="black"];
n82 [shape=diamond, label="_081_", color="black", fontcolor="black"];
n83 [shape=diamond, label="_082_", color="black", fontcolor="black"];
n84 [shape=diamond, label="_083_", color="black", fontcolor="black"];
n85 [shape=octagon, label="clk", color="black", fontcolor="black"];
n86 [shape=octagon, label="cmp", color="black", fontcolor="black"];
n87 [shape=octagon, label="go", color="black", fontcolor="black"];
n88 [shape=diamond, label="mask", color="black", fontcolor="black"];
n89 [shape=octagon, label="result", color="black", fontcolor="black"];
n90 [shape=octagon, label="sample", color="black", fontcolor="black"];

```
n91 [ shape=diamond, label="state", color="black", fontcolor="black"];
n92 [ shape=octagon, label="valid", color="black", fontcolor="black"];
n93 [ shape=octagon, label="value", color="black", fontcolor="black"];
c96 [ shape=record, label="{{<p94> A|<p95> Y}|_084_\nNOT|{}}", ];
x0 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x0:e -> c96:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c97 [ shape=record, label="{{<p94> A|<p95> Y}|_085_\nNOT|{}}", ];
x1 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x1:e -> c97:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c98 [ shape=record, label="{{<p94> A|<p95> Y}|_086_\nNOT|{}}", ];
x2 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x2:e -> c98:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c99 [ shape=record, label="{{<p94> A|<p95> Y}|_087_\nNOT|{}}", ];
x3 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x3:e -> c99:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c100 [ shape=record, label="{{<p94> A|<p95> Y}|_088_\nNOT|{}}", ];
x4 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x4:e -> c100:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c101 [ shape=record, label="{{<p94> A|<p95> Y}|_089_\nNOT|{}}", ];
x5 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x5:e -> c101:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c102 [ shape=record, label="{{<p94> A|<p95> Y}|_090_\nNOT|{}}", ];
x6 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x6:e -> c102:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c103 [ shape=record, label="{{<p94> A|<p95> Y}|_091_\nNOT|{}}", ];
x7 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x7:e -> c103:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c104 [ shape=record, label="{{<p94> A|<p95> Y}|_092_\nNOT|{}}", ];
c105 [ shape=record, label="{{<p94> A|<p95> Y}|_093_\nNOT|{}}", ];
c106 [ shape=record, label="{{<p94> A|<p95> Y}|_094_\nNOT|{}}", ];
x8 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x8:e -> c106:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c107 [ shape=record, label="{{<p94> A|<p95> Y}|_095_\nNOT|{}}", ];
x9 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x9:e -> c107:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c108 [ shape=record, label="{{<p94> A|<p95> Y}|_096_\nNOT|{}}", ];
x10 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x10:e -> c108:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c109 [ shape=record, label="{{<p94> A|<p95> Y}|_097_\nNOT|{}}", ];
x11 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x11:e -> c109:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c110 [ shape=record, label="{{<p94> A|<p95> Y}|_098_\nNOT|{}}", ];
x12 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x12:e -> c110:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c111 [ shape=record, label="{{<p94> A|<p95> Y}|_099_\nNOT|{}}", ];
x13 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x13:e -> c111:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c112 [ shape=record, label="{{<p94> A|<p95> Y}|_100_\nNOT|{}}", ];
x14 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x14:e -> c112:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c113 [ shape=record, label="{{<p94> A|<p95> Y}|_101_\nNOT|{}}", ];
x15 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x15:e -> c113:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c114 [ shape=record, label="{{<p94> A|<p95> Y}|_102_\nNOT|{}}", ];
```

```
x16 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x16:e -> c114:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c116 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_103_\nNAND|{}}", ];
x17 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x17:e -> c116:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c117 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_104_\nNAND|{}}", ];
x18 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x18:e -> c117:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c118 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_105_\nNAND|{}}", ];
x19 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x19:e -> c118:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c119 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_106_\nNAND|{}}", ];
x20 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x20:e -> c119:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c120 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_107_\nNAND|{}}", ];
x21 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x21:e -> c120:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c121 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_108_\nNAND|{}}", ];
x22 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x22:e -> c121:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c122 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_109_\nNAND|{}}", ];
x23 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x23:e -> c122:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c123 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_110_\nNAND|{}}", ];
x24 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x24:e -> c123:p95:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c124 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_111_\nNOR|{}}", ];
x25 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x25:e -> c124:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c125 [ shape=record, label="{{<p94> A|<p95> Y}|_112_\nNOT|{}}", ];
c126 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_113_\nNOR|{}}", ];
c127 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_114_\nNAND|{}}", ];
c128 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_115_\nNOR|{}}", ];
c129 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_116_\nNOR|{}}", ];
c130 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_117_\nNAND|{}}", ];
x26 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x26:e -> c130:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
x27 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x27:e -> c130:p115:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c131 [ shape=record, label="{{<p94> A|<p95> Y}|_118_\nNOT|{}}", ];
c132 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_119_\nNOR|{}}", ];
c133 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_120_\nNOR|{}}", ];
c134 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_121_\nNAND|{}}", ];
x28 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x28:e -> c134:p115:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c135 [ shape=record, label="{{<p94> A|<p95> Y}|_122_\nNOT|{}}", ];
c136 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_123_\nNOR|{}}", ];
c137 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_124_\nNAND|{}}", ];
x29 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x29:e -> c137:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c138 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_125_\nNOR|{}}", ];
x30 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x30:e -> c138:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c139 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_126_\nNOR|{}}", ];
```

```
c140 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_127_\nNAND|{}}",  ];
x31 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x31:e -> c140:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c141 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_128_\nNAND|{}}",  ];
x32 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x32:e -> c141:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c142 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_129_\nNAND|{}}",  ];
c143 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_130_\nNAND|{}}",  ];
x33 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x33:e -> c143:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c144 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_131_\nNAND|{}}",  ];
x34 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x34:e -> c144:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c145 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_132_\nNAND|{}}",  ];
c146 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_133_\nNAND|{}}",  ];
x35 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x35:e -> c146:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c147 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_134_\nNAND|{}}",  ];
x36 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x36:e -> c147:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c148 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_135_\nNAND|{}}",  ];
c149 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_136_\nNAND|{}}",  ];
x37 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x37:e -> c149:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c150 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_137_\nNAND|{}}",  ];
x38 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x38:e -> c150:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c151 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_138_\nNAND|{}}",  ];
c152 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_139_\nNAND|{}}",  ];
x39 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x39:e -> c152:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c153 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_140_\nNAND|{}}",  ];
x40 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x40:e -> c153:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c154 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_141_\nNAND|{}}",  ];
c155 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_142_\nNAND|{}}",  ];
x41 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x41:e -> c155:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c156 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_143_\nNAND|{}}",  ];
x42 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x42:e -> c156:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c157 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_144_\nNAND|{}}",  ];
c158 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_145_\nNAND|{}}",  ];
x43 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x43:e -> c158:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c159 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_146_\nNAND|{}}",  ];
x44 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x44:e -> c159:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c160 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_147_\nNAND|{}}",  ];
c161 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_148_\nNOR|{}}",  ];
c162 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_149_\nNAND|{}}",  ];
x45 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x45:e -> c162:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c163 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_150_\nNAND|{}}",  ];
```

```
c164 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_151_\nNAND|{}}",  ];
x46 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x46:e -> c164:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c165 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_152_\nNOR|{}}",  ];
c166 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_153_\nNAND|{}}",  ];
x47 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x47:e -> c166:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c167 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_154_\nNAND|{}}",  ];
c168 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_155_\nNAND|{}}",  ];
x48 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x48:e -> c168:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c169 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_156_\nNAND|{}}",  ];
x49 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x49:e -> c169:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c170 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_157_\nNAND|{}}",  ];
c171 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_158_\nNAND|{}}",  ];
x50 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x50:e -> c171:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c172 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_159_\nNAND|{}}",  ];
x51 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x51:e -> c172:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c173 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_160_\nNAND|{}}",  ];
c174 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_161_\nNAND|{}}",  ];
x52 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x52:e -> c174:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c175 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_162_\nNAND|{}}",  ];
x53 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x53:e -> c175:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c176 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_163_\nNAND|{}}",  ];
c177 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_164_\nNAND|{}}",  ];
x54 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x54:e -> c177:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c178 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_165_\nNAND|{}}",  ];
x55 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x55:e -> c178:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c179 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_166_\nNAND|{}}",  ];
c180 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_167_\nNAND|{}}",  ];
x56 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x56:e -> c180:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c181 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_168_\nNAND|{}}",  ];
x57 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x57:e -> c181:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c182 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_169_\nNAND|{}}",  ];
c183 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_170_\nNAND|{}}",  ];
x58 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x58:e -> c183:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c184 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_171_\nNAND|{}}",  ];
x59 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x59:e -> c184:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c185 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_172_\nNAND|{}}",  ];
c186 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_173_\nNAND|{}}",  ];
x60 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x60:e -> c186:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="6", fontcolor="6", label=""];
c187 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_174_\nNAND|{}}",  ];
```

```
x61 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="6", fontcolor="6" ];
x61:e -> c187:p94:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c188 [ shape=record, label="{{<p94> A|<p115> B|<p95> Y}|_175_\nNAND|{}}",  ];
c192 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_176_\nDFF|{}}",  ];
x62 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x62:e -> c192:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c193 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_177_\nDFF|{}}",  ];
x63 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x63:e -> c193:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c194 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_178_\nDFF|{}}",  ];
x64 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x64:e -> c194:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c195 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_179_\nDFF|{}}",  ];
x65 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x65:e -> c195:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
c196 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_180_\nDFF|{}}",  ];
x66 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x66:e -> c196:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c197 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_181_\nDFF|{}}",  ];
x67 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x67:e -> c197:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c198 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_182_\nDFF|{}}",  ];
x68 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x68:e -> c198:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c199 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_183_\nDFF|{}}",  ];
x69 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x69:e -> c199:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c200 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_184_\nDFF|{}}",  ];
x70 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x70:e -> c200:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c201 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_185_\nDFF|{}}",  ];
x71 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
x71:e -> c201:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c202 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_186_\nDFF|{}}",  ];
c203 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_187_\nDFF|{}}",  ];
c204 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_188_\nDFF|{}}",  ];
x72 [ shape=record, style=rounded, label="<s0> 0:0 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x72:e -> c204:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="7", fontcolor="7", label=""];
c205 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_189_\nDFF|{}}",  ];
x73 [ shape=record, style=rounded, label="<s0> 1:1 - 0:0 ", colorscheme="dark28", color="7", fontcolor="7" ];
x73:e -> c205:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="1", fontcolor="1", label=""];
c206 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_190_\nDFF|{}}",  ];
x74 [ shape=record, style=rounded, label="<s0> 2:2 - 0:0 ", colorscheme="dark28", color="1", fontcolor="1" ];
x74:e -> c206:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c207 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_191_\nDFF|{}}",  ];
x75 [ shape=record, style=rounded, label="<s0> 3:3 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x75:e -> c207:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="3", fontcolor="3", label=""];
c208 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_192_\nDFF|{}}",  ];
x76 [ shape=record, style=rounded, label="<s0> 4:4 - 0:0 ", colorscheme="dark28", color="3", fontcolor="3" ];
x76:e -> c208:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="4", fontcolor="4", label=""];
c209 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_193_\nDFF|{}}",  ];
x77 [ shape=record, style=rounded, label="<s0> 5:5 - 0:0 ", colorscheme="dark28", color="4", fontcolor="4" ];
```

```
x77:e -> c209:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="5", fontcolor="5", label=""];
c210 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_194_\nDFF|{}}",  ];
x78 [ shape=record, style=rounded, label="<s0> 6:6 - 0:0 ", colorscheme="dark28", color="5", fontcolor="5" ];
x78:e -> c210:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", label=""];
c211 [ shape=record, label="{{<p189> C|<p190> D|<p191> Q}|_195_\nDFF|{}}",  ];
x79 [ shape=record, style=rounded, label="<s0> 7:7 - 0:0 ", colorscheme="dark28", color="8", fontcolor="8" ];
x79:e -> c211:p191:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="2", fontcolor="2", label=""];
x80 [ shape=record, style=rounded, label="<s1> 0:0 - 1:1 |<s0> 0:0 - 0:0 ", colorscheme="dark28", color="2", fontcolor="2" ];
x81 [ shape=record, style=rounded, label="<s0> 1:0 - 3:2 ", colorscheme="dark28", color="2", fontcolor="2" ];
x80:e -> x81:w [arrowhead=odiamond, arrowtail=odiamond, dir=both, colorscheme="dark28", color="8", fontcolor="8", style="setlinewidth(3)", label="<2>"];
n1:e -> c135:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n1:e -> c202:p190:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n10:e -> c154:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n10:e -> c197:p190:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n11:e -> c157:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n11:e -> c198:p190:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n12:e -> c160:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n12:e -> c199:p190:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n13:e -> c167:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n13:e -> c204:p190:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n14:e -> c170:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n14:e -> c205:p190:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n15:e -> c173:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n15:e -> c206:p190:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n16:e -> c176:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n16:e -> c207:p190:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n17:e -> c179:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n17:e -> c208:p190:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n18:e -> c182:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n18:e -> c209:p190:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n19:e -> c185:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n19:e -> c210:p190:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n2:e -> c104:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n2:e -> c126:p94:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n2:e -> c133:p94:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n2:e -> c136:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n2:e -> c200:p190:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n20:e -> c188:p95:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n20:e -> c211:p190:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n21:e -> c146:p95:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n21:e -> c148:p94:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n22:e -> c147:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n22:e -> c148:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n23:e -> c149:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n23:e -> c151:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n24:e -> c150:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n24:e -> c151:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n25:e -> c152:p95:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n25:e -> c154:p94:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n26:e -> c153:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n26:e -> c154:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n27:e -> c155:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n27:e -> c157:p94:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n28:e -> c156:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n28:e -> c157:p115:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n29:e -> c158:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n29:e -> c160:p94:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n3:e -> c129:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n3:e -> c201:p190:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
```

```
n30:e -> c159:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n30:e -> c160:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n31:e -> c161:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n31:e -> c165:p115:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n32:e -> c162:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n32:e -> c163:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n33:e -> c163:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c164:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c168:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c171:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c174:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c177:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c180:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c183:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n33:e -> c186:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n34:e -> c164:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n34:e -> c167:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n35:e -> c165:p95:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c166:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c169:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c172:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c175:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c178:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c181:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c184:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n35:e -> c187:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n36:e -> c166:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n36:e -> c167:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n37:e -> c168:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n37:e -> c170:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n38:e -> c169:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n38:e -> c170:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n39:e -> c171:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n39:e -> c173:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n4:e -> c133:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n4:e -> c203:p190:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n40:e -> c172:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n40:e -> c173:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n41:e -> c174:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n41:e -> c176:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n42:e -> c175:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n42:e -> c176:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n43:e -> c177:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n43:e -> c179:p94:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n44:e -> c178:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n44:e -> c179:p115:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n45:e -> c180:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n45:e -> c182:p94:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n46:e -> c181:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n46:e -> c182:p115:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n47:e -> c183:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n47:e -> c185:p94:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n48:e -> c184:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n48:e -> c185:p115:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n49:e -> c186:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n49:e -> c188:p94:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n5:e -> c139:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n5:e -> c192:p190:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n50:e -> c187:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n50:e -> c188:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n51:e -> c136:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n51:e -> c161:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
```

```
n51:e -> c96:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n52:e -> c116:p94:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n52:e -> c97:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n53:e -> c117:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n53:e -> c98:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n54:e -> c118:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n54:e -> c99:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n55:e -> c100:p95:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n55:e -> c119:p94:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n56:e -> c101:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n56:e -> c120:p94:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n57:e -> c102:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n57:e -> c121:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n58:e -> c103:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n58:e -> c122:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n59:e -> c105:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n59:e -> c162:p115:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n6:e -> c142:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n6:e -> c193:p190:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n60:e -> c106:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n60:e -> c123:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n60:e -> c128:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n61:e -> c107:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n61:e -> c123:p115:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n62:e -> c108:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n62:e -> c122:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n63:e -> c109:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n63:e -> c121:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n64:e -> c110:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n64:e -> c120:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n65:e -> c111:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n65:e -> c119:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n66:e -> c112:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n66:e -> c118:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n67:e -> c113:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n67:e -> c117:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n68:e -> c114:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n68:e -> c116:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n69:e -> c124:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n69:e -> c125:p94:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n69:e -> c126:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n7:e -> c145:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n7:e -> c194:p190:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n70:e -> c125:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n70:e -> c127:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n71:e -> c126:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n71:e -> c138:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n71:e -> c163:p94:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n72:e -> c127:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c129:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c140:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c143:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c146:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c149:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c152:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c155:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n72:e -> c158:p115:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n73:e -> c128:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n73:e -> c129:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n74:e -> c130:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n74:e -> c131:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n75:e -> c131:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
```

```
n75:e -> c132:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n76:e -> c132:p95:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n76:e -> c133:p115:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n77:e -> c134:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n77:e -> c135:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c136:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c139:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c141:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c144:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c147:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c150:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c153:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c156:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n78:e -> c159:p115:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n79:e -> c137:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n79:e -> c165:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n8:e -> c148:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n8:e -> c195:p190:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n80:e -> c138:p95:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n80:e -> c139:p115:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n81:e -> c140:p95:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n81:e -> c142:p94:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n82:e -> c141:p95:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n82:e -> c142:p115:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n83:e -> c143:p95:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n83:e -> c145:p94:w [colorscheme="dark28", color="7", fontcolor="7", label=""];
n84:e -> c144:p95:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n84:e -> c145:p115:w [colorscheme="dark28", color="2", fontcolor="2", label=""];
n85:e -> c192:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c193:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c194:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c195:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c196:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c197:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c198:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c199:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c200:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c201:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c202:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c203:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c204:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c205:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c206:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c207:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c208:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c209:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c210:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n85:e -> c211:p189:w [colorscheme="dark28", color="6", fontcolor="6", label=""];
n86:e -> c105:p94:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n86:e -> c161:p115:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n87:e -> c104:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n87:e -> c127:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n87:e -> c134:p94:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n87:e -> c137:p115:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n88:e -> x1:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x27:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x2:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x30:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x31:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x32:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x33:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x34:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
```

```
n88:e -> x35:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x36:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x37:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x38:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x39:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x3:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x40:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x41:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x42:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x43:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x44:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x4:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x5:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x62:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x63:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x64:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x65:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x66:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x67:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x68:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x69:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x6:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x7:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n88:e -> x8:s0:w [colorscheme="dark28", color="4", fontcolor="4", label=""];
n89:e -> x10:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x11:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x12:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x13:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x14:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x15:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x16:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x46:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x48:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x50:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x52:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x54:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x56:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x58:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x60:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x72:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x73:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x74:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x75:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x76:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x77:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x78:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x79:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n89:e -> x9:s0:w [colorscheme="dark28", color="1", fontcolor="1", label=""];
n9:e -> c151:p95:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n9:e -> c196:p190:w [colorscheme="dark28", color="5", fontcolor="5", label=""];
n90:e -> c124:p115:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n90:e -> c128:p94:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n90:e -> c202:p191:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
n90:e -> x80:s0:w [colorscheme="dark28", color="8", fontcolor="8", label=""];
x81:s0:e -> n91:w [colorscheme="dark28", color="3", fontcolor="3", style="setlinewidth(3)", label="<2>"];
n91:e -> x0:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x25:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x26:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x28:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x29:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x45:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
n91:e -> x70:s0:w [colorscheme="dark28", color="3", fontcolor="3", label=""];
```

```
n91:e -> x71:s0:w [colorscheme="dark28", color="3",
fontcolor="3", label=""];
n92:e -> c132:p94:w [colorscheme="dark28", color="5",
fontcolor="5", label=""];
n92:e -> c203:p191:w [colorscheme="dark28", color="5",
fontcolor="5", label=""];
n92:e -> x80:s1:w [colorscheme="dark28", color="5",
fontcolor="5", label=""];
n93:e -> x17:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x18:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x19:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x20:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x21:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x22:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x23:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x24:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x47:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x49:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x51:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x53:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x55:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x57:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x59:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
n93:e -> x61:s0:w [colorscheme="dark28", color="8",
fontcolor="8", label=""];
}
```

References

[1] R K. Singh and A. Mishra, "Design of Low Power Dynamic Comparator for SAR ADC," *IEEE Conference Publications*, 2021. Available: https://ieeexplore.ieee.org/document/9780696【6】

[2] A Bahari and M. Soleimani, "Design of a Low Power, High-Speed Double-Tail Comparator," *IEEE Conference Publications*, 2017. Available: https://ieeexplore.ieee.org/document/8074370【7】..

[3] B. Goll and H. Zimmermann, "Analysis and Design of a Low-Voltage Low-Power Double-Tail Comparator," *IEEE Transactions on Circuits*

and Systems I: Regular Papers, vol. 60, no. 6, pp. 1648–1656, June 2013. Available: https://ieeexplore.ieee.org/document/6459609【8】

[4] M. Hasan and S. S. Roy, "Design of Double-Tail Dynamic Latch Comparator for Low Power Applications," *IEEE Conference Publications*, 2019. Available: https://ieeexplore.ieee.org/document/8908119【9】

[5] S. Ravi and P. Patel, "Low-Power and High-Speed SAR ADC Using Modified Comparator Design," in *Proceedings of the IEEE Conference on VLSI Design*, 2022.

[6] A. Kumar and R. Verma, "Comparative Study of Double-Tail Dynamic Comparators for SAR ADCs," *IEEE Access*, vol. 8, pp. 34578–34588, 2020

[7] N. Pandey and J. K. Gupta, "Energy-Efficient SAR ADC Design with Enhanced Dynamic Comparator," in *Proceedings of the IEEE Conference on Electronics Design*, 2021.

[8] P. Zhao and K. Wei, "A Subthreshold SAR ADC with Novel Comparator for IoT Devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 937–941, May 2020.

[9] Y. Tang and L. Gao, "Double-Tail Comparator Optimization for Low-Voltage SAR ADCs," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 923–930, April 2018

[10] R. P. Sharma and V. Yadav, "Performance Analysis of Low-Power Comparators in SAR ADCs," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1170–1179, June 2021

.