1 Introduction

1.1 Referenced documents

List all documents (specifications, standards, journal articles, text books) useful to help understand the current document.

• Teja PS. Design of radix-8 booth multiplier using koggestone adder for high speed Arithmetic applications.
Emerging Trends in Electrical, Electronics Instrumentation Engineering: An international Journal (EEIEJ). 2014 Feb;1(1):45-55.

• K. Sindhuja, C. Thiruvenkatesan, 2017, Design of Low Power Approximate Radix-8 Booth Multiplier,
INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT) RTICCT
– 2017 (Volume 5 – Issue 17)

1.2 Design library name

Give library name; adhere

UMC65LLSC

1.3 People involved in the block

Please list all people involved in the design

Akshat Mathur (2023EEN2223)

2 Function

2.1 Brief overview

Give functional and electrical specifications of the design. Specify all inputs, processes (controls and algorithms) and outputs. Describe all relevant concepts and assumptions/limitations. All descriptions in bulleted points

Multiplication Accumulator multiplies the input data optimally and stores that value in accumulator, with subsequent data inputs the preexisting data in the accumulator is added with the new multiplication output.

Multiplication algorithm used is RADIX -8 Booth Algorithm. And pipelining has been introduced at Partial product and multiplication stages

1. OP code generator

   From the multiplier (A) certain bits are extracted with form op codes.

PP bits of b	Partial Products
0 0 0 0	0a
0 0 0 1	+1a
0 0 1 0	+1a
0 0 1 1	+2a
0 1 0 0	+2a
0 1 0 1	+3a
0 1 1 0	+3a
0 1 1 1	+4a
1 0 0 0	-4a
1 0 0 1	-3a
1 0 1 0	-3a
1 0 1 1	-2a
1 1 0 0	-2a
1 1 0 1	-1a
1 1 1 0	-1a
1 1 1 1	0a

Table 1 Radix-8 encoding

2. PP generator

   This partial product generator, based on values from op codes generates Partial products using the multiplicand (B).

3. Driving logic
   Based on spec, the data is truncated and rounded off to 27 bits before driving to output.

2.2 Interfaces

Include a table showing all the ports associated with this block; the table should have the following information:

- Group signals with a common interface together.
- Indicate bus ranges as part of the signal name, for example DATATOPERIPH <15:0>.

Signal name
name of the signal, as it appears on the opus symbol
eg. SIGNAL <a:b>

I/O
input/output

Description
brief description

Logical grouping
eg: peripheral access, link interface control

Signal Name	I/O	Description
clk	Input	Global clock
rst	Input	Global reset
[15:0]A	Input	multiplier Data signal
[15:0]B	Input	multiplicand Input Data signal
[26:0] out	Output	Accumulator output

Table1 : Input Output signal and description

2.3 Architecture

Include a diagram showing the conceptual split of the functions of the block (this is not necessarily the same as the actual logical or physical implementation) to illustrate.

The Basic architecture of MAC is as follows.

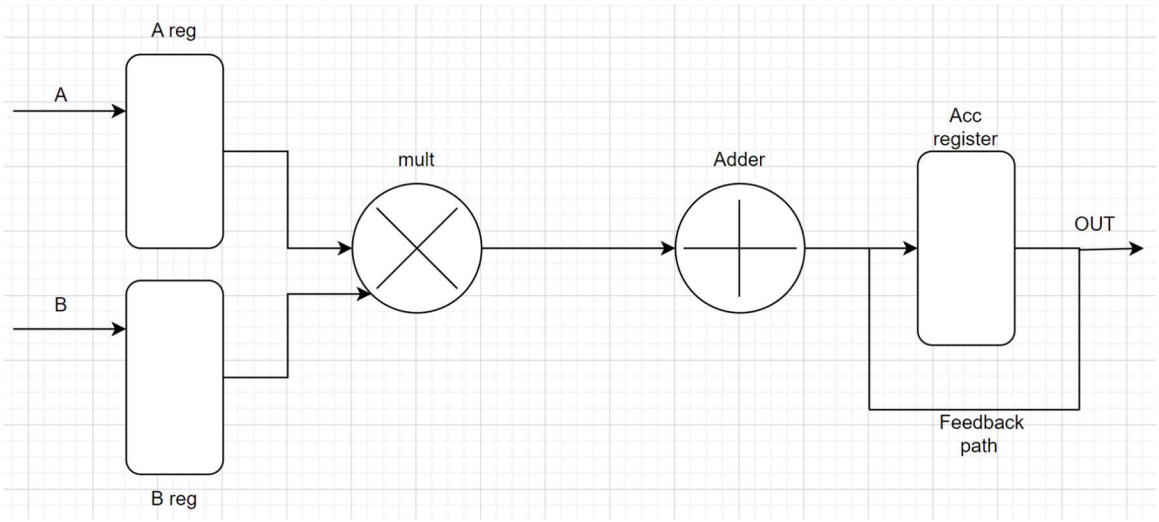

Fig1 : Basic MAC architecture

2.4 Detailed functional description

Include a detailed description of what the block does, using the functional split in the Architecture diagram to aid the description. The information should be sufficient to allow a chip integrator to connect it into a higher level of design hierarchy, and a verifier to establish what the testing/verification requirements of the block are.

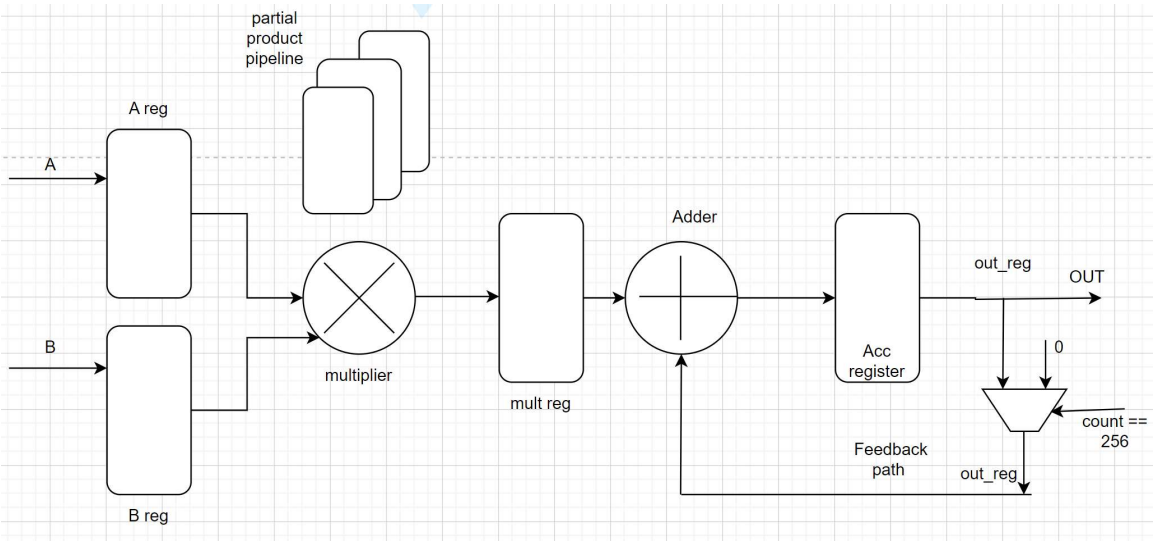

Fig2 : MAC architecture used

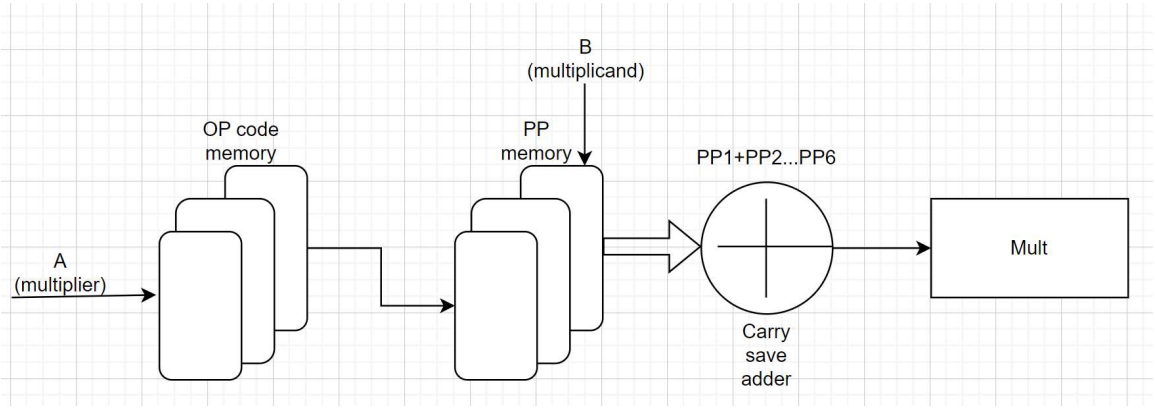

Fig3 : Multiplier block architecture

3 Design parameters

Specify all design parameters which are closely linked with the functional requirements of the block.

1. Input = 16 bits
2. Output = 27 bits

3.1 Performance Requirements

State performance parameters, for example:

- speed/bandwidth requirements
- memory requirements

- latency limits,

Required Clock frequency = 150Mhz (6.66ns)

3.2 Clock Distribution

Documentation must include:
Parameters such as frequency, duty cycle, rise and fall edge characteristics, active edge clock edge, clock skew constraints, where applicable.

Required Clock frequency = 150Mhz (6.66ns)

Sdc command
create_clock clk -period 6.66 -waveform {0 3.33}

```
  Done Checking the design.
Statistics for commands executed by read_sdc:
 "create_clock"            - successful    1 , failed      0 (runtime  0.00)
Total runtime 0.0
```

3.3 Reset

Describe hardware/software resets and their polarity. Include input/output values at reset.

Reset used = Synchronous reset, Active high

3.4 Timing Description

The following information must be included:
- latency,
- update conditions,
- re-timing (low power and normal modes),
- direct path to flip-flop.

Retiming was enabled during optimization

4 Verification Strategy

4.1 Objectives

Briefly describe the design (the name and its main functionality) and the kind of approach you intend to use (dynamic verification, property-checking, emulation, if the environment is completely new or it is a porting from an existing one and any other relevant details).

1. For RTL Verification
   A comprehensive testbench was generated using SystemVerilog

   The verification was done using Constrained randomization for better coverage.

   For Ex:
   rst=0; $urandom_range('h1111, 'hBBBB); B=$urandom_range('h2222, 'h5555);

Trailing bits verification strategy is also used to check for stuck at zero issues. For Ex:
#31 rst=0; A='hFFFF; B='h1111;

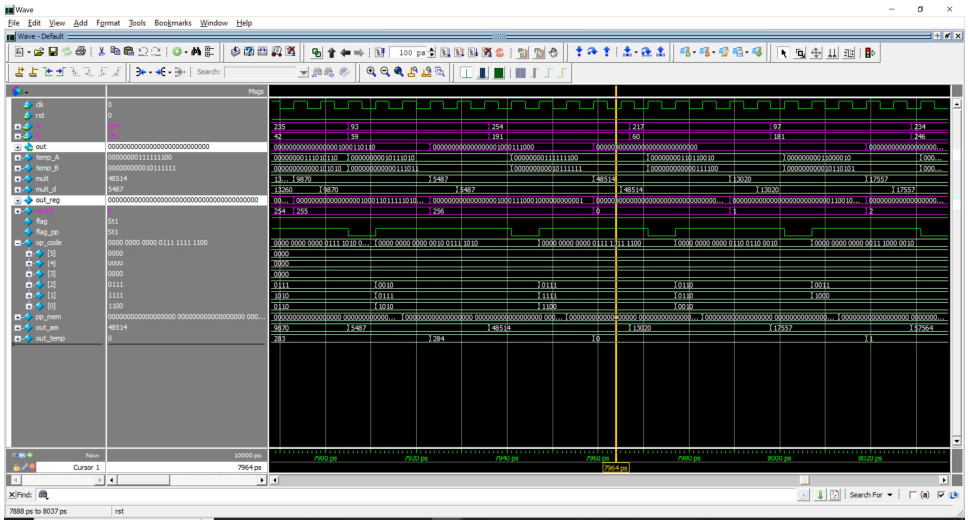

Fig4: Waveform showing the correct counter functionality

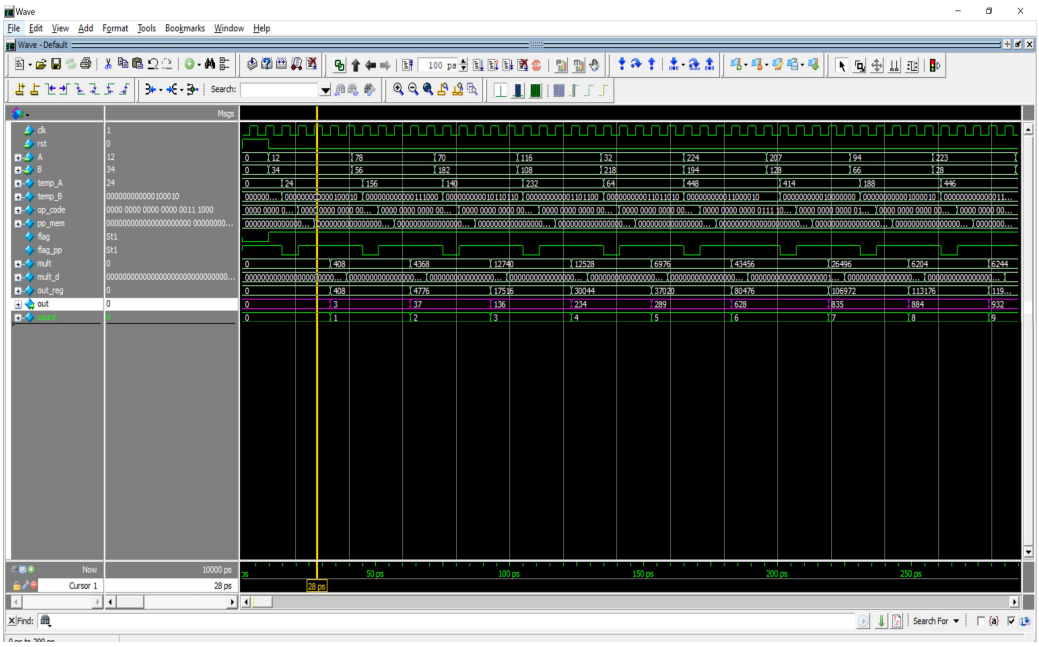

Fig5: Waveform of RTL Verification

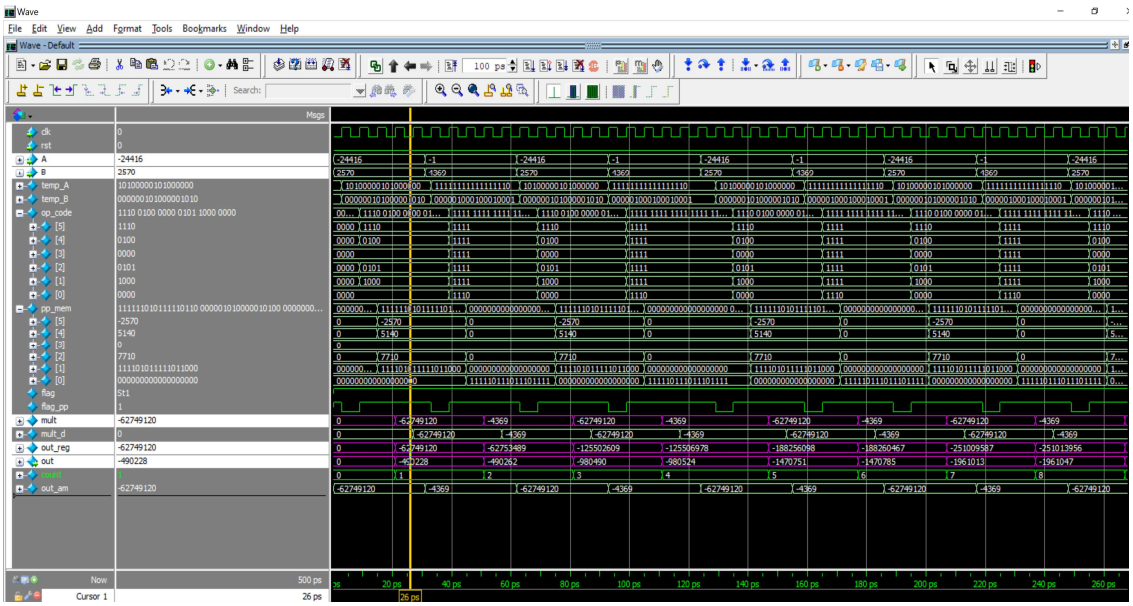

Fig6: Waveform of -ve numbers.

```
    $display($time, "#AM launching fractional values");
    `define FRAC
      for(int i=0; i<5; i++)
      begin
          rst=0; A=16'b0000_0000_0111_1101; B=16'b0000_0000_0111_1010; // a*b * 2^-24 = mult
          #31 rst=0; A=16'b0000_0000_0001_1100; B=16'b0000_0000_0110_1100;
          #31;
      end
```

```
#                  0#AM launching fractional values
#                  0A=0.030518 B=0.029785,  mult = 0.000000, accumulator = 0.000000, out = 0.000000
#                 21A=0.030518 B=0.029785,  mult = 0.000909, accumulator = 0.000909, out = 0.000000
#                 31A=0.006836 B=0.026367,  mult = 0.000909, accumulator = 0.000909, out = 0.000000
#                 51A=0.006836 B=0.026367,  mult = 0.000180, accumulator = 0.001089, out = 0.000977
#                 62A=0.030518 B=0.029785,  mult = 0.000180, accumulator = 0.001089, out = 0.000977
```

Fig: Fraction values

```
$display($time, "#AM launching fractional values");
`define FRAC
  for(int i=0; i<5; i++)
  begin
      rst=0; A=16'b0000_0000_1000_0011; B=16'b0000_0000_1000_0110; // a*b * 2^-24 = mult
      #31 rst=0; A=16'b0000_0000_1110_0100; B=16'b0000_0000_1001_0100;
      #31;
  end
```

```
#                  0#AM launching fractional values
#                  0A=0.031982 B=0.032715,  mult = 0.000000, accumulator = 0.000000, out = 0.000000
#                 21A=0.031982 B=0.032715,  mult = 0.001046, accumulator = 0.001046, out = 0.000977
#                 31A=0.055664 B=0.036133,  mult = 0.001046, accumulator = 0.001046, out = 0.000977
#                 51A=0.055664 B=0.036133,  mult = 0.002011, accumulator = 0.003058, out = 0.002930
#                 62A=0.031982 B=0.032715,  mult = 0.002011, accumulator = 0.003058, out = 0.002930
#                 81A=0.031982 B=0.032715,  mult = 0.001046, accumulator = 0.004104, out = 0.003906
#                 93A=0.055664 B=0.036133,  mult = 0.001046, accumulator = 0.004104, out = 0.003906
```

Fig: Fractional -ve numbers

2. After Layout

   DRC verify was run to check for any DRC issues.

4.2 Tools and Version

Include a brief description of the tools that you plan to use to verify the design, only specifying versions of the tools at the report stage:
_environment
_simulator
_coverage
_verification support tools
_memory models
_compilers/software toolchain

1. Siemens EDA ModelSim 10.5b
2. Cadence Genus 19.12-s121_1
3. Cadence Innovus

4.3 Checking mechanisms

Explain the checking method that you plan to adopt. You should usually say you want to have some sort of self-checking mechanism in place, so that every test is able to report a failure or a success. Just for some (few) cases it could be possible to explain that the visual check of the waveforms had to be used because it was impossible (or too difficult) to put in place a self-checking mechanism. Planning of non self-check tests has obviously to be explicitly agreed with the reviewers and must be highlighted in both the document and the environment.

1. For RTL Verification

A comprehensive testbench was generated using SystemVerilog

The verification was done using Constrained randomization for better coverage.

2. $monitor statement was used to print the data both for fraction and integer values and to check manually

Plan:
Implement self checking mechanism

5 Functional Checklist

Briefly list the main functionality of the design that has to be checked, from a logical point of view:
_interfaces (internal, among blocks, and external, with the testbench),
_protocols (examples: PCI, MPX),
_registers (example: “it is possible to write WO registers, to read RO registers and to read and
write R/W registers”),
_interrupts (we ensure that all the interrupts are raised),
_timings (example: for clock ratios and asynchronous protocols),
_other functions.
This functionality will be detailed in following sub sections.

Main functionality tested

1. OP Code generation – Correct bits were taken from the multiplier to form op code memory – CHECKED.
2. PP generation – Based on the corresponding op code, correct Partial product was generated for addition - CHECKED.
3. Multiplication: The partial products need to be added correctly in order to obtain correct mult output – CHECKED
4. Accumulator operation – The multiplication data is being accumulated and added with previous data in the accumulator – CHECKED.
5. Counter functionality – on counter achieving 256 the accumulator is reset - CHECKED.
6. Truncation and rounding off of Output – Correct bits are truncated, rounded off and sent to output wire. - CHECKED

5 Testbench

5.1 Overview

In this section give a brief explanation of the testbench specification, whether it will use a golden model or not (like COWARE models). Avoid going into many technical implementation details, which could be changed during the verification. This is also the place to highlight eventual criticality (complex harness to be verified stand-alone).

1. For RTL Verification

A comprehensive testbench was generated using SystemVerilog

The verification was done using Constrained randomization for better coverage.

2. $monitor statement was used to print the data and to check manually

5.2 Architecture

At the plan stage explain briefly how do you plan to design your testbench.

It should contain the design under test (DUT) and a number of harnesses and processes.

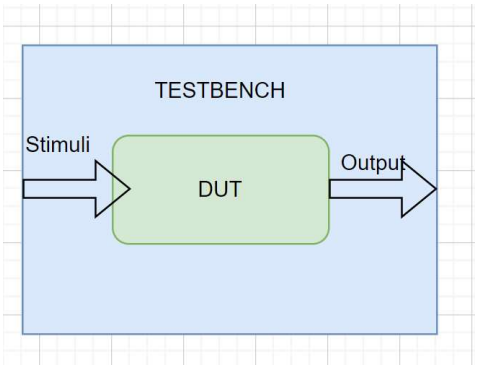

Fig7 : Testbench architecture

6 Tests Specification

List and describe here the functional tests you are going to design.

_Detail here all the tests suite with just a brief description for the tests

1. Constrained randomized tests
2. Unsigned
3. Signed
4. Trailing bits to check for stuck at 0/1 issues.
5. Fractional values

7 Design Microarchitecture

Insert diagram here showing the actual logical design partitioning chosen to meet the functional specification given in associated documentation and listed in the references section above.

7.1 Top Level Interface

All top-level interfaces are listed and described in the functional specification given in associated documentation and listed in the references section above. Please refer the reader to that rather than duplicate the information here - it is more of a maintenance overhead to keep multiple copies of the same information up-to-date.

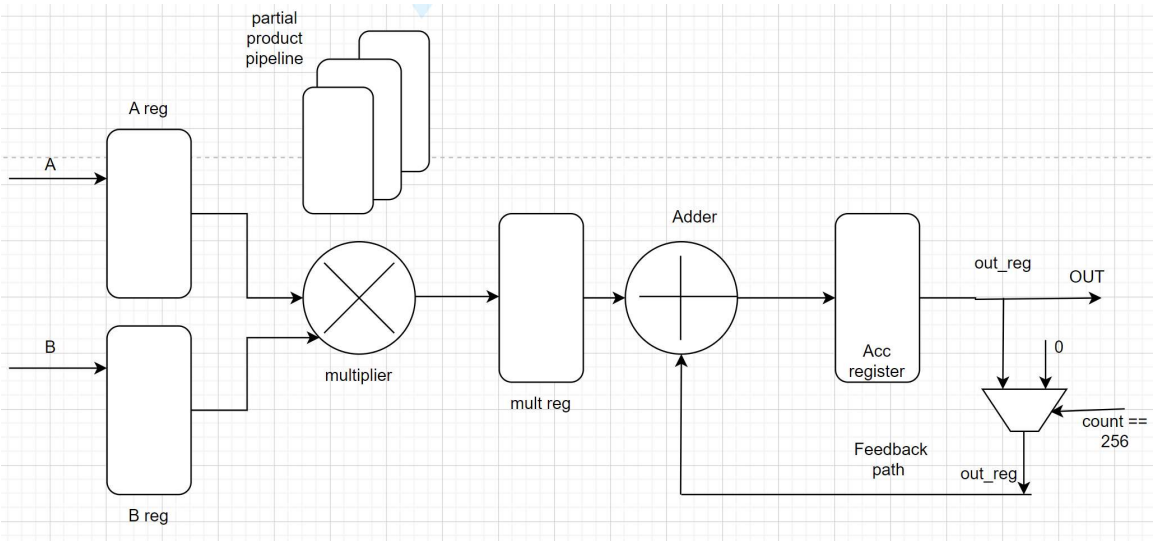

Fig8: MAC architecture used

7.2 Sub-Block Description

Name each sub-block required by the overall design:
For every sub-block, describe logical implementation; use of diagrams of hierarchical partitioning, tables of interface listings, state machine and flow diagrams etc. where appropriate.
Give details of implementation choices. Consider, for example:

- datapath, standard cell
- embedded macros e.g. memories, FIFOs, CAMs
- asynchronous (inter-clock) design
- clock gating
- reset circuitry
- low power control
- special DFT implementation techniques
- any non-standard design circuits or techniques.

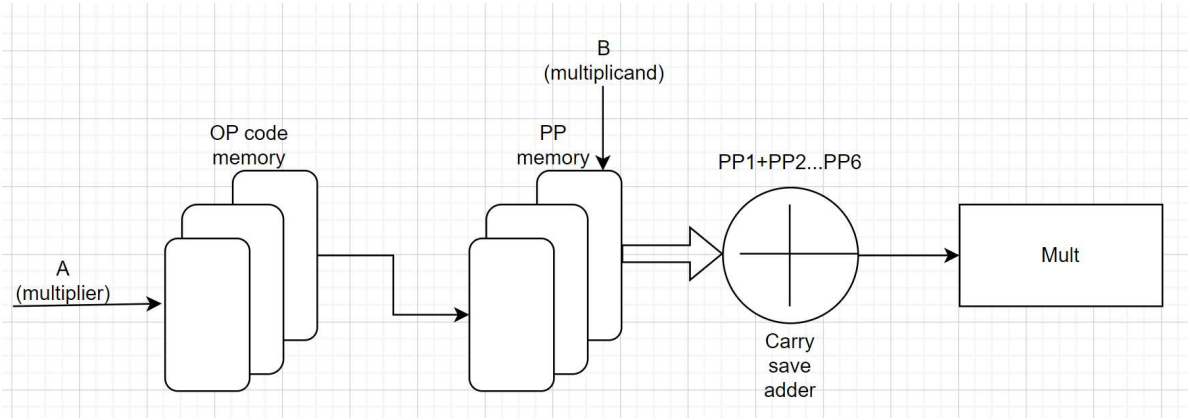

Fig9 : Multiplier block architecture used

7.3 Structural Mapping Process

Describe the process for converting from RTL to gate-level representation of the design including:

logical or physical synthesis
testability insertion and analysis
formal proof that RTL and netlist correspond

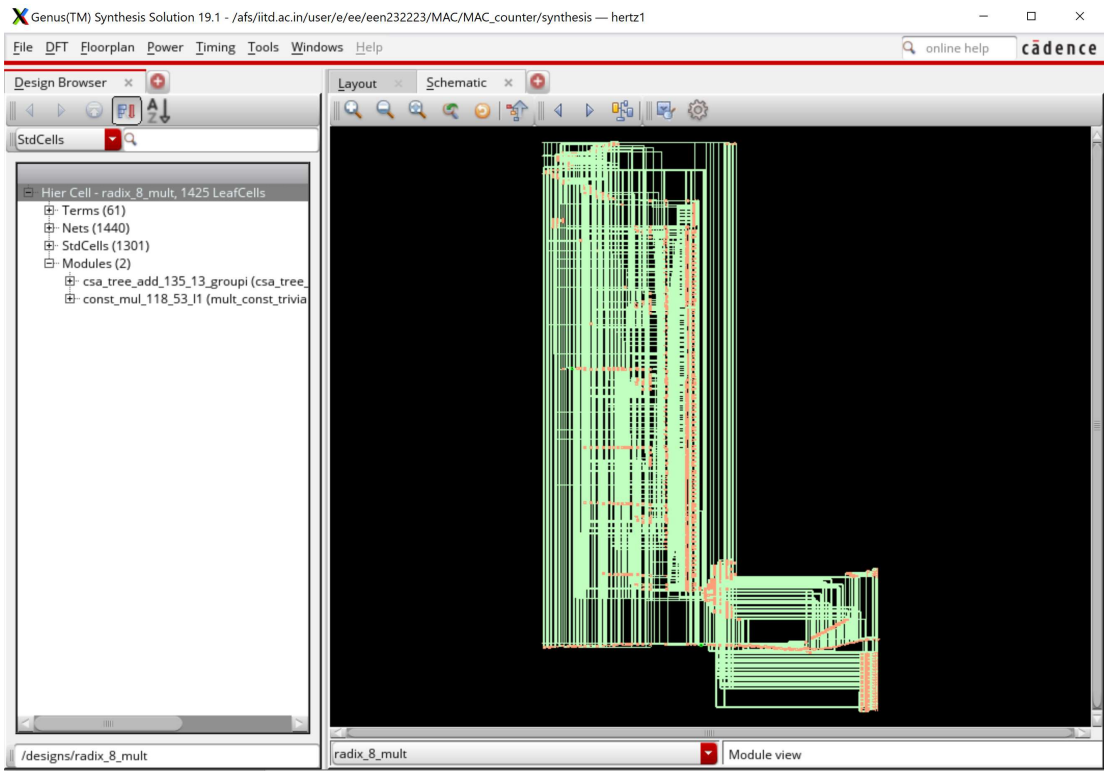

Fig10 : Gate level netlist

8 Physical hierarchy

8.1 Floorplanning

Insert a “plot” here showing the actual floorplan and physical partitioning chosen to meet the design requirements described in Section Design Microarchitecture.
Please give details of abstract size, shape, aspect ratio of dimensions, pin placement criteria, specific placement of hard macros such as RAMs, FIFOs etc., special cell site information or channel routing criteriarequired to resolve layout hot-spots, any special pre-routing of power, critical signals and so on.

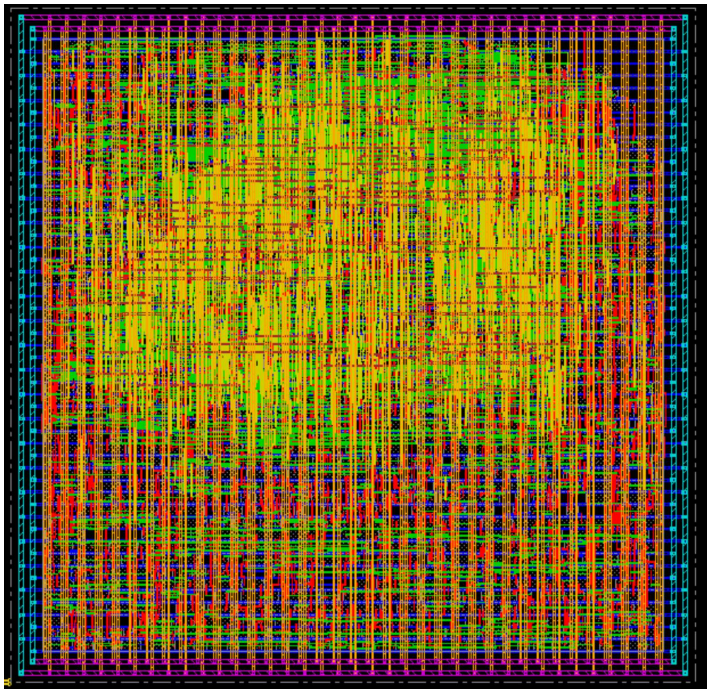

Fig11 : Post Layout design

8.2 Clocktree insertion

Describe here and through reference to the associated functional specification as referenced in Section 7.1: details of the clock strategy and constraints (levels of clock buffering, clock skew and insertion delay targets), relationships between clock domains (if any) and so on.

8.3 Layout Strategy

Describe, either directly here or by reference to a standard methodology document, the process used to go from delivered netlist to completed and verified layout.
Please give details on:

- physical constraints applied
- placement e.g. standard P&R QPlace, Physical Synthesis (or Compilation) etc.
- routing e.g. standard, timing driven, cross-talk avoidance etc.
- physical verification
- generating and delivering extracted parasitics for timing and electrical checking.

9 Results

The purpose of this section is to provide a potential re-user with enough information to determine whether this implementation (as opposed to other implementations) of the block is suitable for their needs; as such, the content should be restricted to the key figures, only giving details where the block does not meet the specification. Do not copy large files and reports into the document; simply highlight or explain areas of importance.

9.1 Area

Give the following information for the block (and its sub-blocks, if relevant); if no layout exists for the block put "N/A" in the last 3 columns.

(Sub)Block Name	ND2-equivalent gate count	% Utilization	Dimensions (WxL)	Layout Area (μm^2)

Depth	Name	#Inst	Area(um^2)
0	radix_8_mult	1404	5418.36
1	csa_tree_add_136_13_groupi	198	806.76
1	const_mul_112_53_I1	17	110.88

Table3: Post CTS Area

10.2 Timing

This section should indicate whether the block meets its timing constraints and how close it is to the original specification. If any constraints are not met, describe them and explain why they have been signed off; describe possible options for speed-up should the block be re-engineered.

Give the cycle time and clock skew of the block.

Design	Slack
Post Synth	**2.929ns**
Pre-CTS - setup	**0.009ns**
Pre-CTS - hold	**-0.009 ns** (within 30% range)
Post-CTS - setup	**0.021ns**
Post-CTS - hold	**-0.009 ns** (within 30% range)

Table2 : Timing numbers

```
------------------------------------------------------------------
Cost Group   : 'clk' (path_group 'clk')
Timing slack :    2929ps
Start-point  : pp_mem_reg[0][3]/CK
End-point    : mult_reg[31]/D

Warning : Timing problems have been detected in this design. [TIM-11]
        : The design is 'radix_8_mult'.
==========================================================
  Generated by:           Genus(TM) Synthesis Solution 19.12-s121_1
  Generated on:           Oct 10 2023  12:56:46 pm
  Module:                 radix_8_mult
  Technology library:     uk65lscllmvbbr_100c25_tc
  Operating conditions:   uk65lscllmvbbr_100c25_tc (balanced_tree)
  Wireload mode:          top
  Area mode:              timing library
==========================================================
```

Fig: Post Synth Slack snippet

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 worst_case

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.009  |  0.009  |  3.603  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   417   |   287   |   253   |
+--------------------+---------+---------+---------+

+---------------+-----------------------------------+------------------+
|               |               Real                |       Total      |
|    DRVs       +------------------+----------------+------------------|
|               |  Nr nets(terms)  | Worst Vio      |  Nr nets(terms)  |
+---------------+------------------+----------------+------------------+
|   max_cap     |      0 (0)       |   0.000        |      0 (0)       |
|   max_tran    |      0 (0)       |   0.000        |      0 (0)       |
|   max_fanout  |      0 (0)       |     0          |      0 (0)       |
|   max_length  |      0 (0)       |     0          |      0 (0)       |
+---------------+------------------+----------------+------------------+

Density: 71.669%
Routing Overflow: 0.00% H and 0.00% V
```

Fig: Pre CTS setup time snippet

```
------------------------------------------------------
          timeDesign Summary
------------------------------------------------------

Hold  views included:
 best_case

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.009  |  0.119  | -0.009  |
|           TNS (ns):| -0.239  |  0.000  | -0.239  |
|    Violating Paths:|   32    |    0    |   32    |
|          All Paths:|   417   |   287   |   253   |
+--------------------+---------+---------+---------+

Density: 71.669%
Routing Overflow: 0.00% H and 0.00% V
```

Fig: Pre CTS hold time snippet

```
------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

Setup views included:
 worst_case

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.021  |  0.021  |  3.577  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   417   |   287   |   253   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 71.583%
Routing Overflow: 0.00% H and 0.00% V
```

Fig: Post CTS setup time snippet

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 best_case

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.009  |  0.120  | -0.009  |
|           TNS (ns):| -0.239  |  0.000  | -0.239  |
|    Violating Paths:|   32    |    0    |   32    |
|          All Paths:|   417   |   287   |   253   |
+--------------------+---------+---------+---------+

Density: 71.583%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

Fig: Post CTS Hold time snippet

POWER UTILIZATION

Total Internal Power:	0.22407079	71.9445%
Total Switching Power:	0.08634351	27.7231%
Total Leakage Power:	0.00103526	0.3324%
Total Power:	0.31144956	

Table3: Post CTS Power utilization

```
Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:        0.22407079            71.9445%
Total Switching Power:       0.08634351            27.7231%
Total Leakage Power:         0.00103526             0.3324%
Total Power:                 0.31144956
-----------------------------------------------------------------------------------------


Group                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
Sequential                       0.1632     0.02058   0.0003411      0.1842       59.13
Macro                                 0           0           0           0           0
IO                                    0           0           0           0           0
Combinational                  0.06083     0.06577   0.0006942      0.1273       40.87
Clock (Combinational)                 0           0           0           0           0
Clock (Sequential)                    0           0           0           0           0
-----------------------------------------------------------------------------------------
Total                            0.2241     0.08634    0.001035      0.3114         100
-----------------------------------------------------------------------------------------
```

Fig: Post CTS Power Utilization snippet

10.3 Testability analysis

Test signals

List all test signals used for this block

Note: this includes signals regarded as a “clock”, e.g. reset and actual clocks.

10.4 DRC rule violations

List any DRC rule violations and explain why they have been retained.

DRC VIOLATIONS – NONE

```
innovus 3> innovus 3> #-report radix_8_mult.drc.rpt           # string, default="", user setting
 *** Starting Verify DRC (MEM: 2676.1) ***

  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area: {0.000 0.000 50.400 50.400} 1 of 4
  VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {50.400 0.000 100.600 50.400} 2 of 4
  VERIFY DRC ..... Sub-Area : 2 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {0.000 50.400 50.400 99.200} 3 of 4
  VERIFY DRC ..... Sub-Area : 3 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {50.400 50.400 100.600 99.200} 4 of 4
  VERIFY DRC ..... Sub-Area : 4 complete 0 Viols.

  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:00.3  ELAPSED TIME: 0.00  MEM: 0.0M) ***
```

Fig12: DRC clean

11 Bugs known at submission date

NONE