Implementation of the Design

The proposed pipelined multiplier design has carry select adder instantiated for the partial product addition, the delay in the computation time for the addition process would be reduced with the help of carry select adder when compared to the ripple carry adder method.

Carry save adder design

A type of Digital adder to efficiently compute the sum of two or more binary numbers. These adders can outperform the computation operation in the multiplication. Instead of using the entire ripple carry adder instantiation in the whole partial product addition process, the carry save adder can be employed to perform the higher bit addition process. The carry save consists of n-full adders, each computes a single sum and carry bit on corresponding bits of three input numbers. The numbers are a, b and c, it produces a partial sum ps and a shift carry sc. The entire sum of the operation can be calculated by the following process. First, Shifting the carry sequence sc left by one place. Second, appending a 0 to the front of the partial sum sequence ps. Third, using a full adder to add these two together and produce the result (n+1) bit value [6].

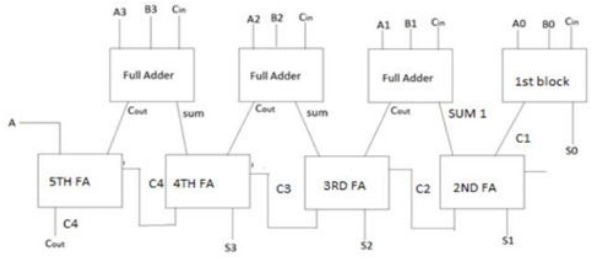

Figure 8: 4-bit carry save adder.

The above Figure shows the instantiated 4-bit Carry Save Adder for the partial product addition computation. The delay response was compared with the corresponding Ripple Carry adder and the Carry Select Adder designs.

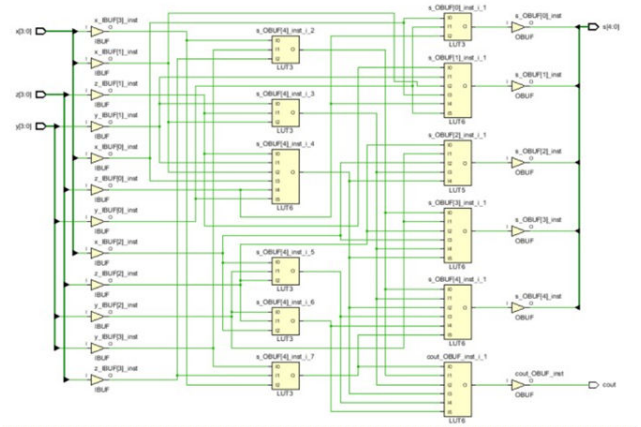

Figure 9: Synthesized schematic design of carry save adder.

Carry select adder

Carry Select Adder is an arithmetic combinational logic unit which sums up two N bit binary sum and 1 bit carry, the main difference in the Carry Select Adder and Ripple Carry Adder is the delay in the propagation of the carry. The Ripple Carry Adder will have longer delay in the carry propagation than the Carry Select Adder.

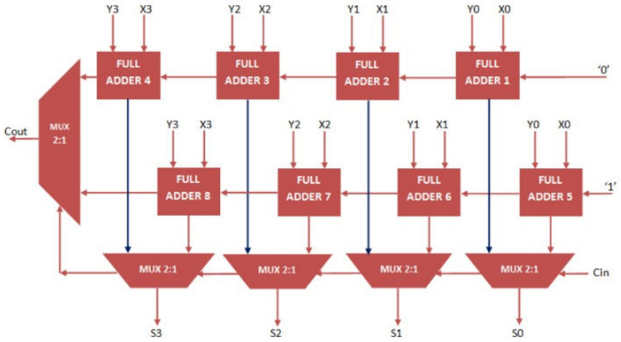

Figure 10: 4-bit carry select adder.

The above figure represents the instantiated 4-bit Carry Select Adder. At the fourth stage of the adder, the propagated carry Cout was taken out and provided as an input to the next stage of the Carry Select Adder.

Pipelined FSM multiplier design

The Proposed pipelined multiplier design with fixed latency FSM, the instantiated adder (Carry Select Adder) in the design based on the delay comparison. The FSM Block controls the Mux and Demux of the Partial product generator blocks.

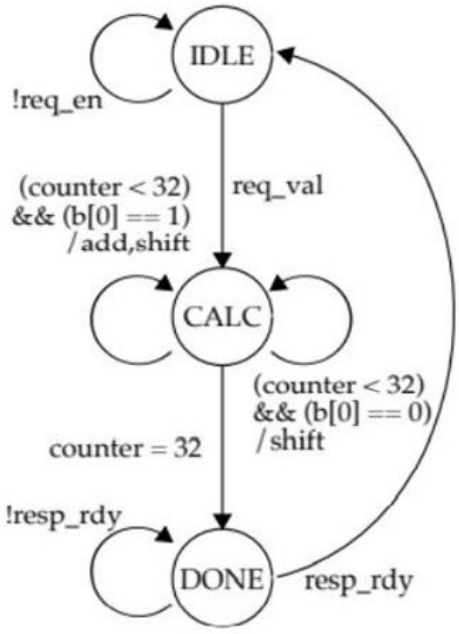

Figure 11: FSM for fixed latency.

The above figure represents the state changes for the FSM (fixed latency), the FSM has a counter to check the repeated shift and add operation. The resp_rdy signal controls the mux select lines. The counter value resets and goes back to the idle state when it reaches 32. The counter remains in Calc state (when counter less than 32) where the multiplication process is done.

Simulation Result

The schematic design of the proposed pipelined multiplier was simulated in the Vivado 2017.4. The proposed multiplier having similar five stages of the existing multiplier, the instantiated adder for the partial product addition is Carry Select Adder (due to less delay in the propagation of the carry).

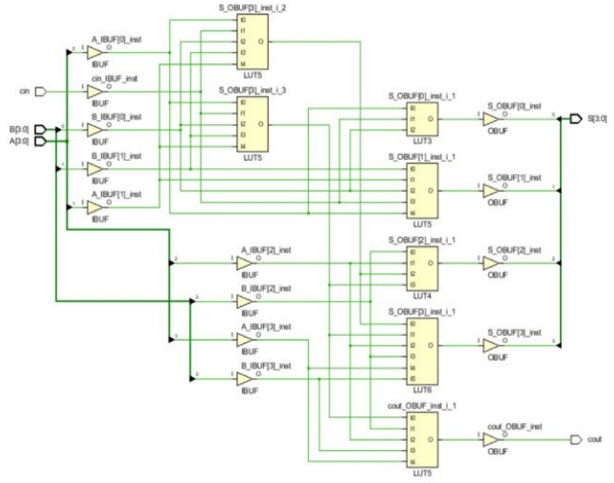

Figure 12: Synthesized schematic design of carry select adder.

The following report shows the delay comparison report of Ripple Carry Adder, Carry Save Adder and Carry Select Adder. The minimum accountable delay should be considered for the design of pipelined multiplier.

```
Path Type:            Max at Slow Process Corner
Data Path Delay:      6.490ns  (logic 3.756ns (57.870%)  route 2.734ns (42.130%))
Logic Levels:         4  (IBUF=1 LUT3=1 LUT6=1 OBUF=1)

  Location            Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                                                  0.000     0.000 r  x[1] (IN)
                      net (fo=0)                  0.000     0.000    x[1]
                      IBUF (Prop_ibuf_I_O)        0.923     0.923 r  x_IBUF[1]_inst/O
                      net (fo=3, unplaced)        0.800     1.722    x_IBUF[1]
                      LUT3 (Prop_lut3_I2_O)       0.124     1.846 r  s_OBUF[4]_inst_i_3/O
                      net (fo=4, unplaced)        1.135     2.981    c1_1
                      LUT6 (Prop_lut6_I1_O)       0.124     3.105 r  s_OBUF[4]_inst_i_1/O
                      net (fo=1, unplaced)        0.800     3.905    s_OBUF[4]
                      OBUF (Prop_obuf_I_O)        2.585     6.490 r  s_OBUF[4]_inst/O
                      net (fo=0)                  0.000     6.490    s[4]
                                                                  r  s[4] (OUT)
```

Figure 13: Timing summary of ripple carry adder.

The overall critical path delay for the ripple carry adder design is 6.49 ns. The reason for the delay is the propagation of the carry in each and every adder stages.

```
Path Type:            Max at Slow Process Corner
Data Path Delay:      5.815ns  (logic 3.756ns (64.587%)  route 2.059ns (35.413%))
Logic Levels:         4  (IBUF=1 LUT3=1 LUT6=1 OBUF=1)

  Location            Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                                                  0.000     0.000 r  in0[2] (IN)
                      net (fo=0)                  0.000     0.000    in0[2]
                      IBUF (Prop_ibuf_I_O)        0.923     0.923 r  in0_IBUF[2]_inst/O
                      net (fo=2, unplaced)        0.800     1.722    in0_IBUF[2]
                      LUT6 (Prop_lut6_I5_O)       0.124     1.846 r  out_OBUF[3]_inst_i_2/O
                      net (fo=2, unplaced)        0.460     2.306    c3
                      LUT3 (Prop_lut3_I0_O)       0.124     2.430 r  cout_OBUF_inst_i_1/O
                      net (fo=1, unplaced)        0.800     3.230    cout_OBUF
                      OBUF (Prop_obuf_I_O)        2.585     5.815 r  cout_OBUF_inst/O
                      net (fo=0)                  0.000     5.815    cout
                                                                  r  cout (OUT)
```

Figure 14: Timing summary of carry select adder.

The above timing summary shows the path delay for Carry Select Adder is 5.815 ns. The delay summary report of the Carry Save Adder design.

```
Path Type:            Max at Slow Process Corner
Data Path Delay:      5.822ns  (logic 3.756ns (64.509%)  route 2.066ns (35.491%))
Logic Levels:         4  (IBUF=1 LUT4=1 LUT5=1 OBUF=1)

  Location            Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                                                  0.000     0.000 r  A[1] (IN)
                      net (fo=0)                  0.000     0.000    A[1]
                      IBUF (Prop_ibuf_I_O)        0.923     0.923 r  A_IBUF[1]_inst/O
                      net (fo=3, unplaced)        0.800     1.722    A_IBUF[1]
                      LUT5 (Prop_lut5_I4_O)       0.124     1.846 r  S_OBUF[3]_inst_i_3/O
                      net (fo=3, unplaced)        0.467     2.313    S_OBUF[3]_inst_i_3_n_0
                      LUT4 (Prop_lut4_I3_O)       0.124     2.437 r  S_OBUF[2]_inst_i_1/O
                      net (fo=1, unplaced)        0.800     3.237    S_OBUF[2]
                      OBUF (Prop_obuf_I_O)        2.585     5.822 r  S_OBUF[2]_inst/O
                      net (fo=0)                  0.000     5.822    S[2]
                                                                  r  S[2] (OUT)
```

Figure 15: Timing summary of carry save adder.

The above timing summary shows the path delay for Carry Save Adder is 5.822 ns. There is a small difference in the delay between Carry Save Adder (5.822 ns) and Carry Select Adder (5.815 ns).

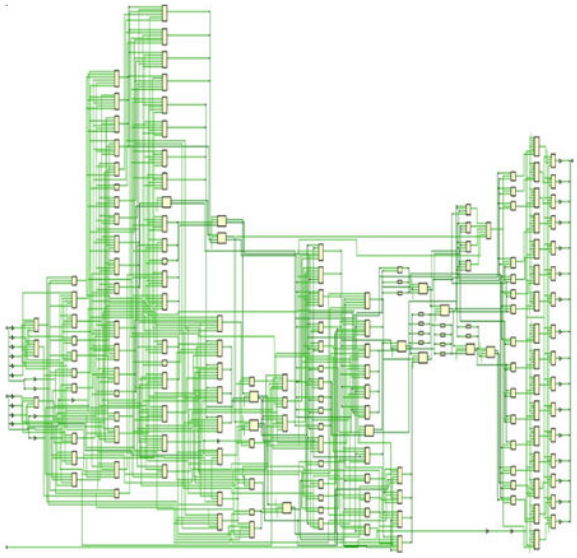

Figure 16: Schematic design of the proposed pipelined multiplier.

The delay reports were compared and thus the Carry Select Adder were instantiated in the Pipelined Multiplier design.

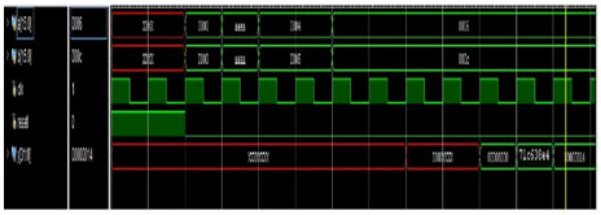

Figure 17: Simulation waveform of pipelined multiplier.

The above figure represents the simulated waveform of the pipelined multiplier, Inputs in Decimal (a-23, b-10, output of the multiplier at 9th cycle-230 in decimal). Timing Constraints include Critical path delay (the maximum delay in the reg-to-reg datapath).

```
Timing constraint: Default OFFSET IN BEFORE for Clock 'f3/x1/out_or00001'
 Total number of paths / destination ports: 768 / 128
--------------------------------------------------------------------------------
Offset:          5.889ns (Levels of Logic 2 5)
  Source:        in2<1> (PAD)
  Destination:   f3/x2/out_2 (LATCH)
  Destination Clock: f3/x1/out_or00001 falling

  Data Path: in2<1> to f3/x2/out_2
                                Gate  Net
    Cell:in->out   fanout  Delay  Delay  Logical Name (Net Name)
    ----------------------------------- ---------
     IBUF:I->O         32  0.821  1.921  in21IBOF (in11IBUF)
     LUT4:I2->O         1  0.551  0.000  f3/x2/Mmux_out_mux0000_935 (f3/x2/Mmux_out_mux0000_935)
     MUXF5:I0 -Y0       1  0.360  0.000  f3/x2/Mmux_out_mux0000_7_f5_34 (f3/x2/Ihour outf1t:20000_7_f535)
     MUXF6:I0           1  0.342  1.140  f3/x2/Mmux_out_mux0000_5_f6_34 (f3/x2/Ihour out_nux0000_5_f635)
     LUT2:I0->O         1  0.551  0.000  f3/x2/sel<3>351 (f3/x2/out_mux0000<9>)
     LD:D               0.203          f3/x2/out_9
    ----------------------------------
    Total                5.889ns (2.828ns logic, 3.061ns route)
                           (48.0% logic, 52.0% route)
```

```
Source:               out_reg[0]/C
                        (rising edge-triggered cell FDRE)
Destination:          out[0]
                        (output port)
Path Group:           (none)
Path Type:            Max at Slow Process Corner
Data Path Delay:      4.016ns  (logic 3.216ns (80.087%)  route 0.800ns (19.913%))
Logic Levels:         2  (FDRE=1 OBUF=1)

  Location            Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                      FDRE                         0.000     0.000 r  out_reg[0]/C
                      FDRE (Prop_fdre_C_Q)         0.456     0.456 r  out_reg[0]/Q
                      net (fo=1, unplaced)         0.800     1.256    out_OBUF[0]
```

Figure 18: Timing report of proposed multiplier.

The above figure shows the timing summary report of the proposed pipelined multiplier, the overall critical path delay for the multiplier is 4.01 ns (Instantiated Carry Select Adder).

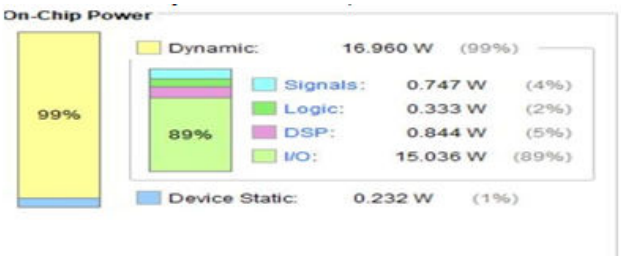

Figure 19: Power summary report of proposed pipelined multiplier.

Observation

From the above Delay comparison table, we can observe that by using Ripple Carry Adder instantiation in the design produces a delay of 7.01 ns, whereas Carry Select Adder produces a delay of 4.01 ns, clearly states that by using Carry Select Adder we can optimize the delay and improve the speed and performance of the design [7]. RCA Pipelined multiplier calculation: The inverse of the Critical path delay (max delay) gives the operating max frequency [8].

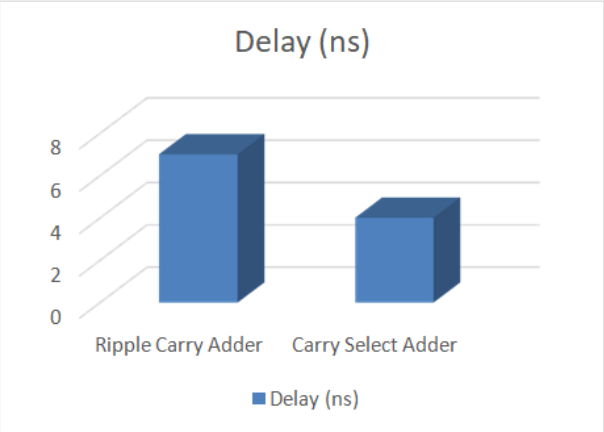

Figure 20: Delay comparison table for ripple carry adder, carry select adder, carry save adder.

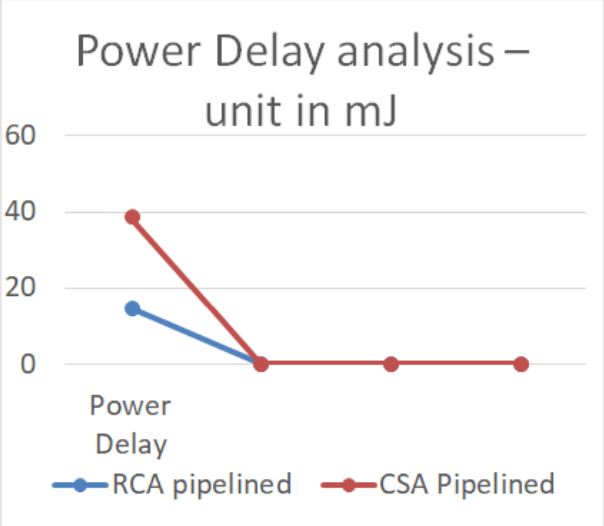

Figure 21: Power delay analysis between RCA and CSA pipelined multiplier.

The speed of the pipelined multiplier can be improved by instantiating Carry Select Adder than Ripple Carry Adder in the partial product addition process. The Power consumption of the overall Pipelined multiplier using Ripple Carry Adder (104 m W), Carry Select Adder (232 mW) [8]. We can clearly observe that there is a tradeoff between Speed and Power (i.e., Delay-Power Tradeoff in the design).

Conclusion

In the presented paper, we have designed 32- bit pipelined multiplier using Carry Select Adder, the main advantage of the Carry Select Adder instantiation is to improve the speed of the partial product addition process. The Carry Select Adder is faster than Ripple Carry Adder instantiation. The overall critical path delay for the proposed design is reduced by 17.21% than the existing path delay. The throughput has been increased by 21.02% than the existing throughput he latency and the clock cycle for the computation of the multiplication process is 9 cycles with 326 MHz frequency. The tradeoff in the proposed design is between Power and Delay. We can go for the proposed pipelined multiplier for high- speed application without the consideration of power usage.

References

1. Arif S, Lal RK (2015) Design and performance analysis of various adder and multiplier circuits using VHDL. Int J Appl Eng Res 10: 17016-17020.
2. Gowthami M, Jalall K, Kiruthika K (2021) High speed and performance analysis of multiplier in field programming gate array. IOP Conference Series: Mater Sci Eng 1084: 012062.
3. McCanny JV, McWhirter JG (1982) Completely iterative, pipelined multiplier array suitable for VLSI. IEE Proceed G-Electr Circuit Syst 129: 40-46.
4. Yan M, De-li W (2010) The design of an architecture-aware 32-bit signed/unsigned multiplier. Int Confer Compu Eng Technol 7: V7-354.
5. Pekmestzi KZ, Kalivas P, Moshopoulos N (2001) Long unsigned number systolic serial multipliers and squarers. IEEE Transact Circuits Syst II: Analog Digital Signal Process 48: 316-321.
6. Di J, Yuan JS, Demara R (2006) Improving power-awareness of pipelined array multipliers using two-dimensional pipeline gating and its application on FIR design. Integration 39: 90-112.
7. Mounica Y, Kumar KN, Veeramachaneni S (2021) Energy efficient signed and unsigned radix 16 booth multiplier design. Comput Electric Eng 90: 106892.
8. Antelo E, Montuschi P, Nannarelli A (2016) Improved 64-bit radix-16 booth multiplier based on partial product array height reduction. IEEE Transact Circuits Syst I: Regular Papers 64: 409-418.