# Low-Level Loop Analysis and Pipelining of Applications mapped to Xilinx FPGAs

Hossein Omidian and Guy Lemieux

FPL conference

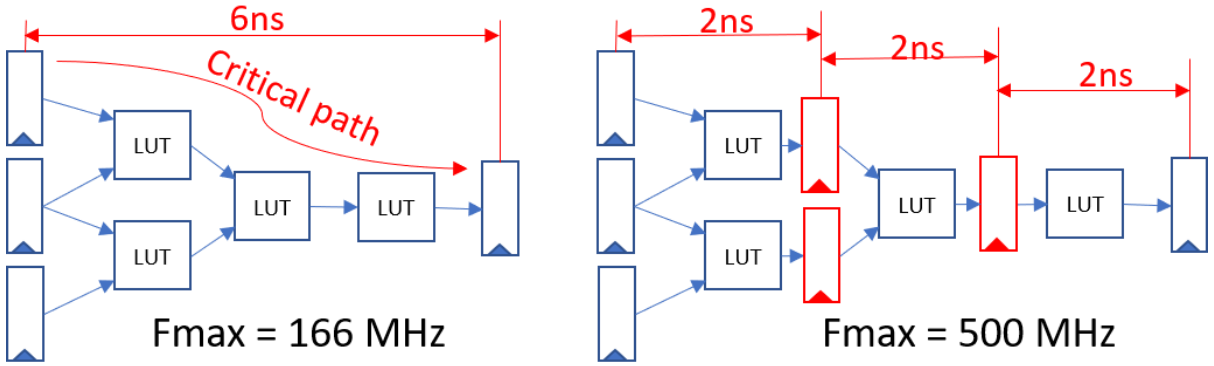
XILINX.

# Problem

## Prior Work:

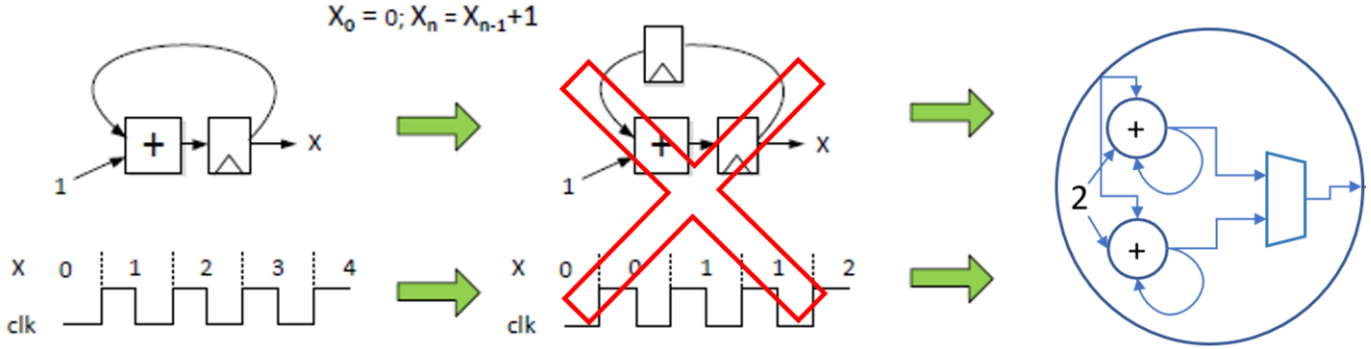
> Feed-forward pipelining is easy

Using report_pipeline_analysis




## This Work:

> Pipelining of sequential loops




**(1) Loop splitting optimization**





XILINX.

# Loops are not always obvious in the HLS/HDL

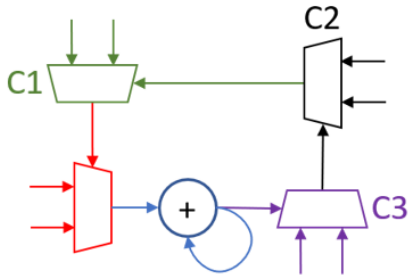
- Different designers working on different parts
- Can be created when using HLS

```
If (C1)
    counter <= counter +1;
```

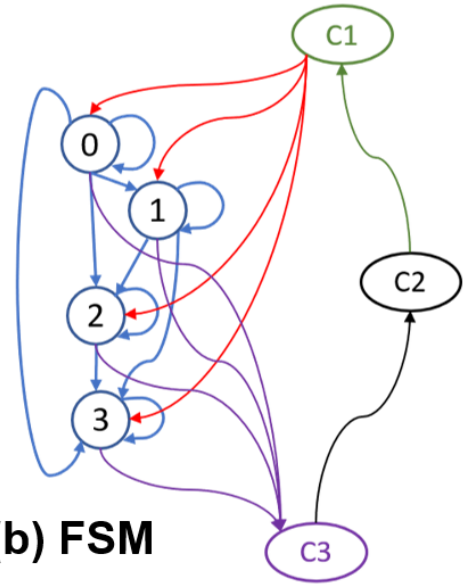
```
always @(posedge clk)
begin
    if (reset)
        C3 <= 1;
    else if ( counter != N)
        C3 <= 0;
    else if
        C3 <=1;
```

```
always @(posedge clk)
begin
    if (reset)
        C2 <= 0;
    else if (C3)
        C2 <= 1;
```

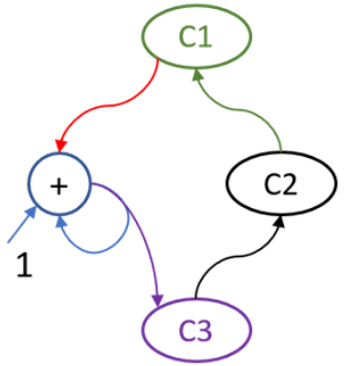
```
always @(posedge clk)
begin
    if (reset)
        C1 <= 0;
    else if (C2)
        C1 <= 1;
```



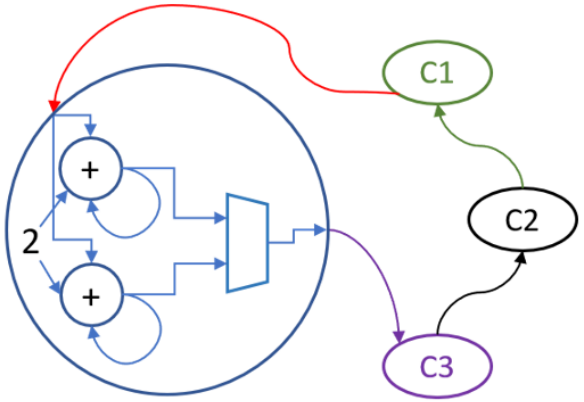

(a) Synthesized netlist




(b) FSM




(c) Simplified FSM




(d) Optimized FSM after loop splitting

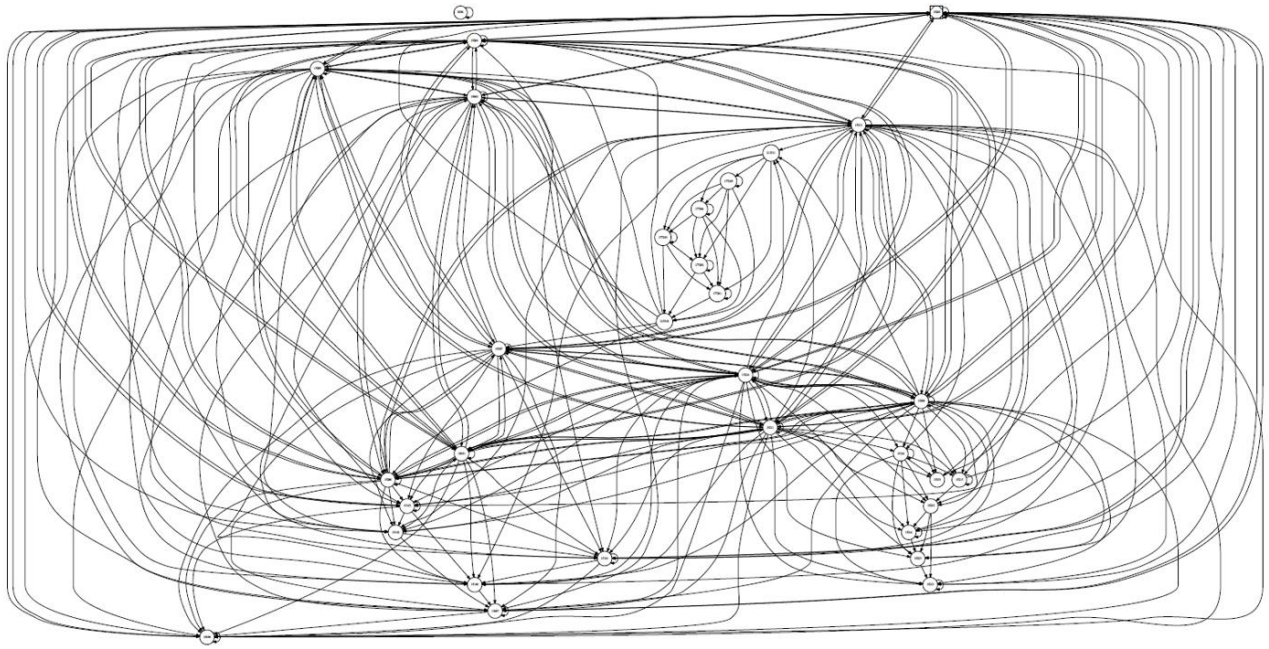
# Loop Analysis

Create a tool which can:

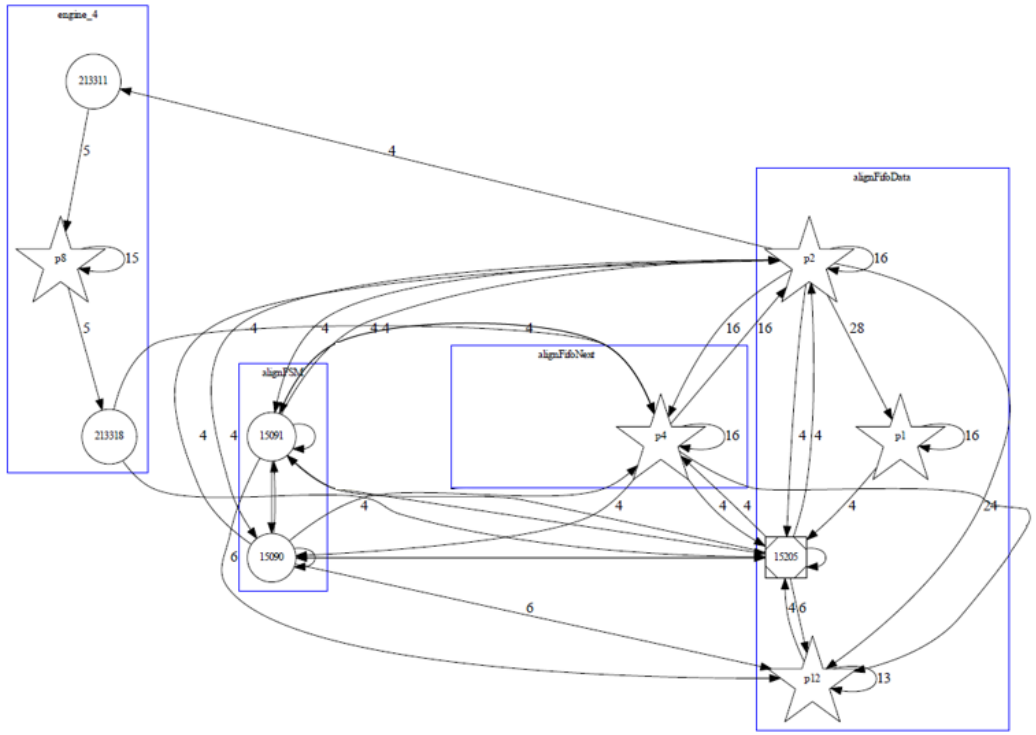
- Locate low-level loops in the netlist (after HLS, after logic synthesis)
- Identify and simplify 11 different loop patterns
- Automatically optimize ~8 loop patterns, user hints for ~3 patterns

| Pattern | Method | Automated Mod. |
|---|---|---|
| 1. Counter | Splitting | Yes |
| 2. Combinational chain | Refactoring | Yes |
| 3. Combinational chain | Combining | Yes |
| 4. FSM | Early condition | Some |
| 5. Triangle Accumulator | Cutting | Yes |
| 6. Combined counters w/ controller | Cutting | Yes |
| 7. Counter with a controller loop | Cutting | Yes |
| 8. CRC lookalike | Combining | Yes |
| 9. FIFO chain | Cutting | Yes |
| 10. FIFO with read/write pointer | Cutting | No |
| 11. FIFOs with a controller | Cutting | No |

# Pattern finding and simplifying



Loop from netlist




Loop simplified by the tool

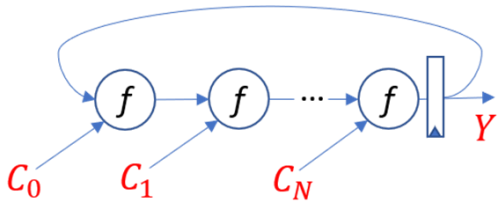




XILINX.

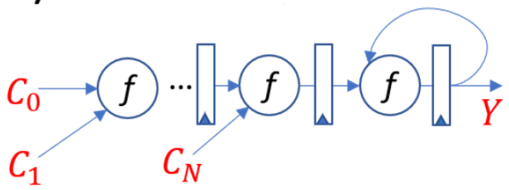
# Techniques

- Splitting
- Associative refactoring
- Combining (C-slow retiming, saves area)
- Early condition



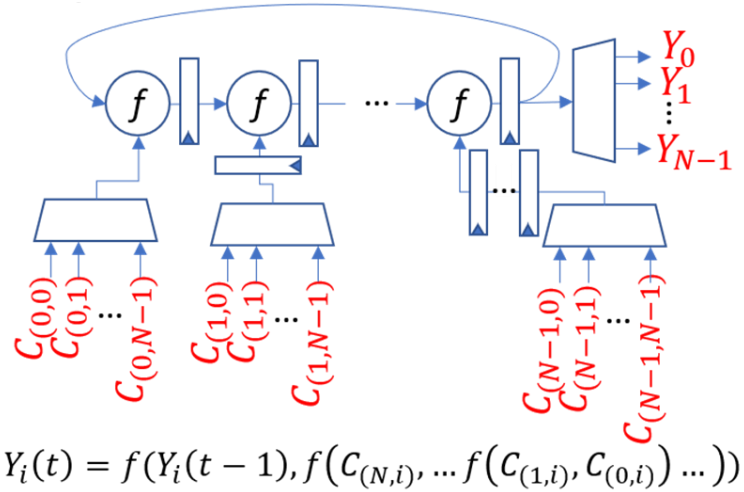

$$Y(t) = f(C_N, f(\dots f(C_1, f(C_0, Y(t-1)))\dots))$$

**a)**

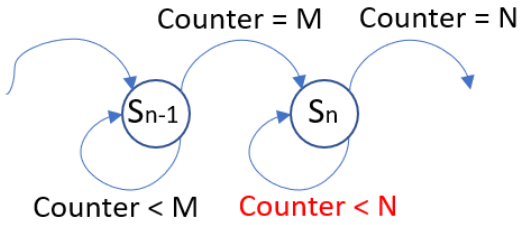



$$Y(t) = f(Y(t-1), f(C_N, \dots f(C_1, C_0)\dots))$$

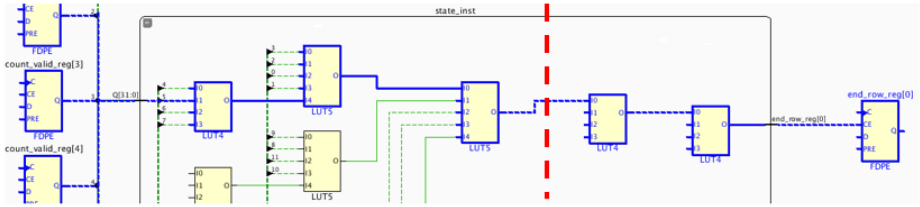
**b)**



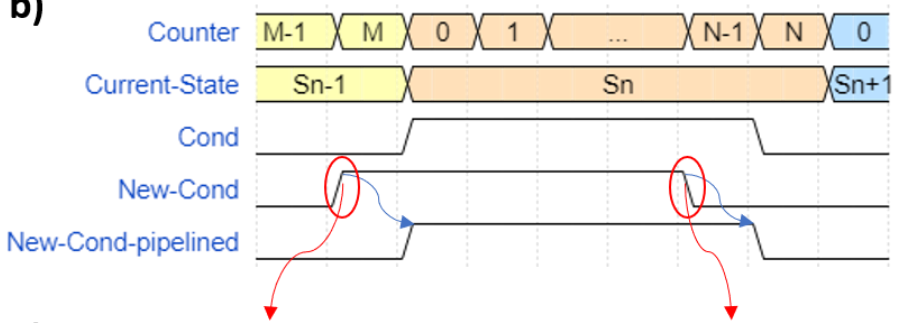

$$Y_i(t) = f(Y_i(t-1), f(C_{(N,i)}, \dots f(C_{(1,i)}, C_{(0,i)})\dots))$$




**a)**



**b)**




**c)** State=Sn-1 and Counter>M-1 State=Sn and Counter<N-1

New condition:
((State = Sn-1 AND Counter > M-1) OR
(State = Sn AND Counter < N-1))

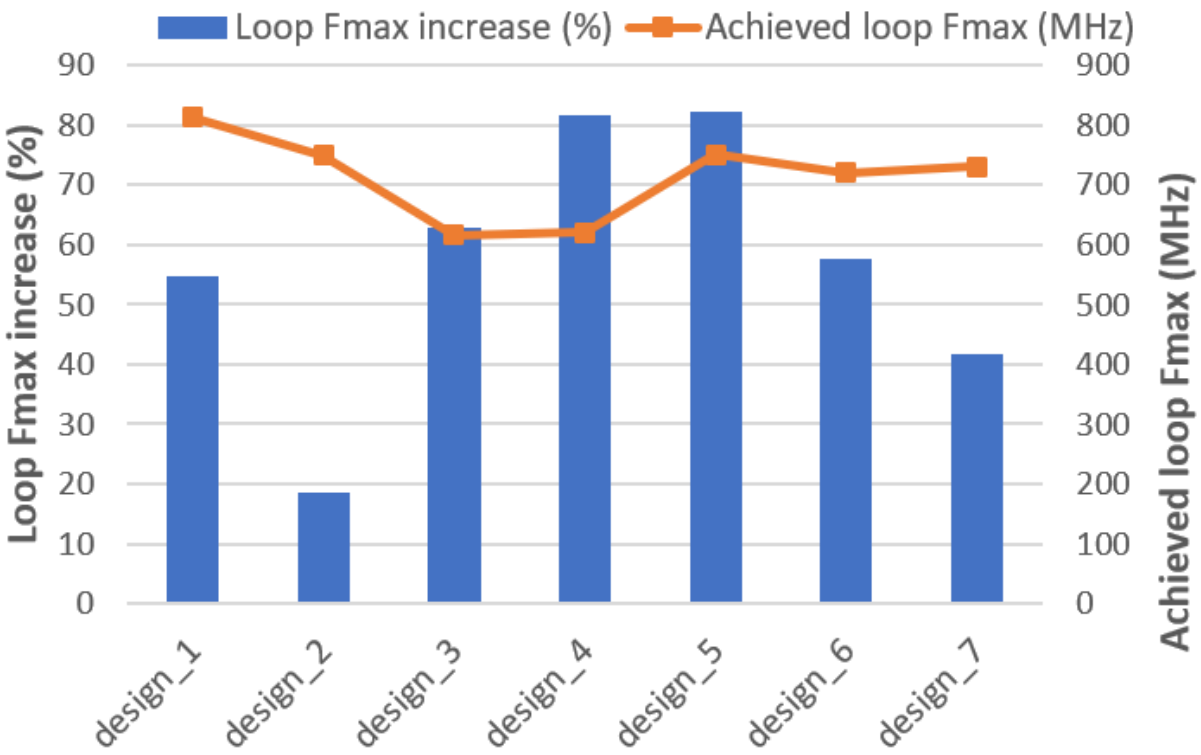




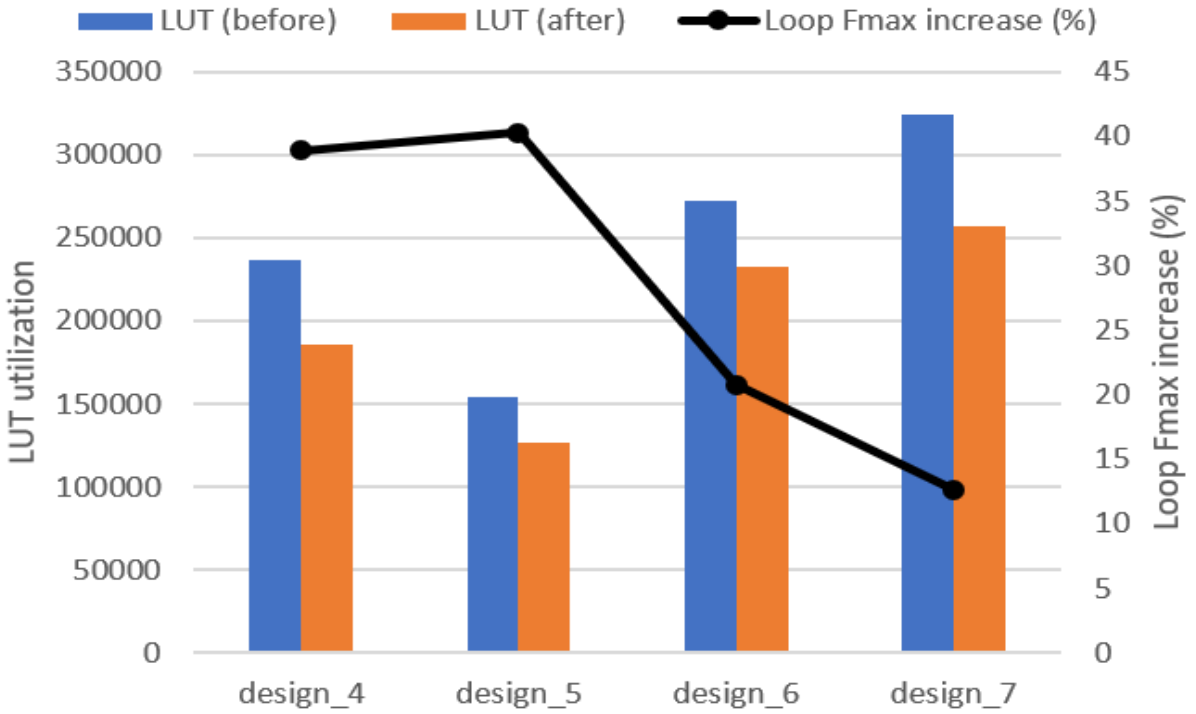
XILINX.

# Results

## Fmax Optimization




## Area+Fmax Optimization




**Applied after physical synthesis process**

- **Fmax optimized: average Fmax of 714MHz (+57%)**
- **Area+Fmax optimized: Fmax +15% to +41%, Average area -18%**

# Adaptable.
# Intelligent.

XILINX