Multi Cycle Paths

**Timing path that is not expected to propagate a signal in one cycle**

This input changes once every 2nd cycle

To undo a `set_multicycle_path` command, use `reset_path` or `reset_design`.
False Paths

You can exclude false paths from an Static Timing Analysis run.

False paths are considered unconstrained.

Pipelining – a fully manual approach

Increases the throughput of designs to meet high timing constraints.

Original design
Too low throughput

Go get better performance – symbolical!

HDL Add some registers

Distribute the registers evenly over the operators

Note: This approach will create an overall larger latency due to the extra registers, but the throughput will increase!
Re-timing – a semi-automatic approach

*Increases the throughput of designs to meet high timing constraints.*

Original design
- Too bad through-put

Go get better performance
- HDL: Add some registers
- In Synopsys:
  - Set your timing constraints
  - Compile your design
  - `dc> balance_registers`

The `balance_registers` will try to distribute the registers evenly over the operators

Note: This approach will create an overall larger delay due to the extra registers but the through-put will increase!

---

Re-timing - Limitations

*Only works on mapped/compiled designs*

No Set or Reset on the registers allowed

"Loop-backs" not possible to Re-time

Not possible to move inside loop
Re-timing – registered outputs

How to get the output registers right?

Desire - We want to keep the output registers
Three possibilities:

1. Group the extra together in the HDL
   - Separate Component
   or Separate Process + group -hd1 in DC

2. Group the combinatorial logic and the extra registers manually in synopsys
   dc> group ("REG2" "REG3" "Op1") \
   -design_name Op1_ret
   dc> current_design Op1_ret
   Set the timing constraints
   dc> balance Registers

3. Put a dont_touch attribute on the last register in Design Compiler after the timing constraints are set and the design is compiled
   dc> set_dont_touch ("REG4")
   dc> balance_registers

Finite State Machines

Standard Mealy Machine:

ENTITY FSM IS
  PORT (Clk : IN BIT;
         Toggle : IN BIT;
         Op : OUT BIT);
END FSM;

ARCHITECTURE FSM_behave OF FSM IS
  TYPE STATE_TYPE IS (ZERO, ONE);
  SIGNAL current_state : STATE_TYPE;
  SIGNAL next_state : STATE_TYPE;
BEGIN
  seq_logic : PROCESS(Clk)
  BEGIN
    IF (Clk'EVENT and Clk = '1') then
      current_state <= next_state;
    END IF;
  END PROCESS;
  comb_logic : PROCESS (current_state, Toggle)
  BEGIN
    Op <= '0';
    next_state <= ONE;
    CASE current_state IS
    WHEN ZERO =>
      IF (Toggle = '0') THEN
        ...
  end process;
END FSM;
Synthesis of Finite State Machines

Idea - Make Synopsys aware of that the logic represents an FSM

Standard Mealy Machine:

ENTITY FSM IS
  PORT (Clk : IN BIT;
         Toggle : IN BIT;
         Op : OUT BIT);
END FSM;

ARCHITECTURE FSM_behave OF FSM IS
  TYPE STATE_TYPE IS (ZERO, ONE);
  SIGNAL current_state : STATE_TYPE;
  SIGNAL next_state : STATE_TYPE;
BEGIN
  seq_logic : PROCESS (Clk)
  BEGIN
    Order is important!
    ATTRIBUTE STATE_VECTOR OF FSM_behave ARCHITECTURE IS "current_state";
  END PROCESS;
END FSM_behave;

Dedicated Synopsys Attributes
Used to identify state-related logic during state encoding extraction

Synthesis flow of FSMs in Synopsys

Compilation

 VHDL
  Compile
  Gates
  State Tables
  FSM compile
  Change State Encodings

Extract

Extraction of the state-vector in a design where the state-vector is the only sequential elements

dc> analyze -f vhdl fsm.vhdl
dc> elaborate fsm
dc> replace_synthetic /* "Lighter" version of compile */
dc> extract
dc> report -fsm
Synthesis Flow of FSMs in Synopsys

- Extraction of the state-vector in a design where the state-vector is not the only sequential elements
  ```
  dc> analyze -f vhdl state_vector.vhdl
  dc> elaborate fsm -arch fsm_behave
  dc> group -fsm -design_name extracted_fsm
  dc> current_design = extracted_fsm
  dc> extract
  dc> report -fsm
  ```

- Extraction of the state-vector in a design where the state-vector attribute is not set in the HDL

- Existing Registers
  ```
  U1:FLIP_FLOP port map (NEXT_STATE[0], CLK, STATE[0]);
  U2:FLIP_FLOP port map (NEXT_STATE[1], CLK, STATE[1]);
  ```

- How to give the register the state attributes
  ```
  set fsm_state_vector { U1, U2 }
  set fsm_encoding { "S0=2#00", "S1=2#01", "S2=2#10", "S3=2#11" }
  ```

- Use the script above for the rest

---

Synthesis flow of FSMs in Synopsys

![Flowchart](image)

**How to set the encoding of the fsm to one-hot and perform an FSM compile**

The script was extracted from command-window after graphical interface manipulations:

```
  dc> set fsm_order { S0 S1 S2 S3 }
  dc> set fsm_encoding [ ]
  dc> set fsm_encoding_style one_hot
  dc> set fsm_order { S0 S1 S2 S3 }
  dc> set fsm_encoding [ "S0=2#0001" "S1=2#0010" "S2=2#0100" "S3=2#1000" ]
  dc> set fsm_minimize true
  dc> compile -map_effort medium
```

For more info: Design Compiler Reference Manual:
Optimization and Timing Analysis, Chapter 4