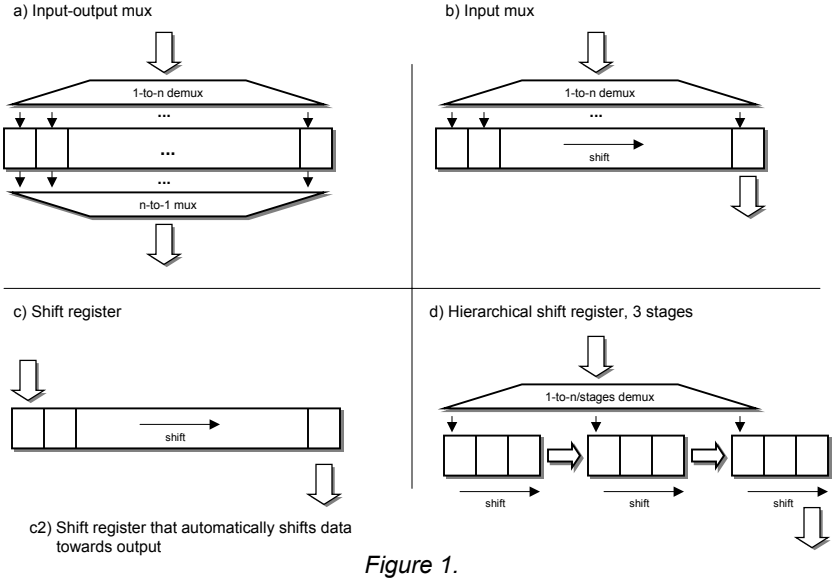
FIFO comparison

- Compare different FIFO implementations
  - area
  - speed
- Three architectures
  - several versions on some architectures
  - two versions of shift register included

FIFO structures



c2) Shift register that automatically shifts data towards output

Figure 1.

FIFO

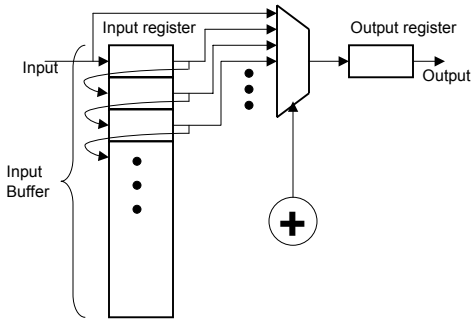


Figure 2.

- Output and Input are always the same registers
- Data is shifted in the input buffer when write occurs
- Mux chooses which data is moved to the output after read

Different FIFOs (1)

- fifo_iom: uses mux in the input and output (Fig1a)
- fifo_im: uses mux in the input (Fig1b)
  - fifo_im_case: implemented as fifo_im except that uses "case" instead of "if".
    - makes it concurrent rather than sequential
  - there is a problem with fifo_im (both) when trying to write data even though fifo is full.
- Fifo_cases (Fig2):
  - tested: doesn't set the output to '0's when last data is read.
  - Result: doesn't have a significant impact on anything

Different FIFOs (2)

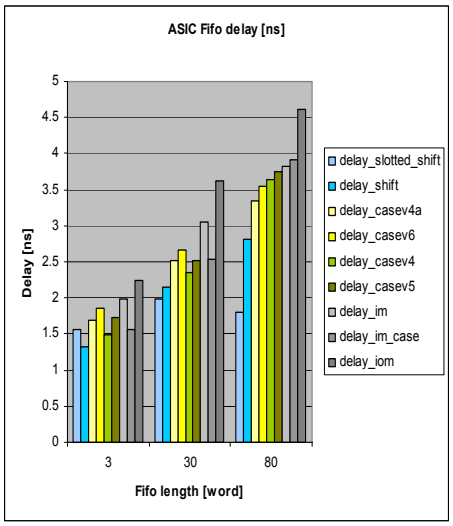
- Fifo_case (all versions)
  - input and output-register is always the same.
  - after the first data the following data is stored to a shift register
- Differencies between fifo_case versions:
  - v3 : bug discovered - don't use
  - v4 : uses a variable for Data_amount to update control signals (Empty, Full, etc.).
    - register inferred from variable
    - v4a : same as v4 but doesn't reset to zero any register. only control + data_amount
  - v5 : signal is used for Data_amount
    - v6 : same as v5 but doesn't reset buffer and output registers
  - Use either v4 or v5 which have a proper reset!
    - v5 seems more orthodox

Synthesis results

- Uses double_fifo_muxed_read components which include two FIFOs and a read multiplexer
- FIFOs are 3b wide
- ASIC
  - ST microelectronics 0.18 um
  - Synopsys Design Compiler
- Synthesis for FPGA
  - used Mentor Graphics Precision RTL Synthesis 2003a.29
  - Synthesized for Altera Excalibur ARM, frequency 20 MHz

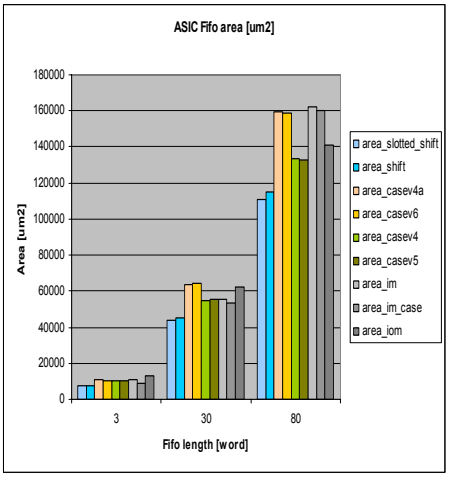
ASIC Delay 0.18 um

- Sorted according to delay of 80 units long fifo
- casev4 fastest fifo
- iom slowest fifo
  - im faster than iom



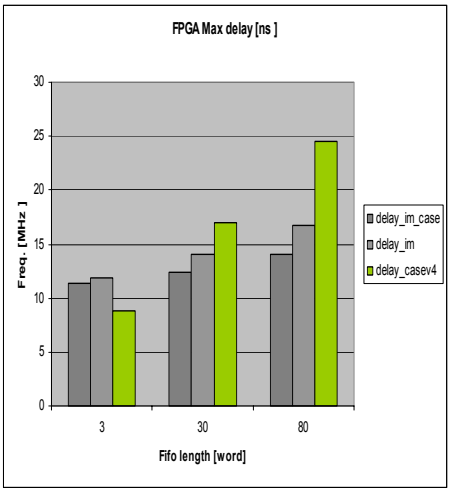
ASIC Area 0.18 um

- Sorted according to delay of 80 units long fifo
- casev5 smallest fifo
- casev4/im biggest
- Not reseting buffer registers increases area



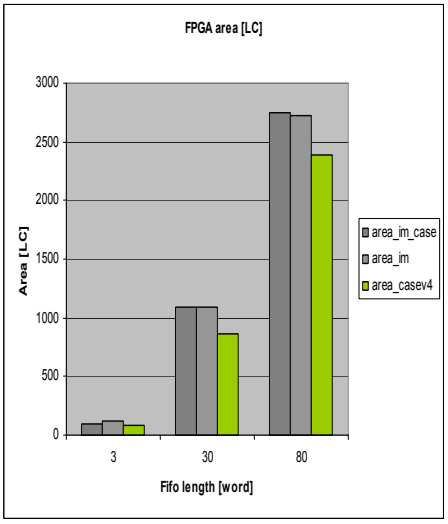
FPGA Delay

- Sorted according to delay of 80 units long fifo
- case bigger than muxed fifo

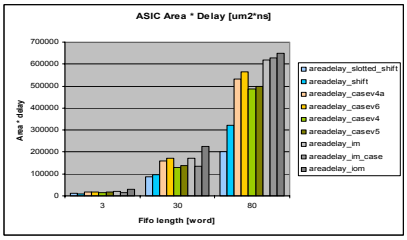


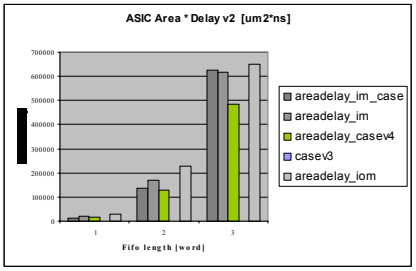
FPGA Area

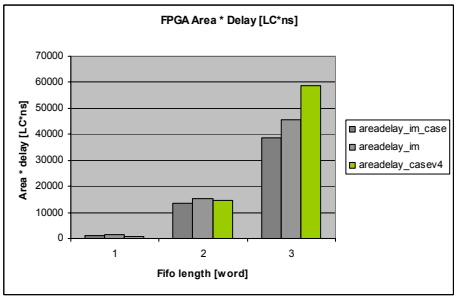
- Sorted according to delay of 80 units long fifo
- case smaller than muxed fifo



Area * Delay







Conclusion

- Shift register much smaller and faster than FIFO
- Areas does not change much with coding style
  - Delay is the main concern
- ASIC and FPGA favor different FIFOs
  - ASIC : *casev4* and FPGA: *im_case*
  - *casev4* inferres register with variable, scary...
- **im_case seems best** (perhaps, probably...)

ASIC

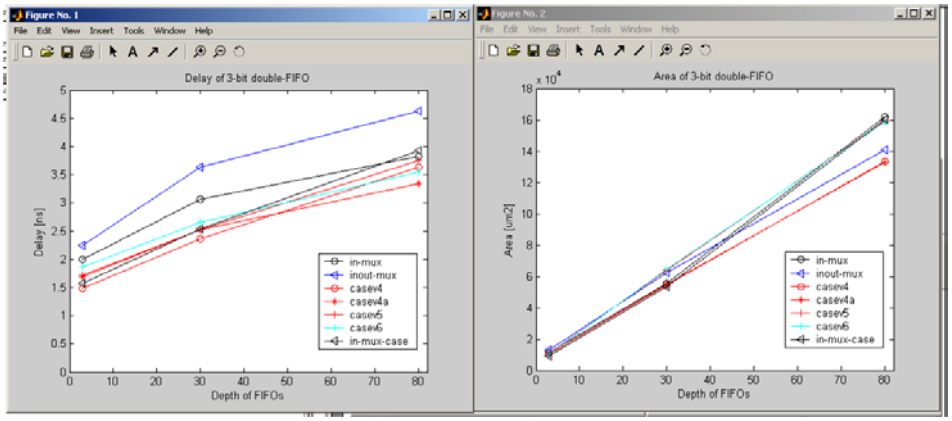


Figure No. 1
Delay of 3-bit double-FIFO
Delay [ns]
Depth of FIFOs
in-mux
inout-mux
casev4
casev4a
casev5
casev6
in-mux-case
Figure No. 2
Area of 3-bit double-FIFO
x 10^4
Area [um2]
Depth of FIFOs
in-mux
inout-mux
casev4
casev4a
casev5
casev6
in-mux-case

FPGA

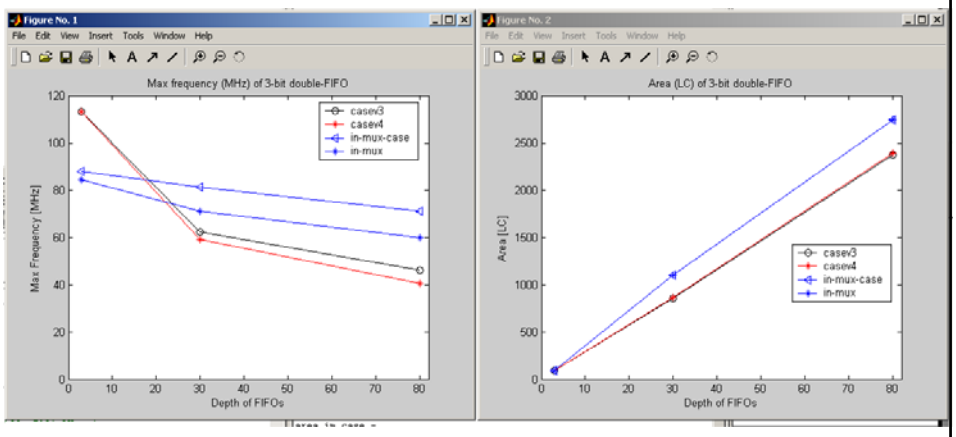


Figure No. 1
Max frequency (MHz) of 3-bit double-FIFO
Max Frequency [MHz]
Depth of FIFOs
casev3
casev4
in-mux-case
in-mux
Figure No. 2
Area (LC) of 3-bit double-FIFO
Area [LC]
Depth of FIFOs
casev3
casev4
in-mux-case
in-mux