FIR IP (FIR_DA_full_v4)

Figure 1 shows the FIR DA IP with its I/Os and parameters. 'sclr' clears the input register chain of the 1D FIR core. 'rst' clears the shift register for 'v' (to protect it after PR). 'v' is output valid. Note that usually 'v' is a delayed version of 'E', except when E and sclr are asserted at the same time.
It also allows for the **antisymmetric mode** (SYMMETRY = AYES).

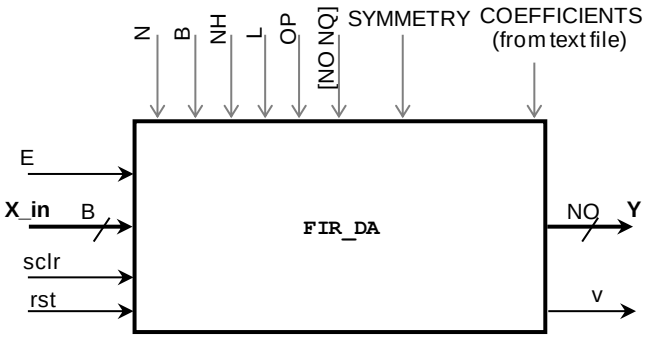


Figure 1. FIR Filter IP

FSL INTERFACE (fsl_ip_gen_v5) in Virtex-4

Fig. 2 shows the FSL interface for the real filter IP. We support 3 I/O cases: B=NO=8, B=NO=16, and B=8,NO=16. Also, 4 modes are available: STREAM, IMG, FILT, and CONV. NX is the size of the input stream (not applicable to STREAM mode). The state machine that controls the DPR process is also shown. The state machine for the FSL bus is inside the PRR and varies depending on the mode (see later).

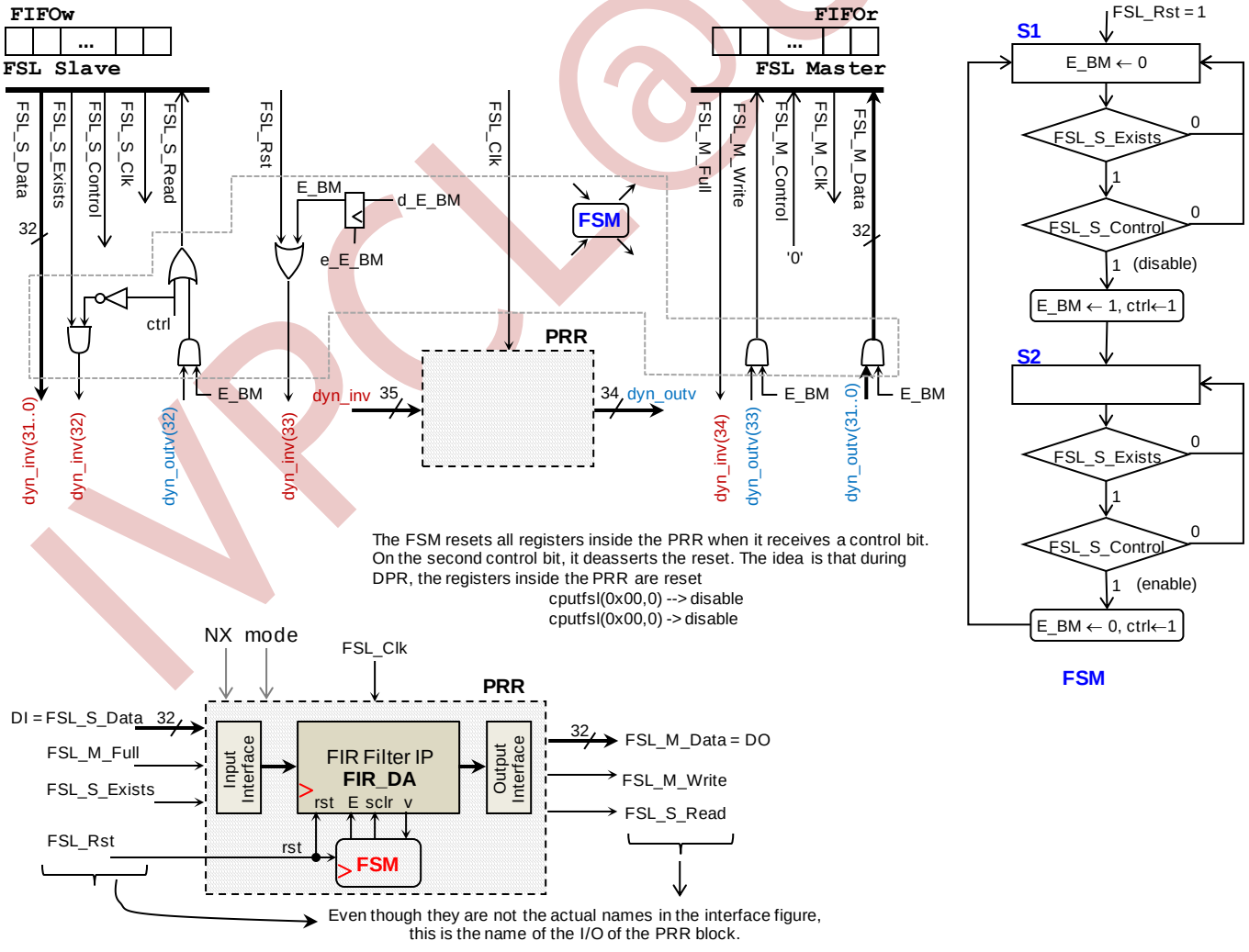


Figure 2. FSL interface for real filter IP

FIR IP (FIR_DA_full_v3)

Figure 1 shows the FIR DA IP with its I/Os and parameters. 'sclr' clears the input register chain of the 1D FIR core. It is not the last FIR IP (FIR_DA_full_v4), so it does not accept ANTISYMMETRY and it does not have a valid output. This is to be updated.

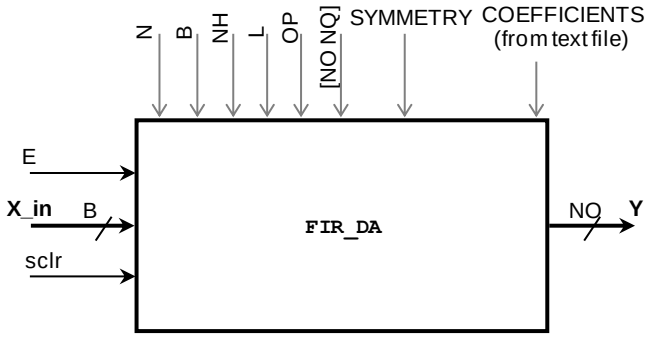


Figure 1. FIR Filter IP

FSL INTERFACE (fsl_ip_gen_v4)

Fig. 2 shows the FSL interface for the real filter IP. We support 3 I/O cases: B=NO=8, B=NO=16, and B=8,NO=16. Also, 4 modes are available: STREAM, IMG, FILT, and CONV. NX is the size of the input stream (not applicable to STREAM mode). The state machine that controls the DPR process is also shown. The state machine for the FSL bus is inside the PRR and varies depending on the mode (see later).

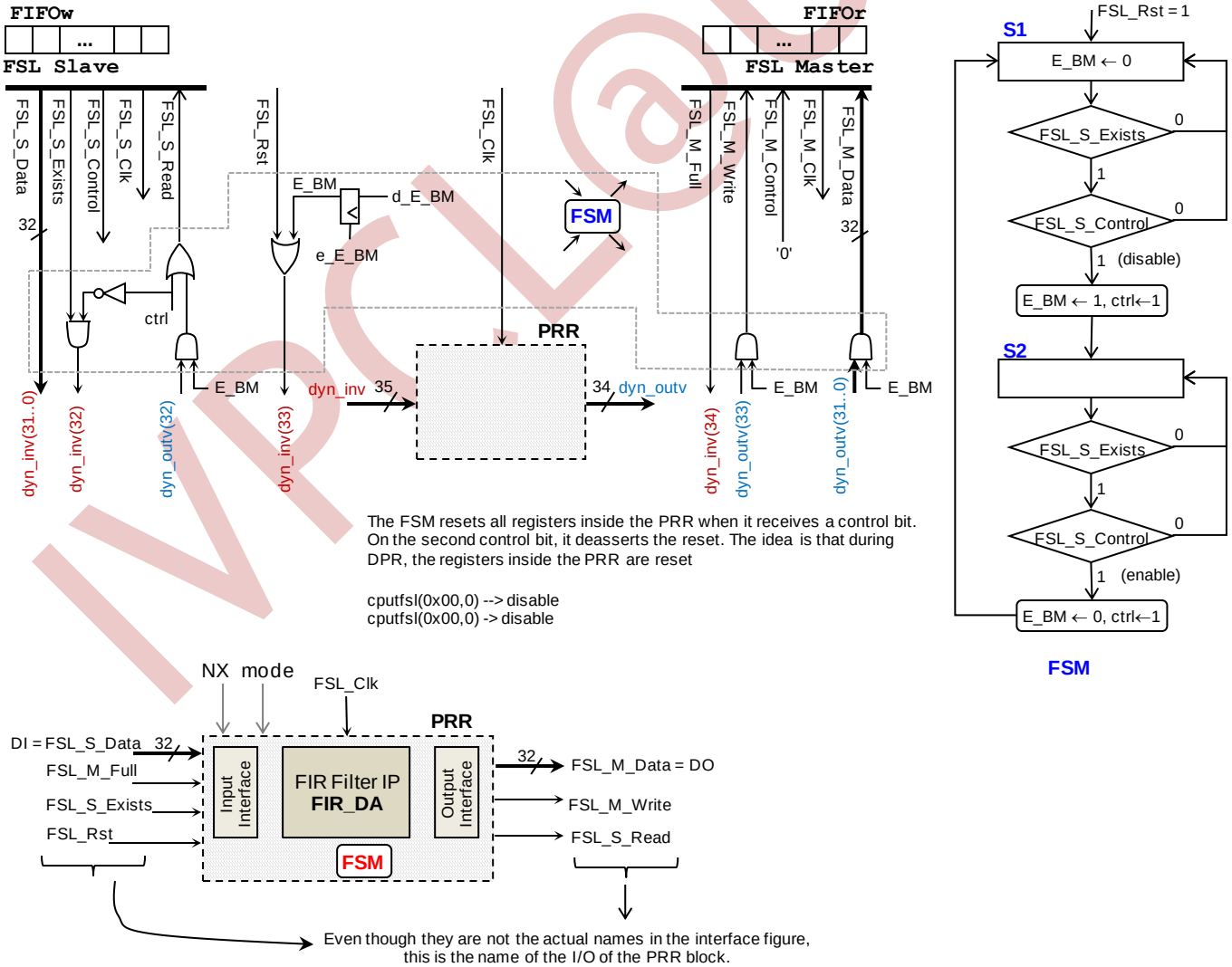


Figure 2. FSL interface for real filter IP

There are 4 modes: STREAM, IMG, FILT, CONV. Each mode requires a different State Machine.

$NWI = \frac{32}{B}$ → # of input real pixels contained in a 32-bit word

$NWO = \frac{32}{NO}$ # → # of output real pixels contained in a 32-bit word

Fig. 3 shows the 3 I/O cases in detail for the modes 'STREAM' and 'FILT'. It also shows how some signals are generated. A state machine is in charge of controlling all these signals.
Ir also shows the real pixel representation. If B=8, we can fit four real pixels in a 32-bit word. If B=16, we can only fit 2 real pixels.

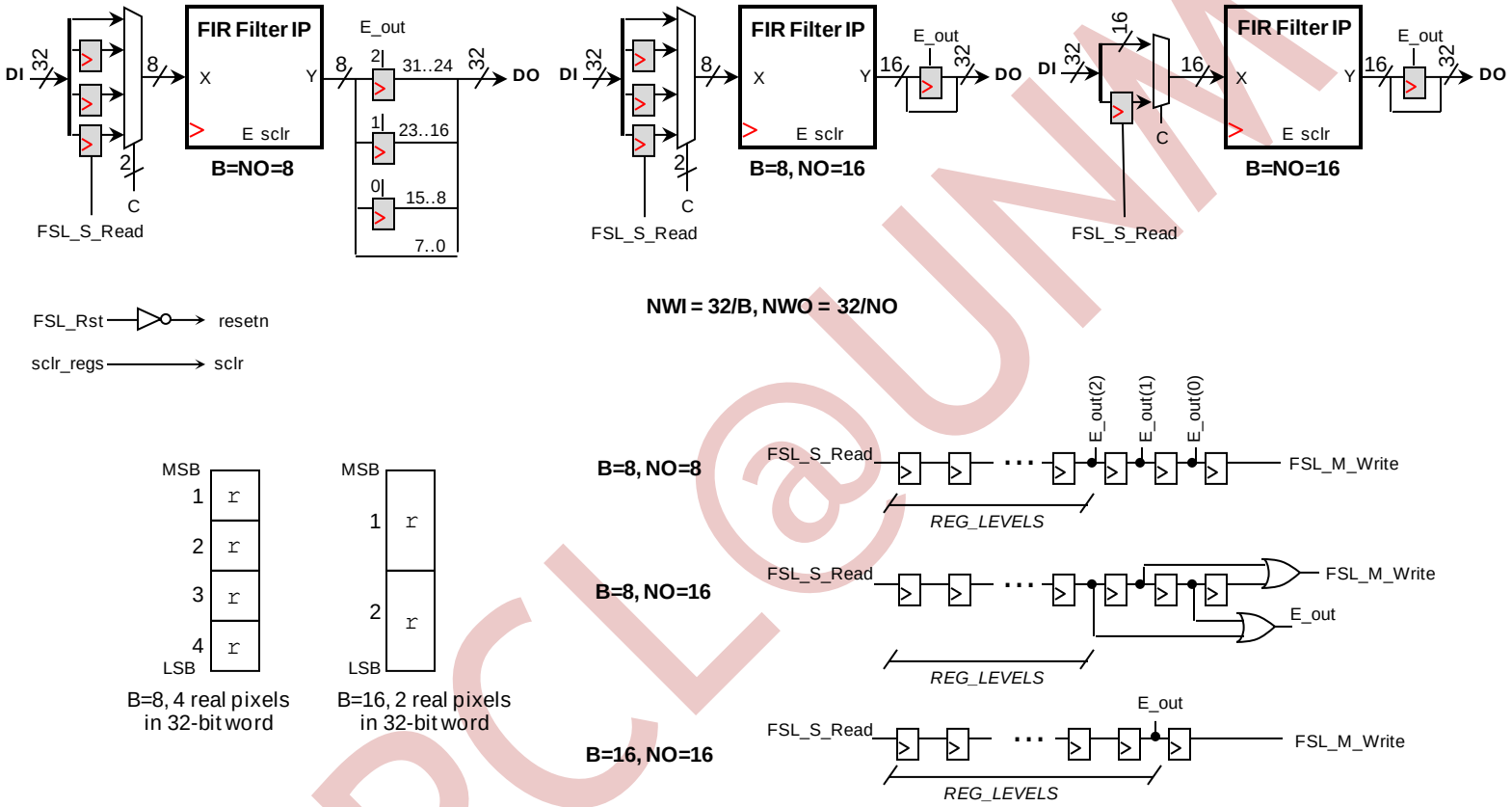


Figure 3. FSL interface. 3 I/O cases. Modes: 'STREAM', 'FILT'

`sclr`: It ONLY clears the input register chain.
* ML405: the external reset is low-level ('resetn')

DYNAMIC REGION (PRR) I/O:

`dyn_inv(31..0) ← FSL_S_Data`	`FSL_M_Data ← dyn_outv(31..0) and E_BM`
`dyn_inv(32) ← FSL_S_Exists and not(ctrl)`	`FSL_S_Read ← dyn_outv(32) and E_BM or ctrl`
`dyn_inv(33) ← FSL_Rst or E_BM`	`FSL_M_Write ← dyn_outv(33) and E_BM`
`dyn_inv(34) ← FSL_M_Full`	

Fig. 4 shows the 3 I/O cases in detail for the modes 'IMG' and 'CONV'. It also shows how some signals are generated. A state machine is in charge of controlling all these signals.

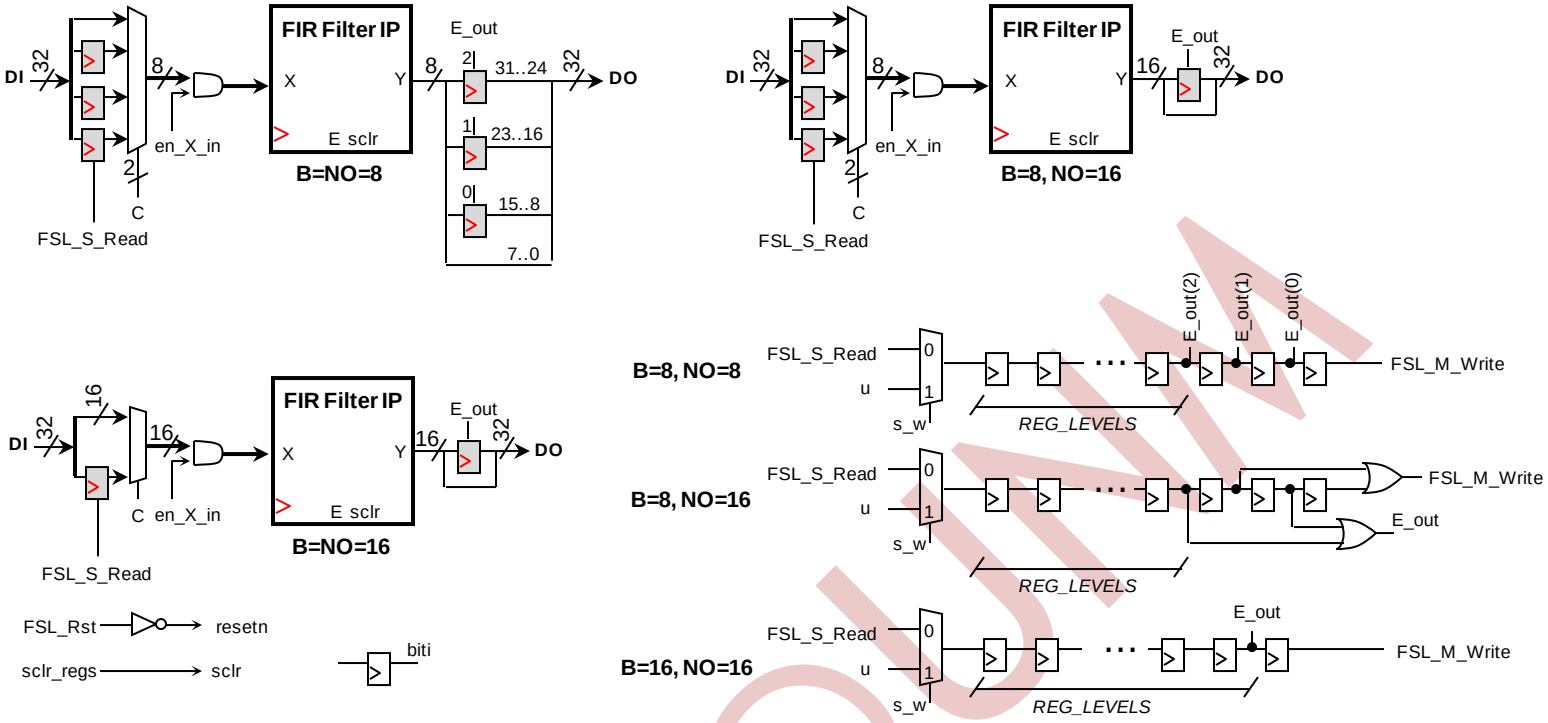


Figure 4. FSL interface. 3 I/O cases. Modes: 'IMG', 'CONV'

FSM for the mode STREAM

Fig. 5 shows the FSM for the STREAM mode. Note that 'sclr_regs' is the same signal as '`sclr`'. The choice of '`sclr_regs`' is so that we are actually clearing the register chain inside the FIR filters.
NWI > 1: If B = 8 → NWI = 4. If B = 16 → NWI = 2

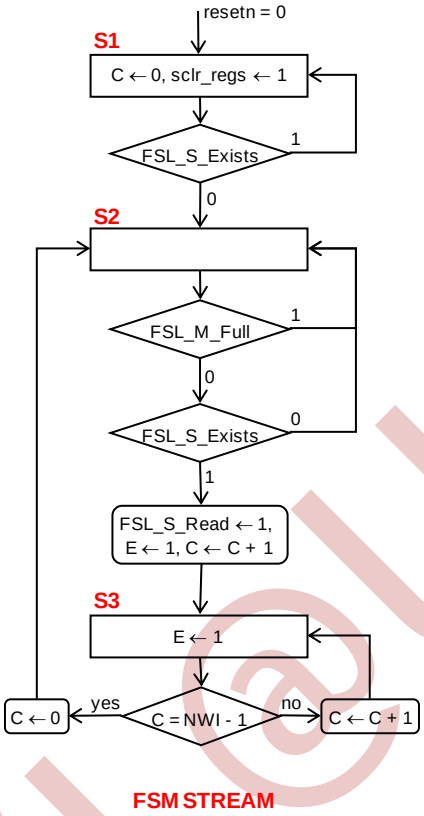
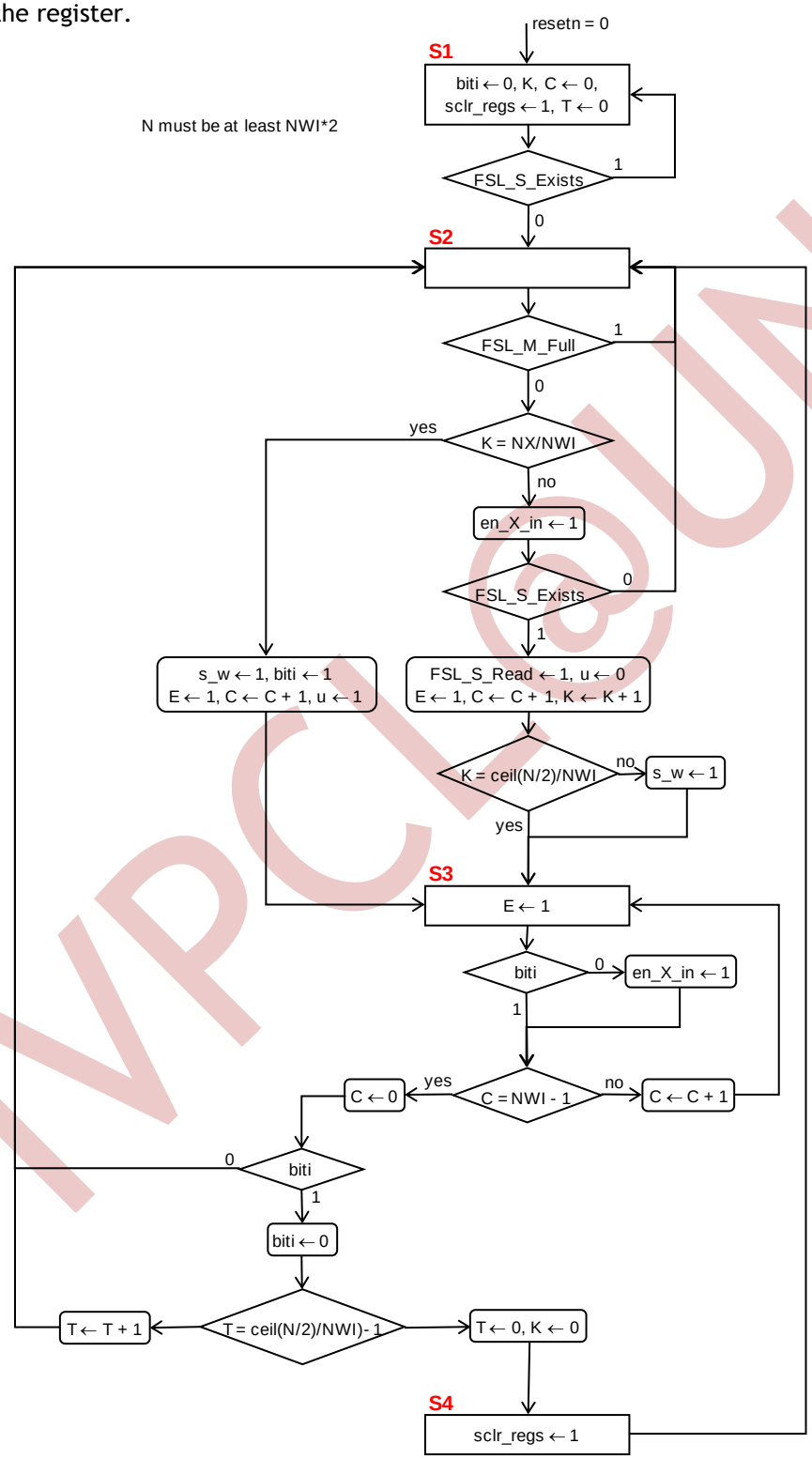


Figure 5. FSM for the STREAM mode.

FSMs for the mode IMG

Fig. 6 shows the FSM for the IMG mode. This FSM controls the input side. '`sclr_regs`' only clears the register chain inside the FIR filter (it does not clear other registers). Constraints: *NX* has to be a multiple of *NWI*; N must be at least NWI*2.

Since in this mode, the filter runs on a stream-by-stream basis, we need to clear the register chain after a stream of length *NX* has been processed. We have to let the last zero-valued pixel enter the chain, that is why we clear the chain after the last zero-valued pixel enters the chain. This is why we need a new state just for clearing the register.



FSM IMG

Figure 6. FSM for the IMG mode (input side)

FSM for the mode FILT

Fig. 7 shows the FSM for the FILT mode. '`sclr_regs`' only clears the register chain inside the FIR filter (it does not clear other registers). The only constraint is that *NX* has to be a multiple of *NWI*.
Since in this mode, the filter runs on a stream-by-stream basis, we need to clear the register chain after a stream of length *NX* has been processed. We have to let the last zero-valued pixel enter the chain, that is why we clear the chain after the last zero-valued pixel enters the chain. This is why we need a new state just for clearing the register.

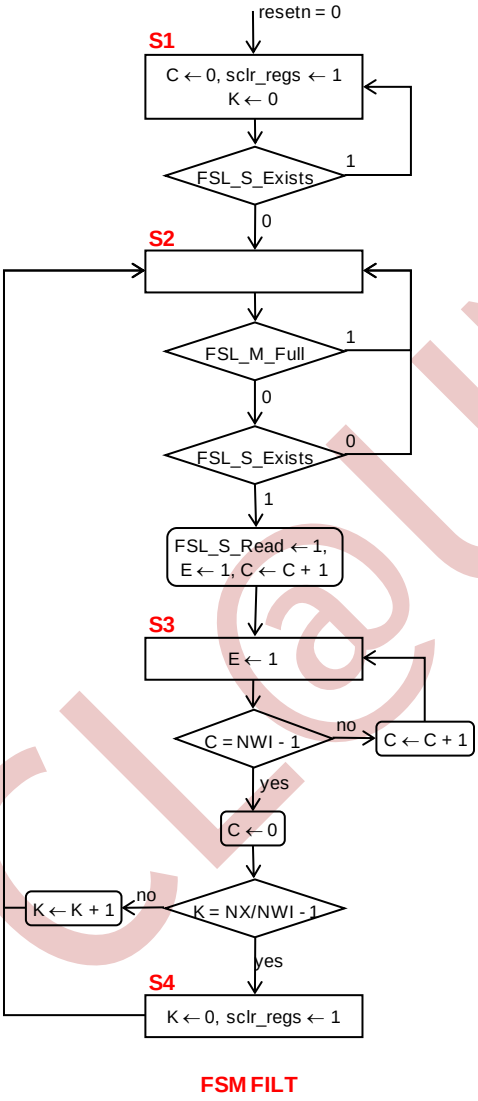


Figure 7. FSM for the FILT mode

FSMs for the mode CONV

Fig. 8 shows the FSM for the CONV mode. This FSM controls the input side. '`sclr_regs`' only clears the register chain inside the FIR filter (it does not clear other registers). Constraint: *NX* has to be a multiple of *NWI*.

NX-1 is not a multiple of NWI. Thus, ceil((N-1)/NWI) samples are output. Then, the ceil((N-1)/NWI)*NWI - N samples have to be discarded. This is because we process NWI samples at a time.

In this mode, the filter runs on a stream-by-stream basis. We thus need to clear the register chain after a stream of length *NX+N-1* has been processed. We have to let the last zero-valued pixel enter the chain, that is why we clear the chain after the last zero-valued pixel enters the chain. This is why we need a new state just for clearing the register. `E_out` is generated in the same way as it was generated for the mode IMG.

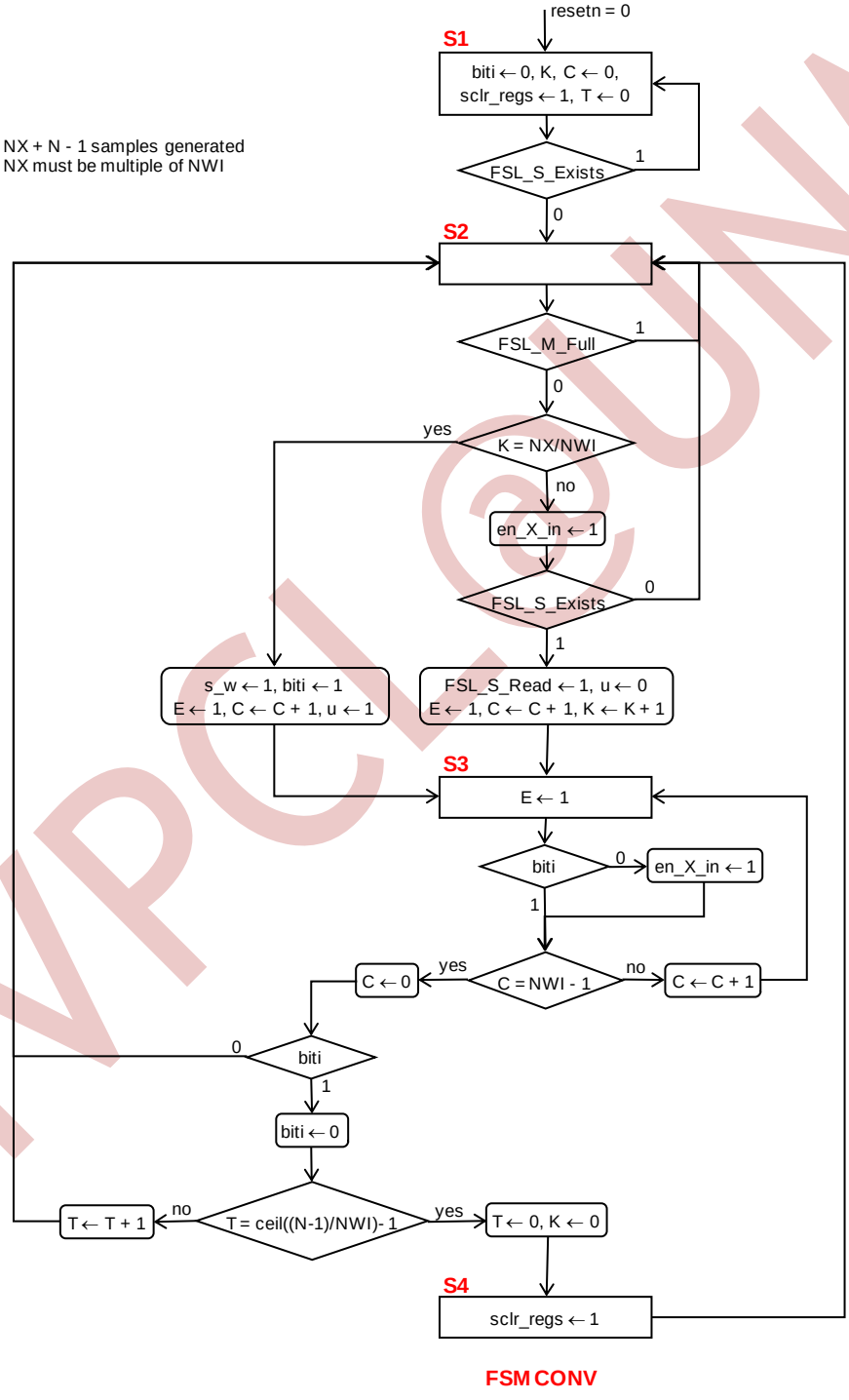


Figure 8. FSM for the CONV mode.