An Extended Cell Set for Self-Timed Designs

Ajay Khoche

UUCS-93-003

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

February 26, 1993

Abstract

it The high level synthesis approach described in [1] uses hopCP[2] language for behavioral descriptions. The behavioral specifications are then translated into Hop Flow Graphs (HFGs). The actions in the graph are then refined such that refined actions can be directly mapped onto asynchronous circuit blocks. This report describes library of such blocks called action-blocks. The action blocks use two phase transition signalling protocol for control signals and bundled protocol for data signals. The blocks have been designed using ViewLogic Design tools.

[1]

[1]This research was sponsored in part by NSF, award no. MIP 8902558.

The high level synthesis approach described in [1] involves compilation of hopCP specifications into asynchronous circuits by deriving inplementation for every action in the corresponding Hop Flow Graphs(HFGs). There are three types of actions in a typical HFG.

1. **Control Actions:** These actions denotes the synchronization.
2. **Data Actions:** These actions involve value communication in addition to synchronization.
3. **Expression Actions:** These actions represent computations.

All these actions are mapped onto a circuit abstraction called action block. These action blocks are classified into three categories

1. **Primitive Action Blocks -** These blocks denote the leaf cells of the compiler e.g. C element, Merge, Register.
2. **Predicate Action Blocks -** These blocks denote the conditional expressions.
3. **Function Action Blocks -** These blocks implement various functions like ADD, SUB, SHIFT etc.

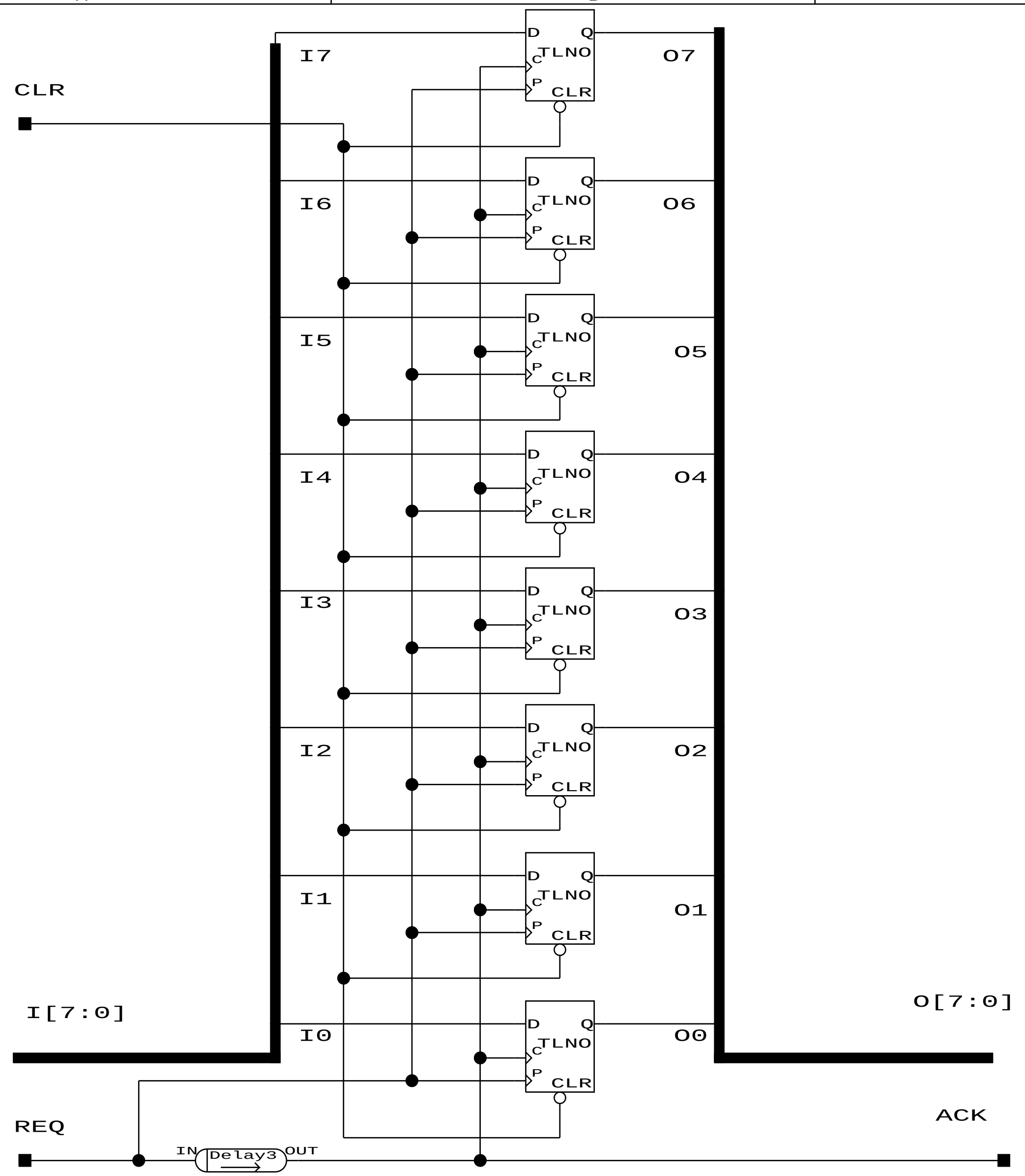
All the action blocks are selftimed blocks which use two phase signalling for the control signals and bundled protocol for data signals. Some of the primitive action blocks like C element, Merge, Call, Select, Toggle etc. have been described in [3] hence are not described here.
The report is organized as follows. For each block its symbol is shown in the summary page. The schematics for all level of hierarchies are given with the textual information about its inputs, outputs and iyts function.This indicates no. of logic modules used by that schematic at all levels. The exact no. of Actel modules used can only be found using Action Logic System(ALS) but no. of logic modules divided by three serves as good approximation for Actel Module count usage.

Register

Inputs : I[7:0], WREQ, RREQ, CLR

Ouput : O[7:0], WACK, RACK

Function : This is a read/write register. The I[7:0] is bundled data. When a request comes on WREQ line the data on I lines is latched in the register and an acknowledge is produced on WACK line. When a read request comes the data is available on O[7:0] lines and a read acknowledge is produced on the RACK line. Upon clear the register is set to zero. The register is madeup from transition lateches normally opaque(TLNOs).



Async Library

8-bit latch
with clr

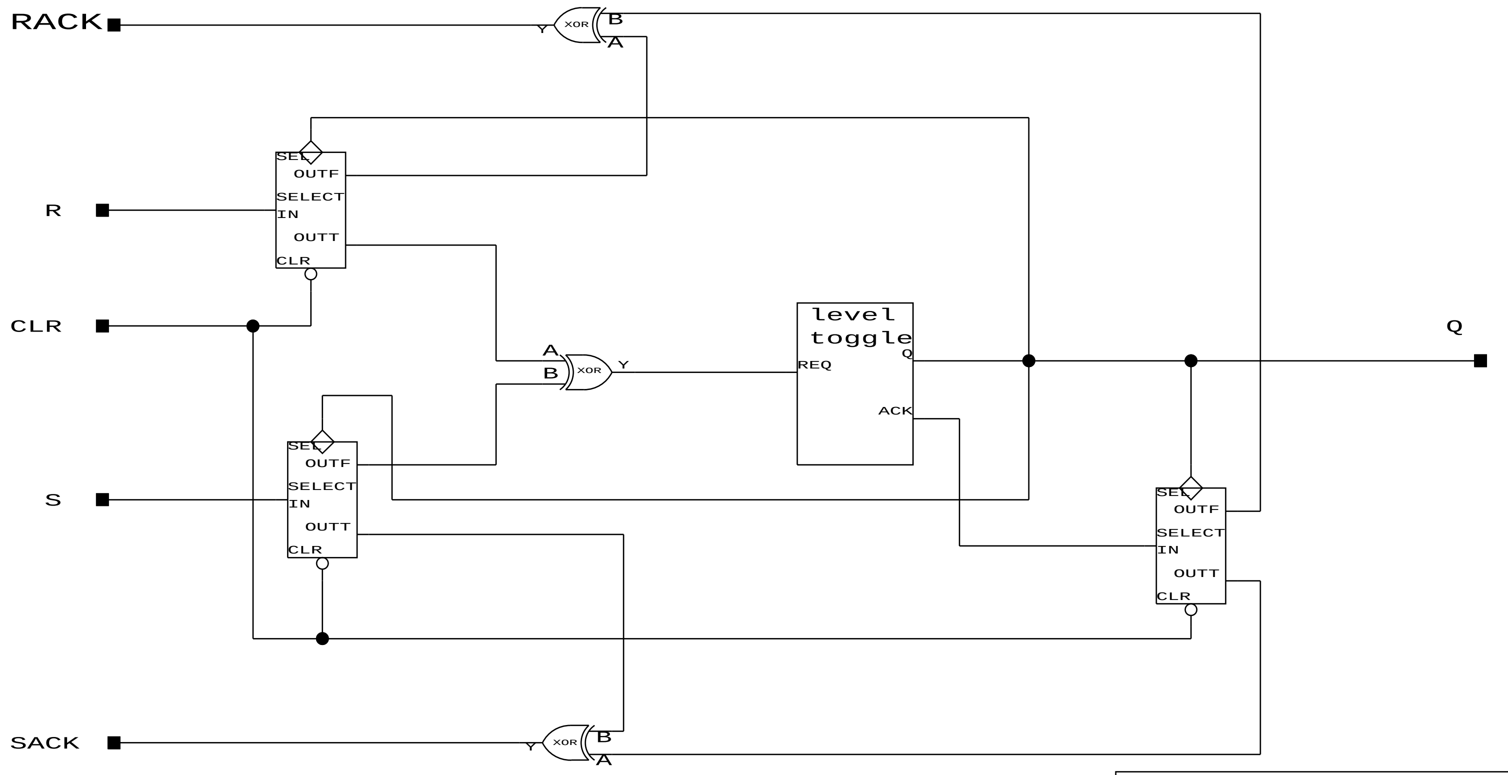
SHEET:	DATE:	DRAWN BY: AVK

Boolean Register(Breg)

Inputs : S, R, CLR

Ouput : Q, SACK, RACK

Function : This is a single bit storage element which can be set or reset by transitions. When A transition on S line occurs the register is set to high and an acknowledge is produced on SACK line. Similarly when a transition occurs on R line the register value is reset to low and an acknowledge is produced on the RACK line. Upon clear the register value is set to zero.



no of logic modules 63

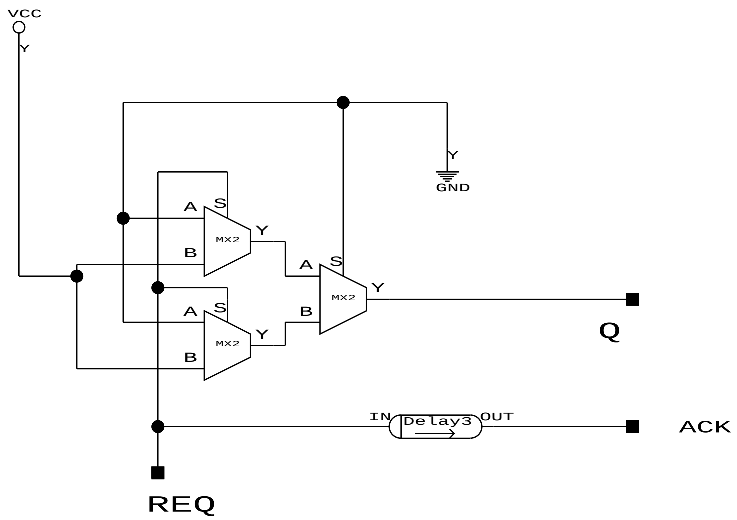
Async Library

BREG

SHEET:

DATE: 10/24/91

DRAWN BY: AVK



Async Library

level toggle

SHEET:

DATE:

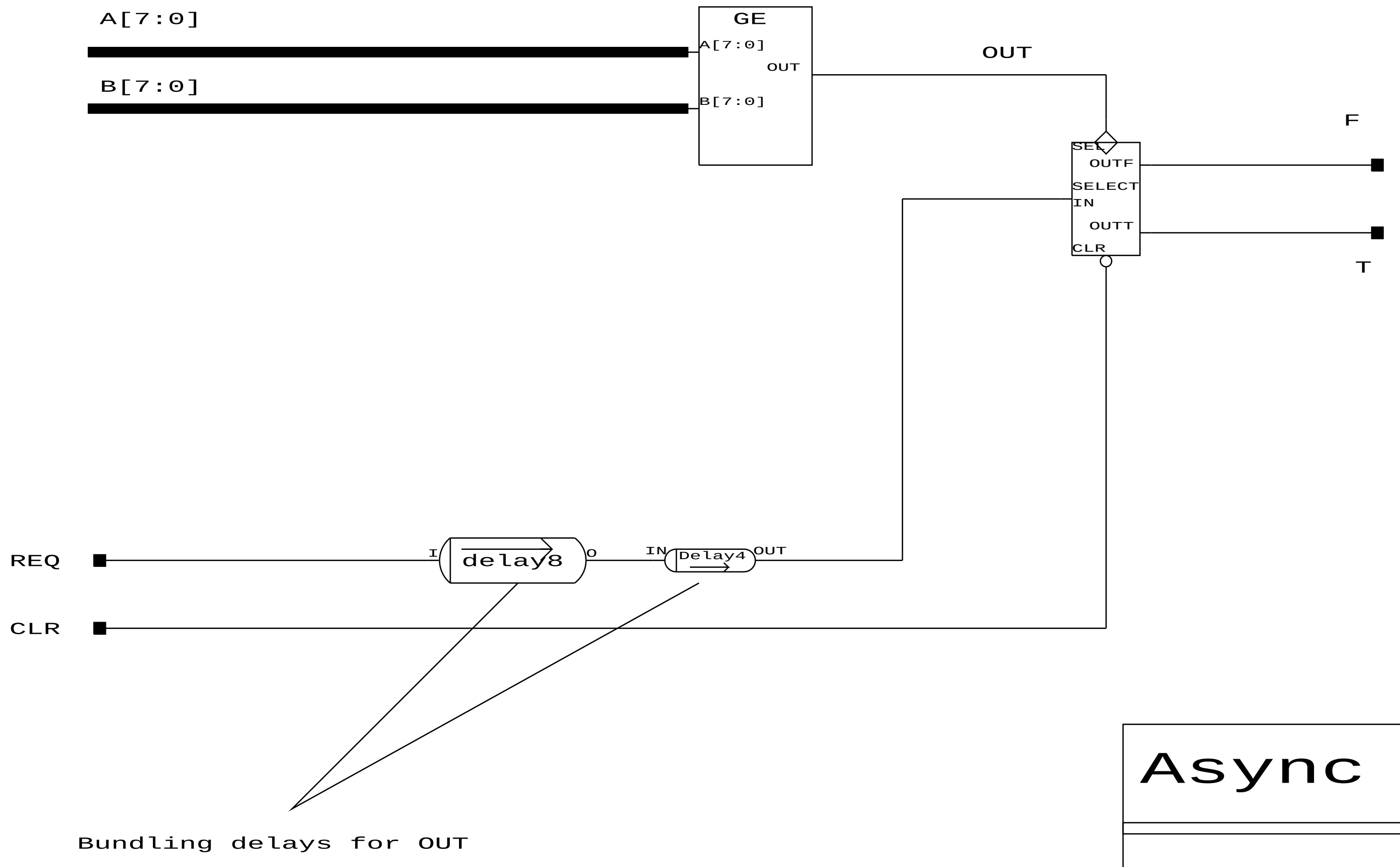
DRAWN BY: AVK

Greater than or Equal to(GE)

Inputs : A[7:0], B[7:0], REQ, CLR

Ouput : T, F

Function : This block tests for greater than or equal condition for bundled data inputs A and B. When A transition on REQ occurs the condtion is evaluated and depending on the result of condition check a transition is produced on either T or F. If the condition is true a transition is produced on T otherwise on F. All outputs are set to low upon clear.

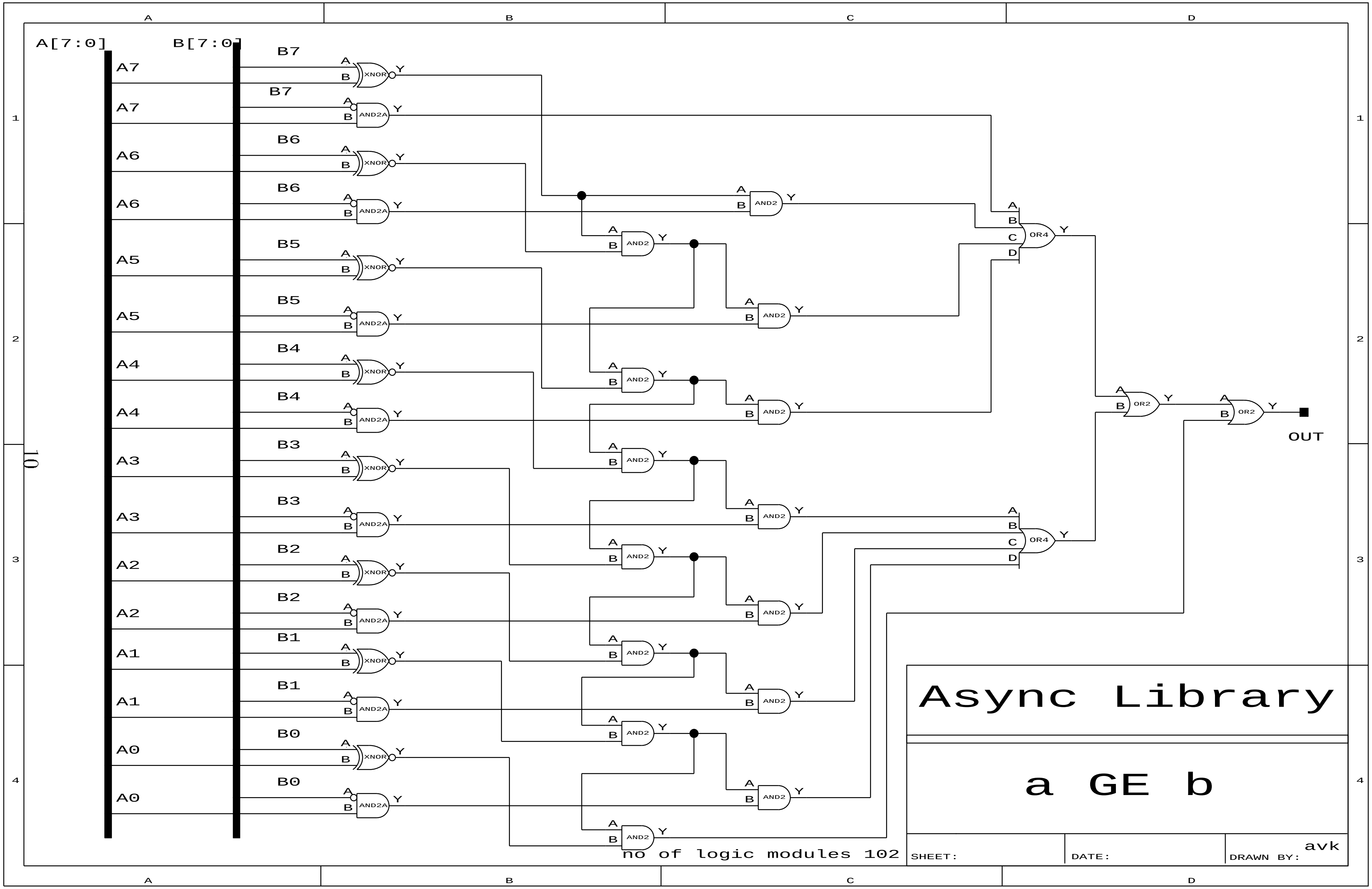


no of logic modules 150

Async Library

a GE b

SHEET:	DATE:	DRAWN BY: avk

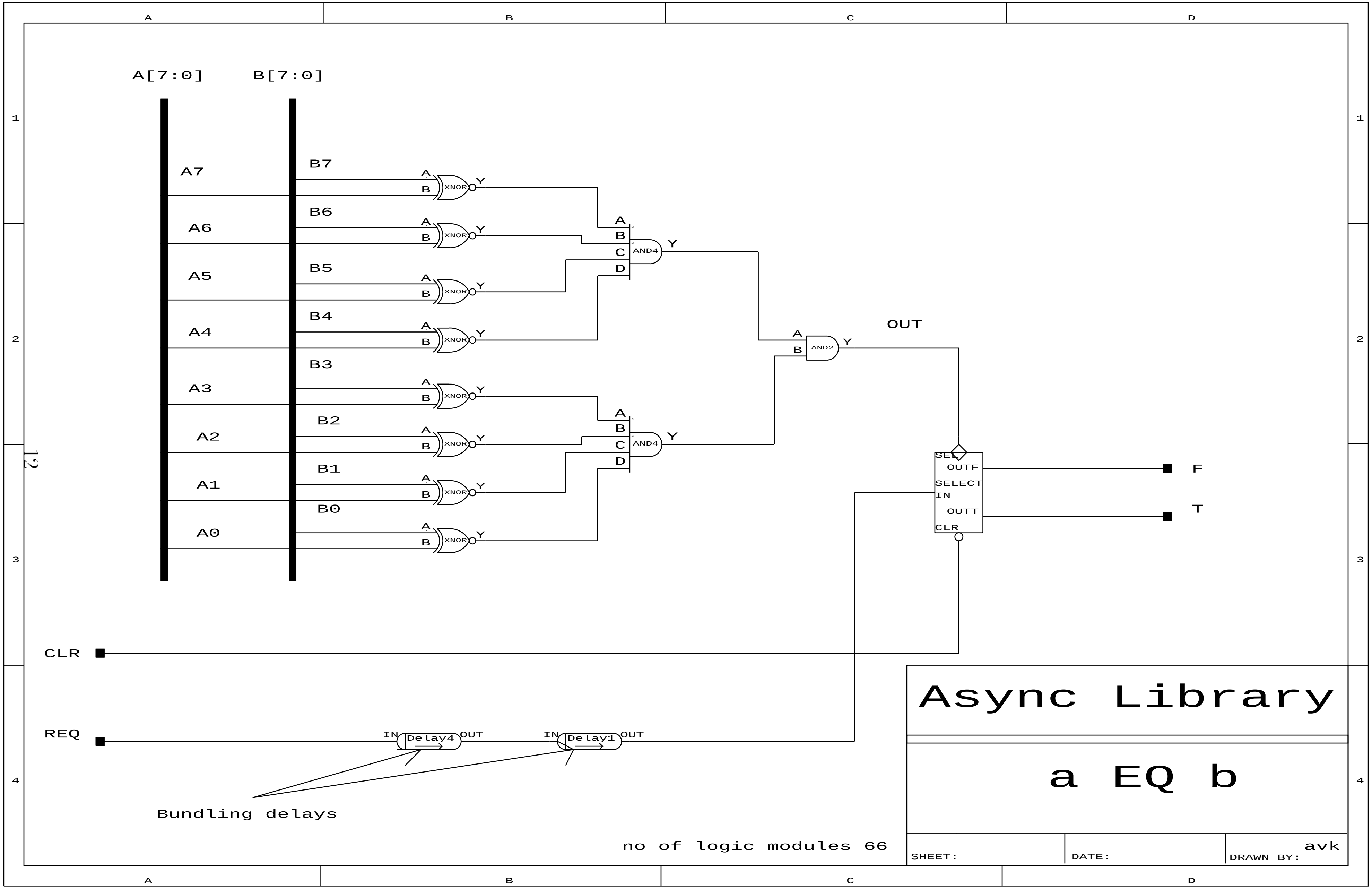

A[7:0]
B[7:0]
A7
B7
A6
B6
A5
B5
A4
B4
A3
B3
A2
B2
A1
B1
A0
B0
XNOR
AND2A
AND2
OR4
OR2
OUT
Async Library
a GE b
no of logic modules 102
SHEET:
DATE:
DRAWN BY:
avk

Equal (EQ)

Inputs : A[7:0], B[7:0], REQ, CLR

Ouput : T, F

Function : This module checks for the equality condition between bundled data inputs A and B. A transition on T is produced if the condition evaluates to true otherwise a transition is produced on F. When CLR is pulled low all the outputs are set to low.

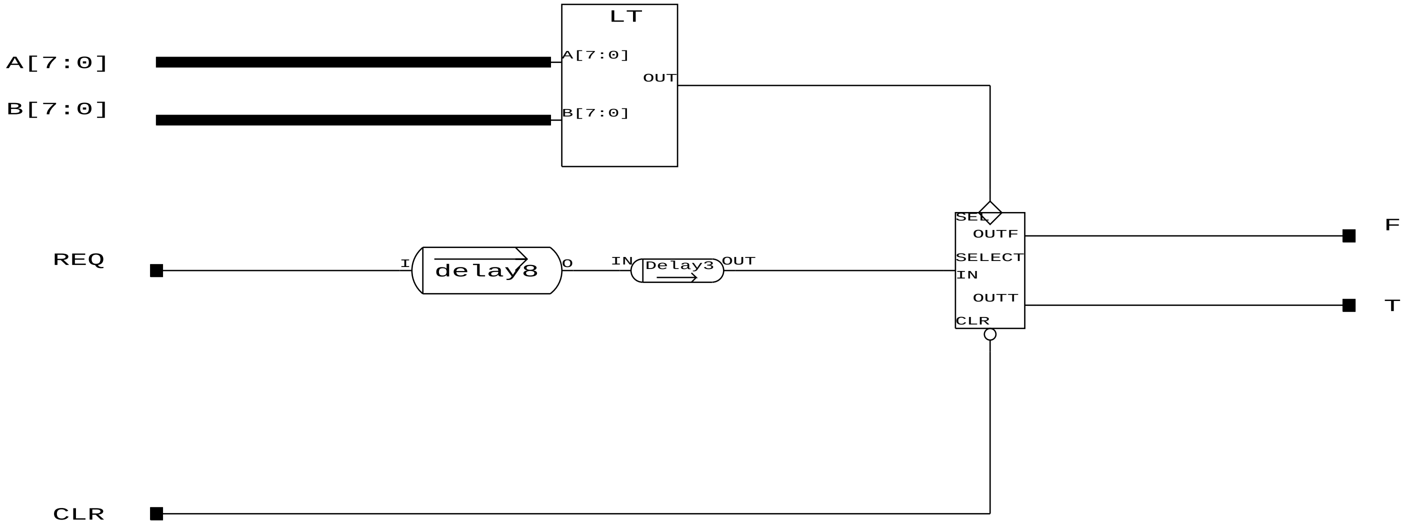

A[7:0]
B[7:0]
A7
A6
A5
A4
A3
A2
A1
A0
B7
B6
B5
B4
B3
B2
B1
B0
XNOR
AND4
AND2
OUT
SEL
OUTF
SELECT
IN
OUTT
CLR
F
T
CLR
REQ
Delay4
Delay1
Bundling delays
no of logic modules 66
Async Library
a EQ b
SHEET:
DATE:
DRAWN BY:
avk

Less Than (LT)

Inputs : A[7:0], B[7:0], REQ, CLR

Ouput : T,F

Function : This module checks for A Greater than B condition. When a transition on REQ input occurs the condition is tested and if it evaluates to true then a transition on T is produced otherwise a transition on F output is produced. Upon clear all the outputs are set to low.

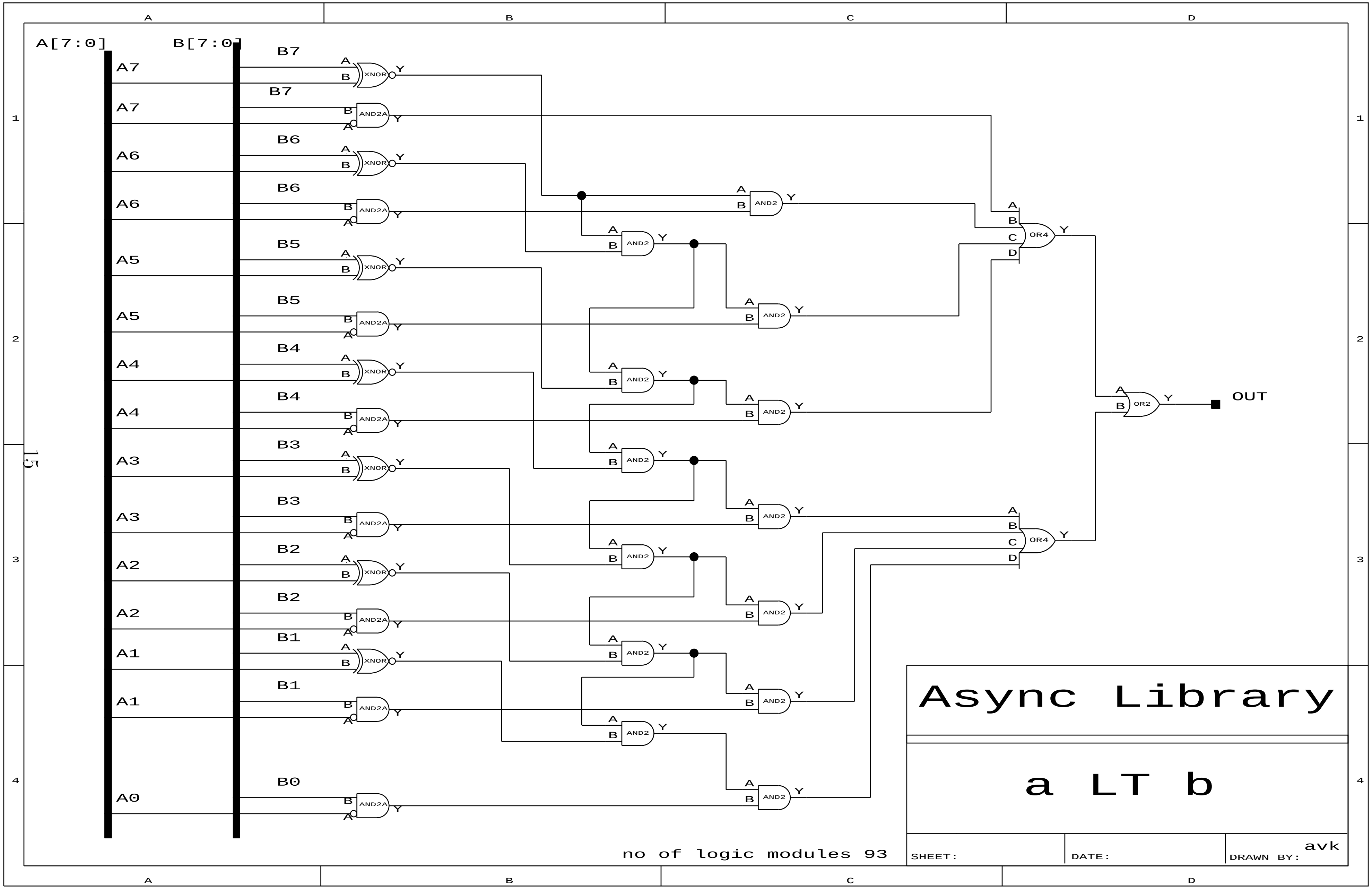


Async Library

a LT b

no of modules 138

SHEET: DATE: DRAWN BY: avk

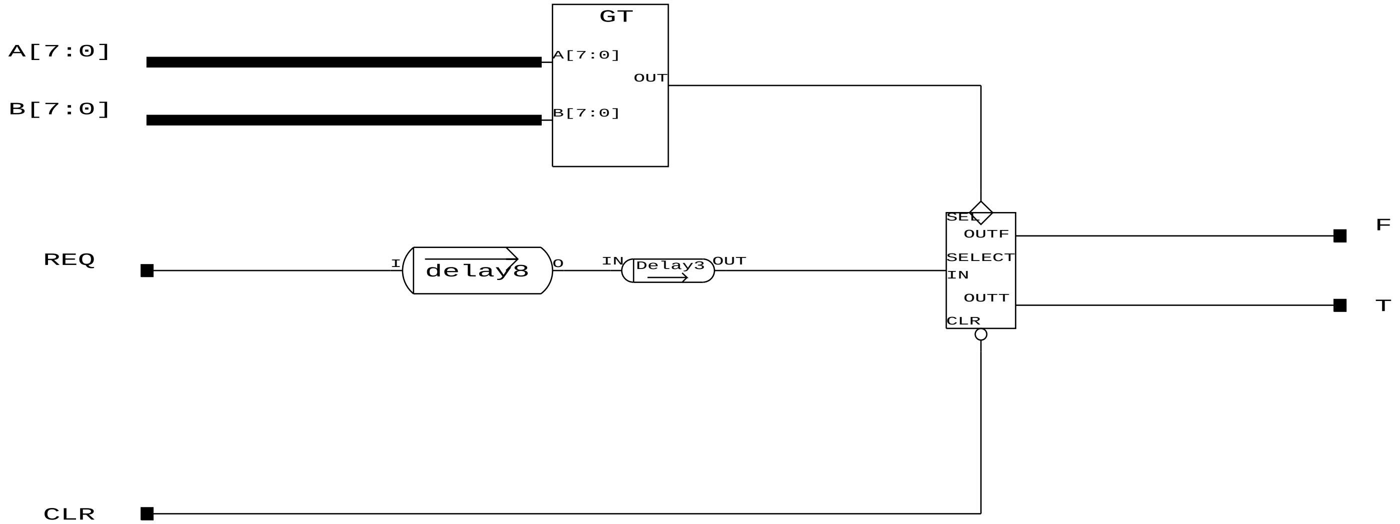

A[7:0]
B[7:0]
A7
B7
A6
B6
A5
B5
A4
B4
A3
B3
A2
B2
A1
B1
A0
B0
XNOR
AND2A
AND2
OR4
OR2
OUT
Async Library
a LT b
no of logic modules 93
SHEET:
DATE:
DRAWN BY:
avk

Greater Than (GT)

Inputs : A[7:0], B[7:0], REQ, CLR

Ouput : T, F

Function : This module checks for A less than B condition. When a transition on REQ input occurs the condition is tested and if it evaluates to true then a transition on T is produced otherwise a transition on F output is produced. Upon clear all the outputs are set to low.

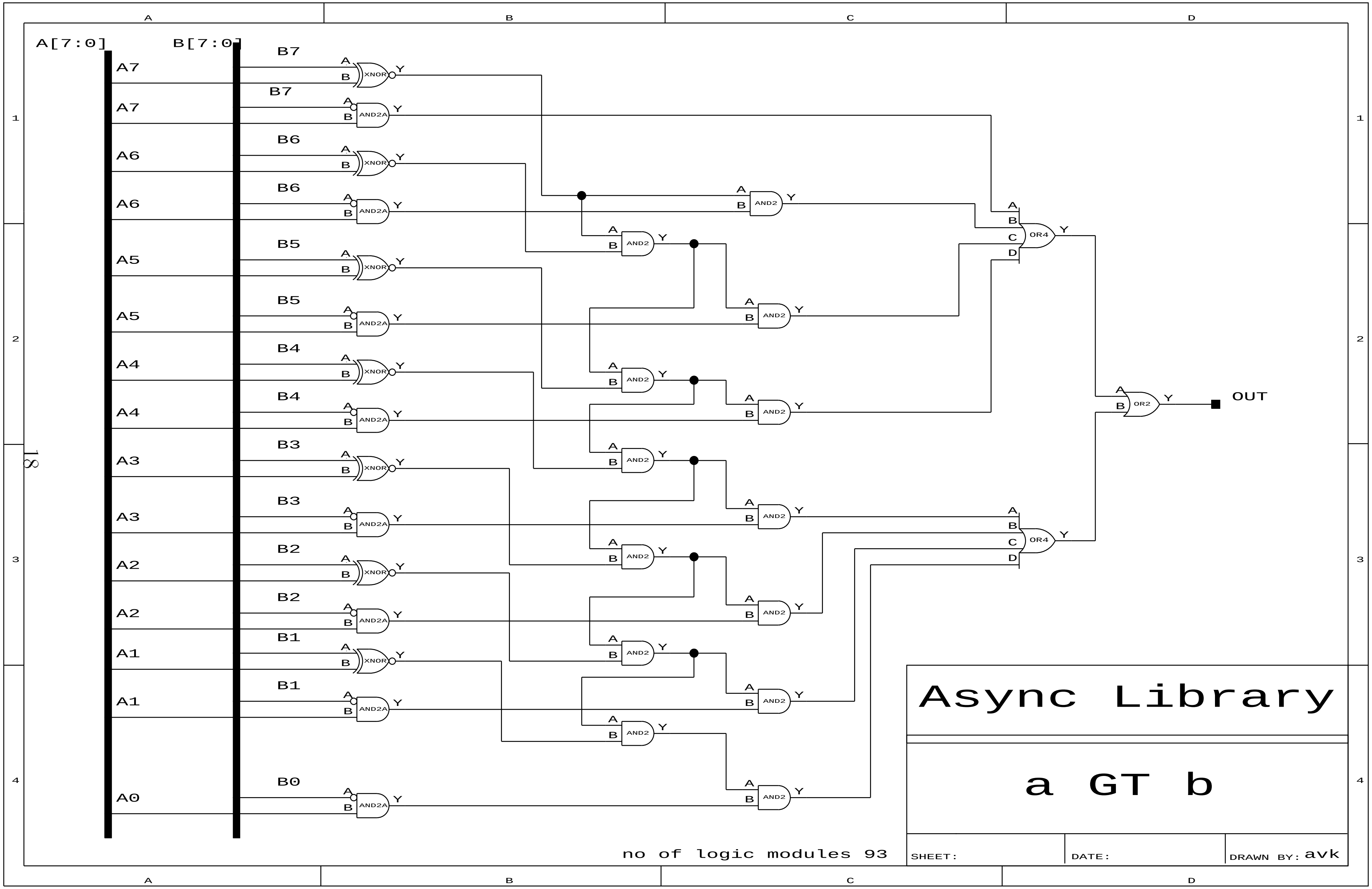


no of modules 138

Async Library

a GT b

SHEET: DATE: DRAWN BY: avk

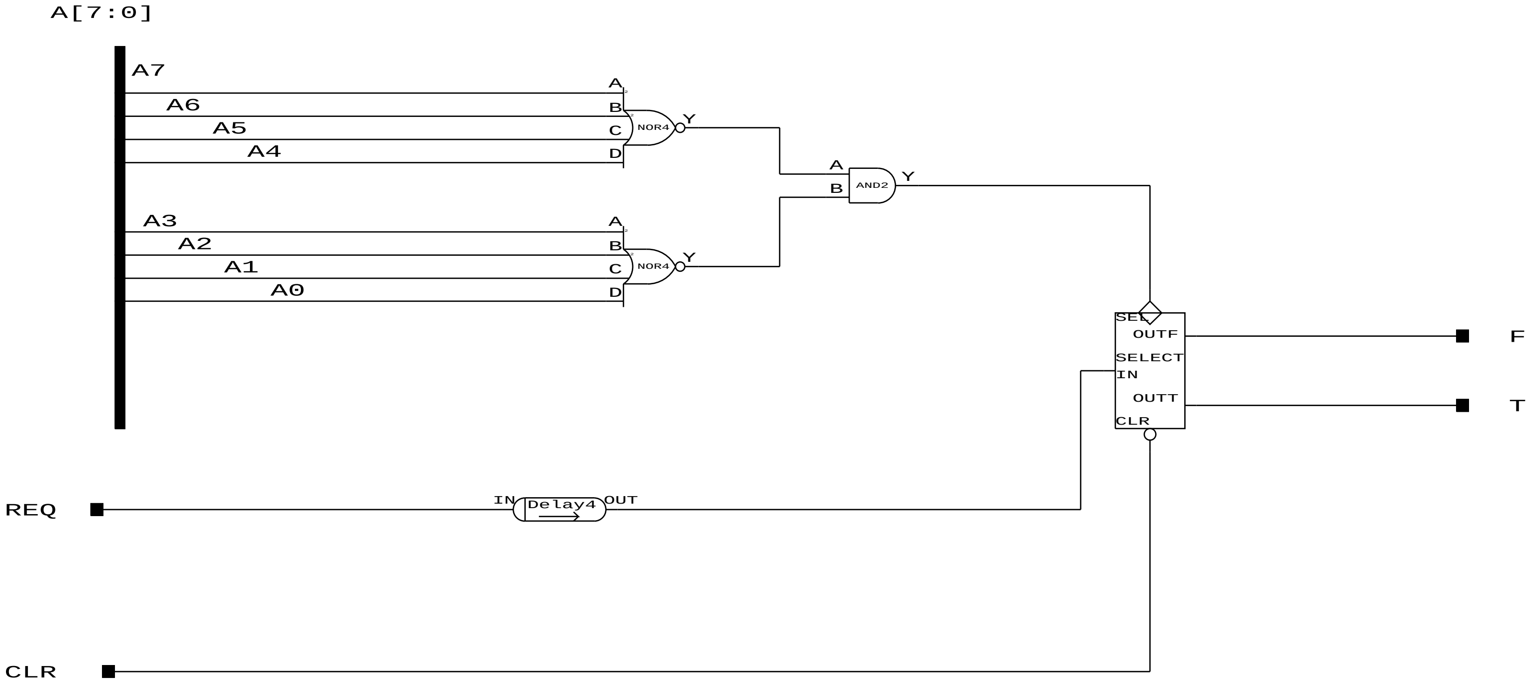

A[7:0]
B[7:0]
OUT
Async Library
a GT b
no of logic modules 93
SHEET:
DATE:
DRAWN BY: avk

ZERO

Inputs : A[7:0], B[7:0], REQ, CLR

Ouput : T, F

Function : This module tests for its operand to be zero. When as transition on REQ input occurs the condition is tested and if it evaluates to true then a transition on T output is produced otherwise a transition on F output is produced.



Async Library

a = 0

no of logic modules 39

SHEET:

DATE:

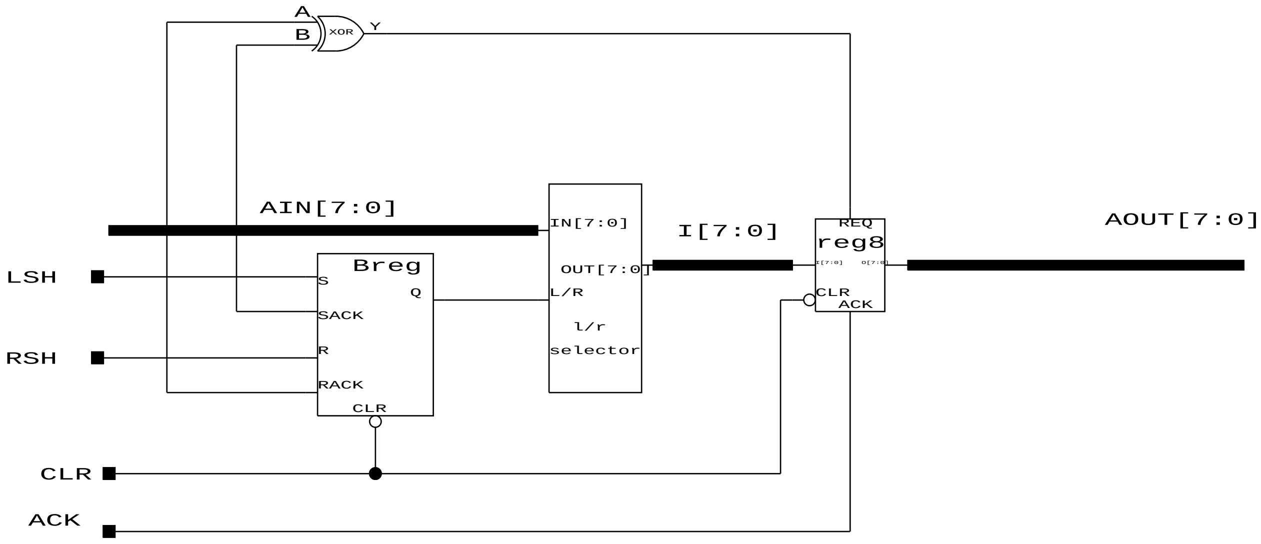
DRAWN BY: avk

Left/Right Shifter

Inputs : LSH, RSH, AIN[7:0], CLR

Ouput : AOUT[7:0], ACK

Function : This module shifts the data left or right by one bit depending on which input gets the transition. When a transition occurs on LSH line the bundled data AIN is left shifted by one bit and an acknowldge is produced on ACK line. Similarly when a transition occurs on RSH line the bundled data AIN is right shifted one bit and an acknowledge is produced on ack line.The shifted output occurs on the AOUT lines in both the cases. A low on CLR lines reset all the output to zero.

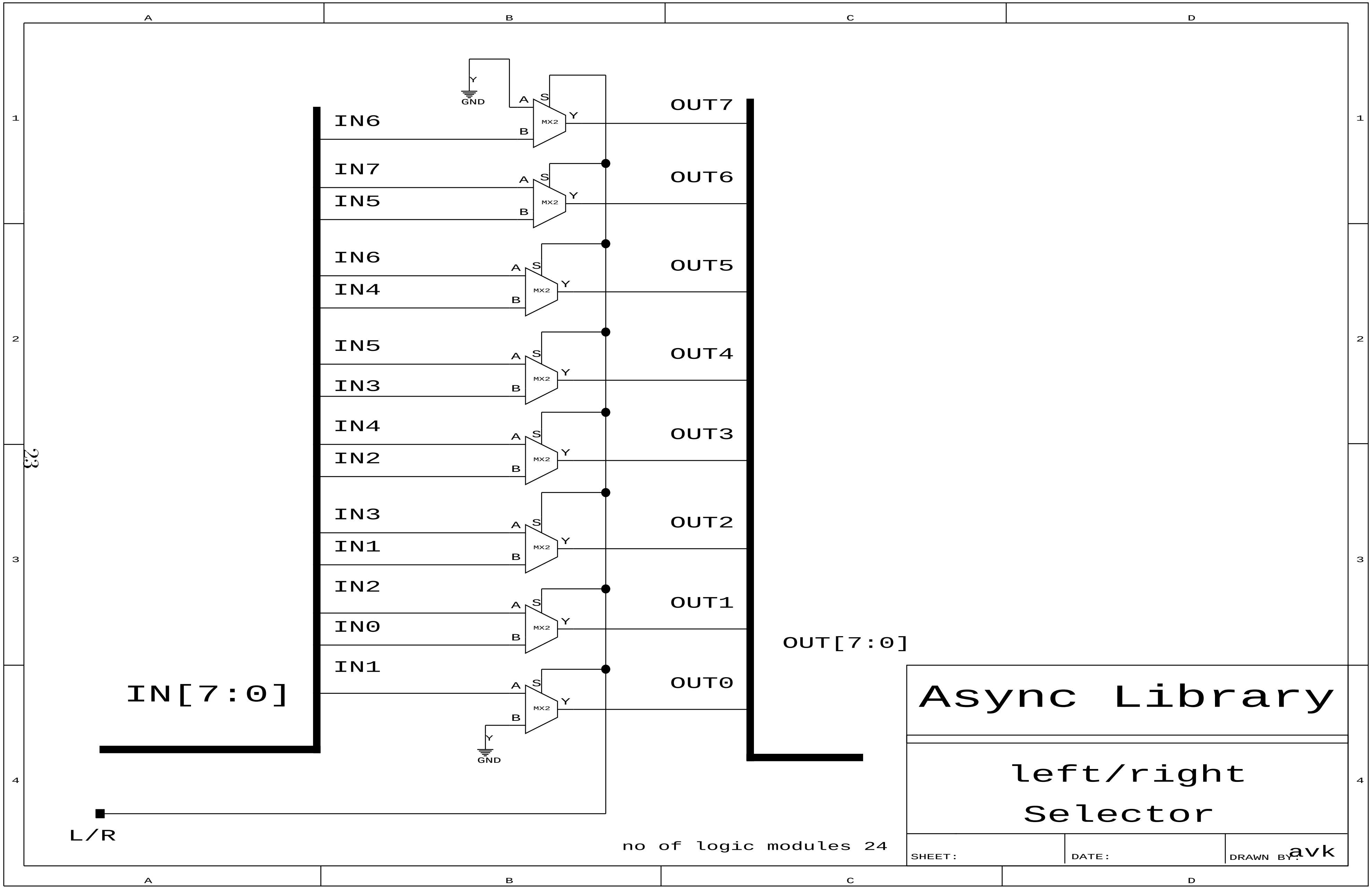


no of logic modules 147

Async Library

l/r shifter

SHEET: DATE: DRAWN BY: avk

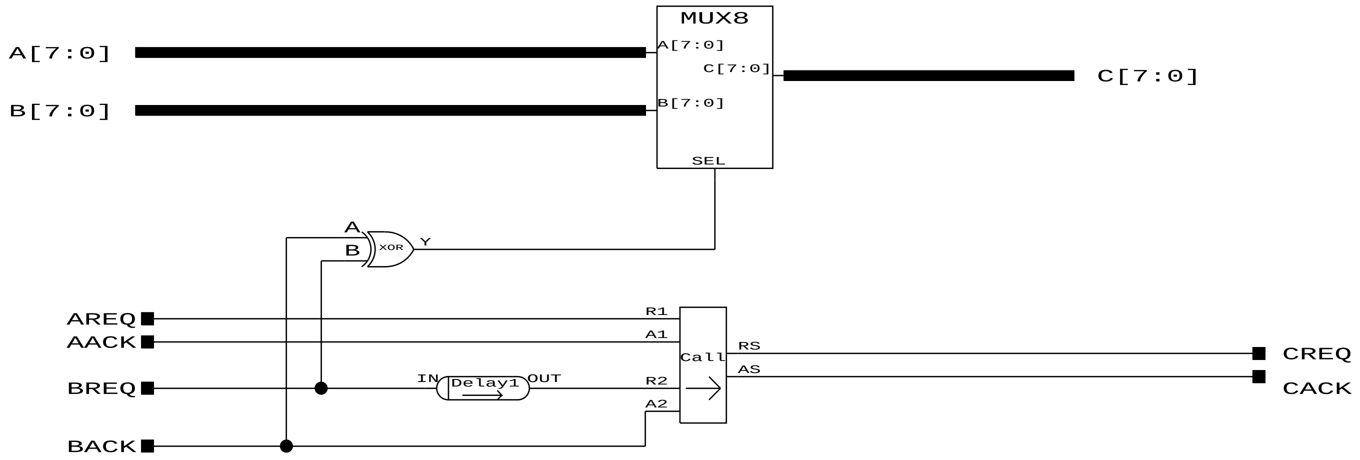

GND
IN6
IN7
IN5
IN6
IN4
IN5
IN3
IN4
IN2
IN3
IN1
IN2
IN0
IN1
IN[7:0]
MX2
OUT7
OUT6
OUT5
OUT4
OUT3
OUT2
OUT1
OUT0
OUT[7:0]
L/R
no of logic modules 24
Async Library
left/right
Selector
SHEET:
DATE:
DRAWN BY: avk

Asynchronous Multiplexer(AMUX)

Inputs : AREQ, A[7:0], BREQ, B[7:0], CACK

Ouput : AACK, BACK, CREQ, C[7:0]

Function : This is a transition multiplexer. The inputs are multiplexed on the output when corresponding requests arrive. When a transition on AREQ occurs the bundled data A is put on the output C and a transition is produced on CREQ. When this request is acknowledged on CACK an acknowledge is produced on AACK. Similarly when a transition occurs on BREQ the B input is put onn the output and a transition on CREQ is generated and when an acknowledge signal occurs on CACK an acknowldge is produced on BACK. The input request should be mutually exclusive.



Note : The delay-1 element has been used to compensate for the delay of XOR gate.

no of logic modules 45

Async Library

AMUX

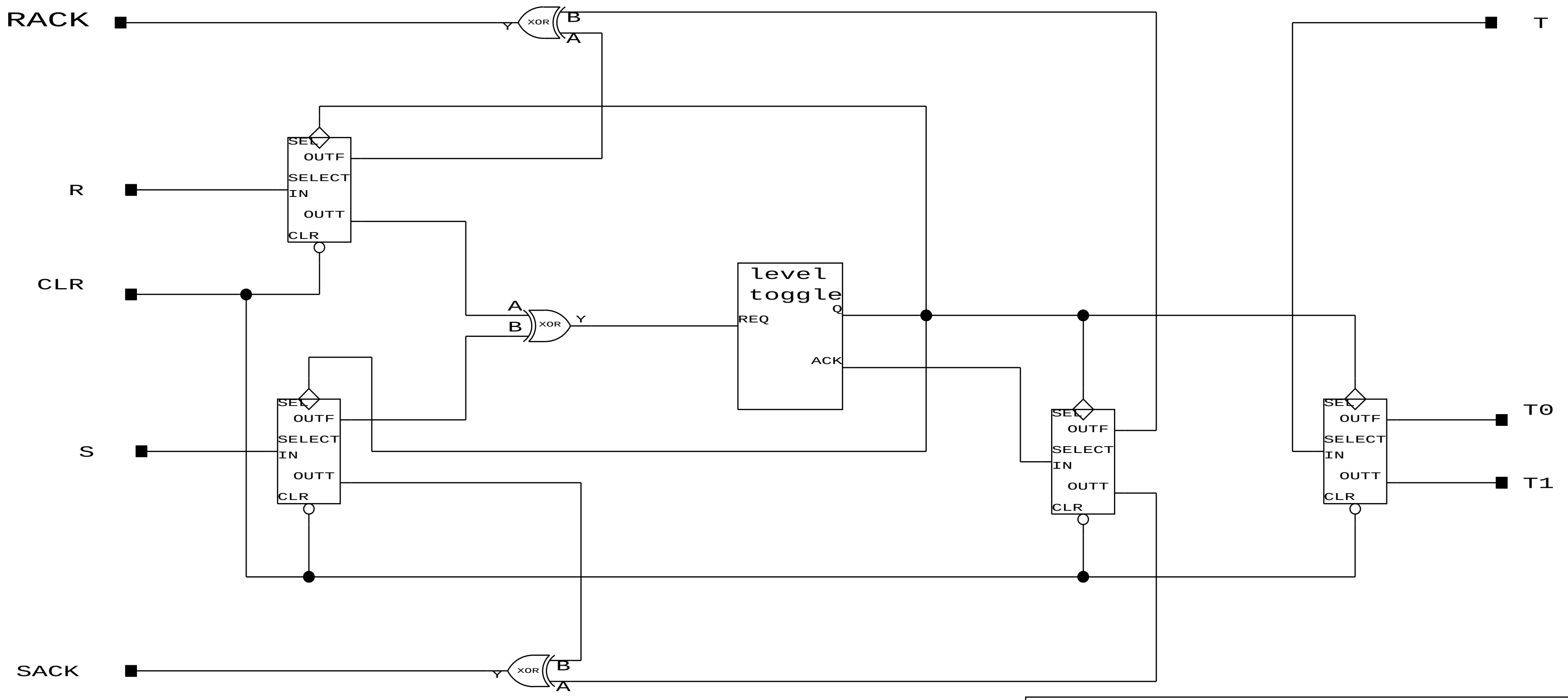
SHEET: DATE: DRAWN BY: avk

Test and Set

Inputs : S, R, T, CLR

Ouput : SACK, RACK, T0, T1

Function : This module is a single bit flag register which can be set or reset and later can be probed for the stored value. When a transition on S input occurs the flag is set to high and when a transition on R occurs it is reset to low. When a transition on T occurs depending on the value of the flag a transition on T0 or T1 is generated. If the flag value is high then a transition occurs on T1 otherwise on T0. At clear the flag and all outputs are reset to low. The set and reset request should not occur at the same time as no arbitration is provided in the circuit.



no of logic modules 75

Async Library

TS

SHEET:

DATE: 10/24/91

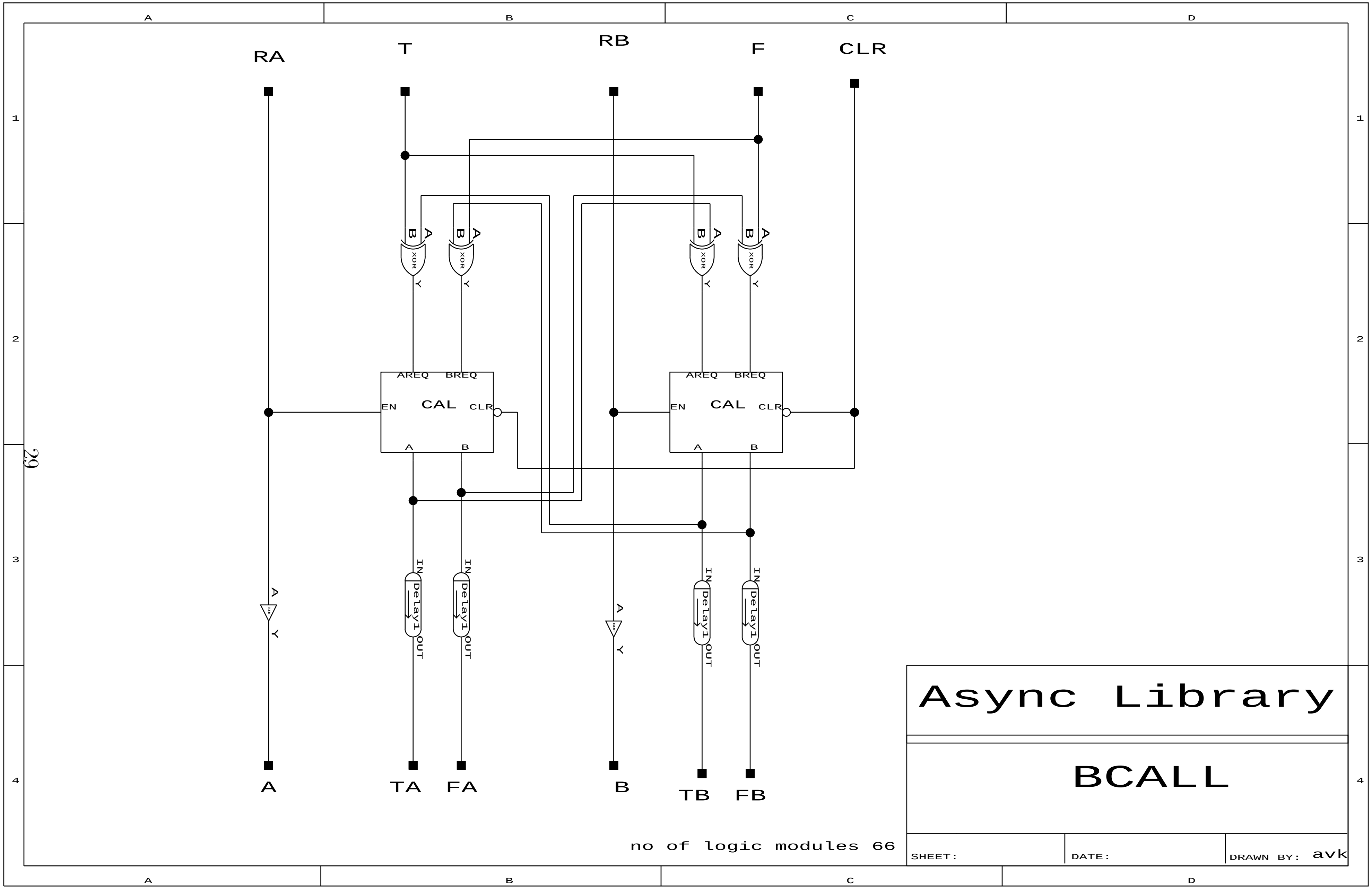
DRAWN BY: AVK

Boolean Call (BCall)

Inputs : RA, RB, T, F, CLR

Ouput : A, B, TA, FA, TB, FB

Function : This module is modified Call module. It allows one to call one of the two modules to be called. The called module is a Predicate Action Block(PAB) in this case, which returns true or false transition. the returned value of the call is then routed to the appropriate caller. If a transition on request line RA occurs then PAB A is called by generating a transition on output A. The called module produces a transition on T or F inputs depending on the predicate evaluates to true or false. This returned value produced on the TA or FA outputs depending on which input T or F get a transition. Similarly when a request occurs on RB input the PAB B is called and the result are routed to TB or FB.

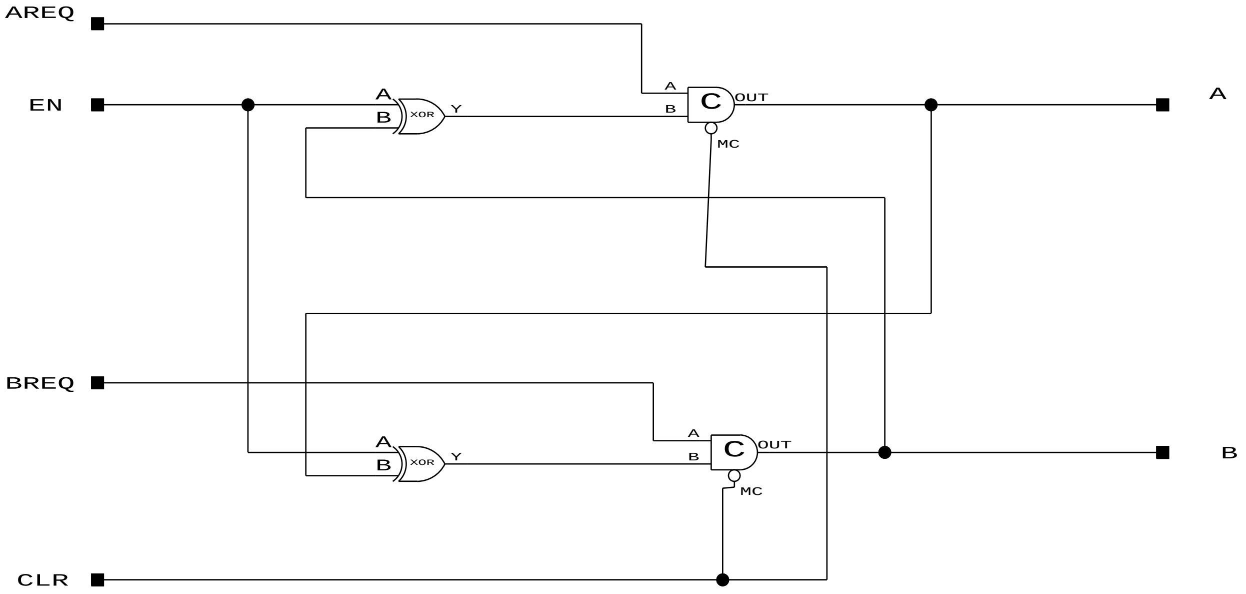

RA
T
RB
F
CLR
XOR
AREQ
BREQ
CAL
EN
CLR
A
B
IN
Delay1
OUT
A
TA
FA
B
TB
FB
Async Library
BCALL
no of logic modules 66
SHEET:
DATE:
DRAWN BY: avk

CAL

Inputs : AREQ, BREQ, EN, CLR

Ouput : A, B

Function : This module is a generalized C element. When a transition on AREQ and EN occurs a transition is produced on output A, similarly When a transition on BREQ and EN occurs, a transition is produced on output B. The input request AREQ and BREQ should not occur at the same time. The request and EN transitions may come in any order. Upon clear all the outputs and internal stateis set to low. To observe the above behavior all the inputs must be in same state.



no of logic modules 18

Async Library

CAL

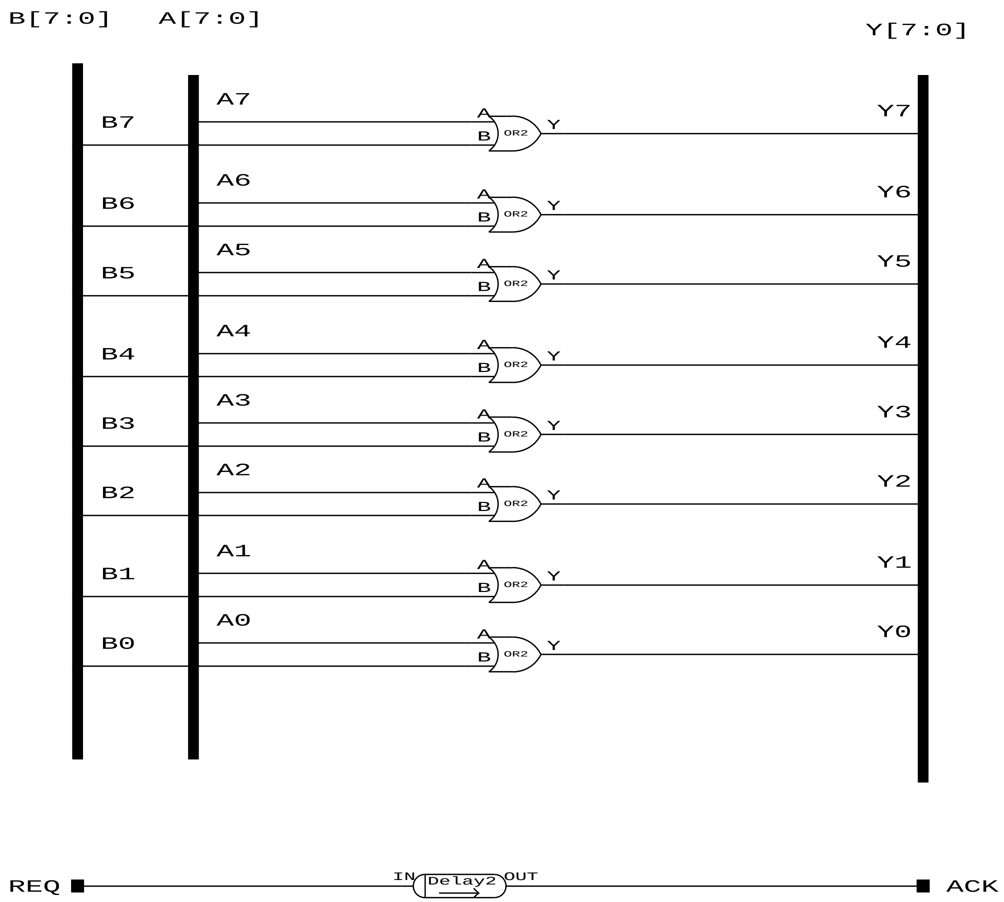
SHEET: | DATE: | DRAWN BY: avk

OR

Inputs : A[7:0], B[7:0], REQ

Ouput : Y[7:0], ACK

Function : This module performs logical OR of the operands upon getting a transition on the REQ input and produces an acknowledge transition on ACK output.



no of logic modules 30

Async Library

B bit OR

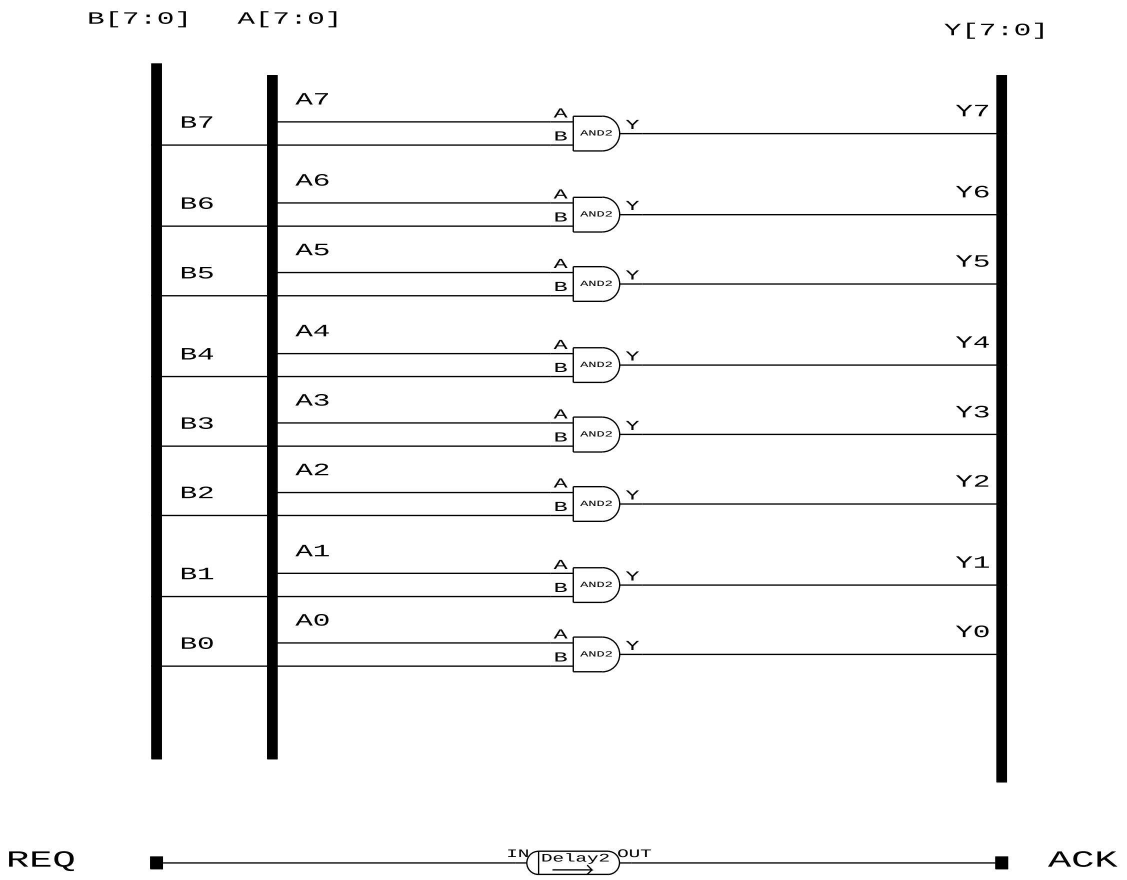
SHEET:	DATE:	DRAWN BY: avk

AND

Inputs : A[7:0], B[7:0], REQ

Ouput : Y[7:0], ACK

Function : This module performs logical AND of the operands upon getting a transition on the REQ input and produces an acknowledge transition on ACK output.



no of logic modules 30

Async Library

B bit AND

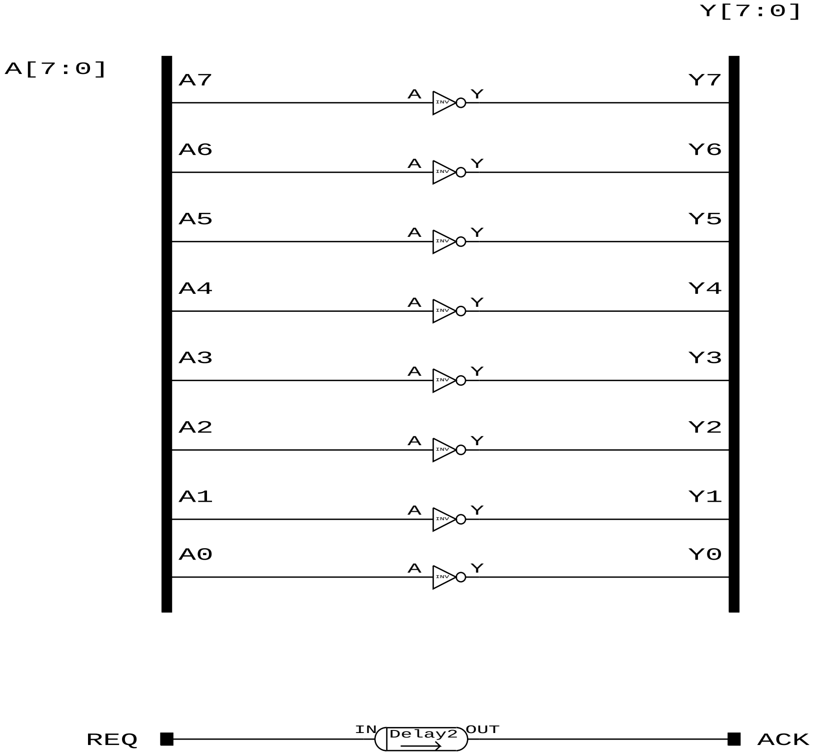
SHEET:	DATE:	DRAWN BY: avk

NOT

Inputs : A[7:0], REQ

Ouput : Y[7:0], ACK

Function : This module inverts the input upon transition on REQ input and produces acknowledge transition on ACK output.



no of logic modules 30

Async Library

8 bit inveter

SHEET:

DATE:

DRAWN BY: avk

References

1. Venkatesh Akella and Ganesh GopalKrishnan.*From Process Oriented Specifications to Efficient Asynchronous Circuits* Tech Report UUCS-91-XX, Dept. of Computer Science, University of Utah.

2. Venkatesh Akella. *hopCP: Language Definition, Semantics and Examples.* Tech Report UUCS-91-XX, Dept of Computer Science, University of Utah.

3. Erik Brunvand. *A cell Set for Self-Timed Design using Actel FPGAs.* Tech report UUCS-91-013, Dept. of Computer Science, University of Utah.