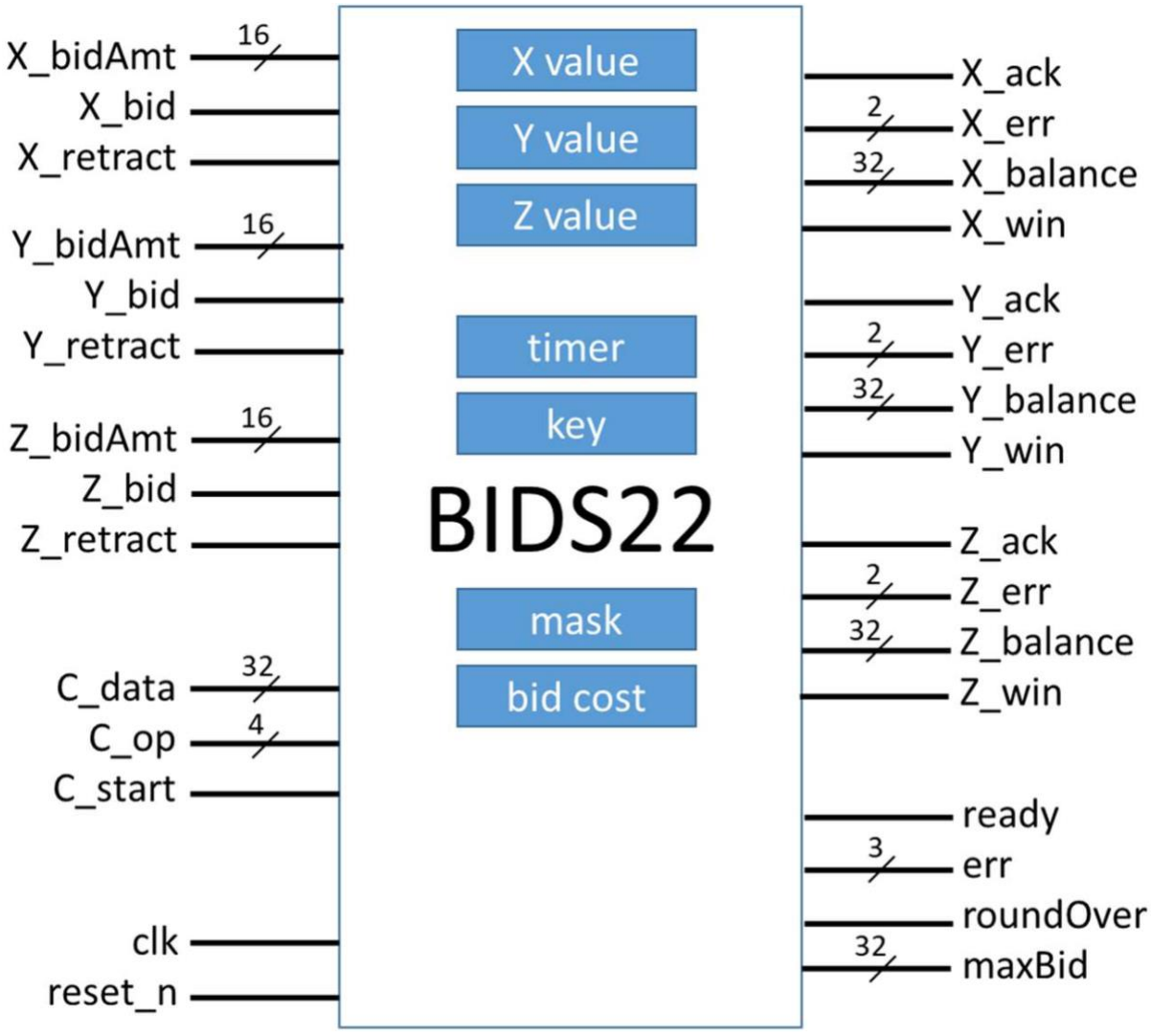
ECE – 593: Fundamentals of Pre-Silicon Validation
Assignment – 1

- **Design Plan:**

Assignment – 1 included coming up with a Reference Model for a BIDS22 Architecture. The Architecture had hardware components compete for resource over time. The components received a set of pre – allocated tokens that can be used by the components in turn to make bids to gain access to the architecture.

- **Block Diagram of the resource: (Reference – Dr. Schubert's slides)**



## ♦ Design – Specification's brief explanation:

The resource had three components namely X, Y, Z competing to gain access to the resource.

X, Y, Z bidders can use tokens to make bids.

The resource architecture has clock and reset_n input signals. The control input signals namely bid_Amt (bid amount) can be utilized to specify the amount bid by the respective bidder, bid signal is to specify initiating a bid and a respective acknowledge signal for indicating the bidder is allowed to bid an amount.

Retract signals can be used to retract an already issued bid, and win signal specifies which bidder won the round.

## ♦ Implementation:

Implemented a package called bids22defs. The package contained typedef enum logic definitions specifically for global errors of the output that dealt with the errors associated with getting a bad key, attempting to unlock when already in the unlocked state, start signal asserted when in the unlocked state and another typedef enum definitions specifically for bidder errors that include invalid request, insufficient funds, no bid, round inactive error and so on.

Implemented a packed structs definitions for input signals (inputs_t) of the bidders that included bid amount, bid and retract signal and output signals (outputs_t) of acknowledge, win, error and balance for each bidder.

Implemented an interface that imported the package definition, defined bidders' input and output signals, defined a modport called bidmaster that defined input bidder signals, output bidder signals and few other input, output signals to accept data or acknowledge the bidder signals.

## bids22 – actual bids22's FSM Code.

Defined extra internal FSM Registers for timer, storing the last bid, cooldown-timer, cooldown-timer value, key, bidcost, and the mask.

Enum definitions for defining states namely Reset, Unlocked, Locked, Cooldown, Roundstarted, Roundover and Ready Next.

Parameterized the number of bidders and which was 3 in our case.

* X = `bidder[0]`, Y = `bidder[1]`, Z = `bidder[2]`

Implemented an always flip flop block for sequential logic:

If negative edge triggered reset was high, FSM would be in the Unlocked state, all the bidders would get a value 0 assigned to them and their respective last bid.

All the FSM internal miscellaneous registers would get their default values.

If the reset condition turned out to be false, it would execute a sequence of case statements.

**RESET:** Failed state, FSM should not end up there.

**UNLOCKED:** The cooldown timer should be reset so that next incorrect attempt is set to the correct cooldown value.

Within the Unlocked state itself, there will be multiple case statement block where the states are triggered depending upon the opcode.
The opcode decides if one of the following states would be executed:

**LOCK:** The internal register Key will be assigned value from the control input C_data.

**LOADX / LOADY / LOADZ:** The mentioned states will have X_value, Y_value, Z_value being assigned values from the the control input C_data.

**SETMASK:** The set mask state will have the mask register assigned value from C_data.

**SETTIMER:** The state will be assigning the cooldown timer value register the value from control input C_data.

**SETBIDCHARGE:** This state assigns bidcost register with data from control data input.

**COOLDOWN:** The state will keep decrementing the cool down timer value till it does not reach zero.

**LOCKED:** The locked state has no logic.

**ROUNDSTARTED:** In this state, it is checked whether the bid signal and mask for a particular bidder is high, indicating that the bid cannot be masked out.
A nested if statement is then checked for the respective bidder has its value greater than the amount bid and the bid cost endured, the bid value is subtracted with the bid cost and the bid amount is recorded in the last bid register.

Else if the retract signal is high for the respective bidder, the last bid register is assigned zero.

**ROUNDOVER:** In this state, if the bidder has been declared a winner, the bidder value is deducted with the last bid amount.

**READYNEXT:** Redundant state for disambiguating future references.

## Next State Logic:

Always comb block implemented for next state logic.
Case statement logic implemented with the following states –
**RESET:** Redundant state for disambiguating future references.

**UNLOCKED**: If the C_op is equal to LOCK, the next state would be LOCKED or it will go to UNLOCKED.

**COOLDOWN:** If cool down timer is not equal to 0, next state will be COOLDOWN or next state will be LOCKED.

**LOCKED:** If the C_start signal is asserted, next state will be ROUNDSTARTED, else if the C_op is equal to UNLOCK and C_data is not equal to Key (Bad Key) next state will be COOLDOWN, else next state will be LOCKED.

**ROUNDSTARTED:** If C_start is asserted, next state will be ROUNDSTARTED or will be ROUNDOVER.

**ROUNDOVER:** Next state will be READYNEXT.

**READYNEXT:** Next state will be LOCKED.

## Output Logic:

Always comb block implementation for case statements.

**RESET:** should never enter failed state, throw error message.

**UNLOCKED**: If C_start asserted in this state throw error signal. And if in UNLOCKED state, internal case statement will be executed and in it whilst in UNLOCK state, error message for when C_op will be be unlock, also test for invalid opcode.

**COOLDOWN:** Error for bad key being presented.

**LOCKED:** Redundant state for disambiguating future references.

**ROUNDSTARTED:** For a particular bidder having its bid high and its mask being 0, the request gets masked out and a bidder error for invalid request gets generated and acknowledge signal is 0.
Else if the mask is 1, check for insufficient funds error where the bid amount and bid cost together is higher than the bidder value.

Also check for no bid error, update maxbid after every cycle, check for duplicate bids.

**ROUNDOVER:** If the maxbid is lesser than last bid, declare particular bidder as the winner.

**READYNEXT:**

## ♦ TESTPLAN:

- Test the functioning for the design.
- Test the working of all fsm state transitions.
- Test for decrementing timer value not reaching 0 or going out of bounds.
- Test for maximum and minimum values for bid amounts of all bidders.
- Test for repeatedly giving bad key.
- Test for out of bounds value assigned to internal registers.
- Test for when X, Y, and Z bid the highest amount.
- Test for bid cost being deducted properly.
- Test for global errors: